Das Nagios/Icinga Kochbuch

Book · August 2013

CITATIONS
0

READS
8,193

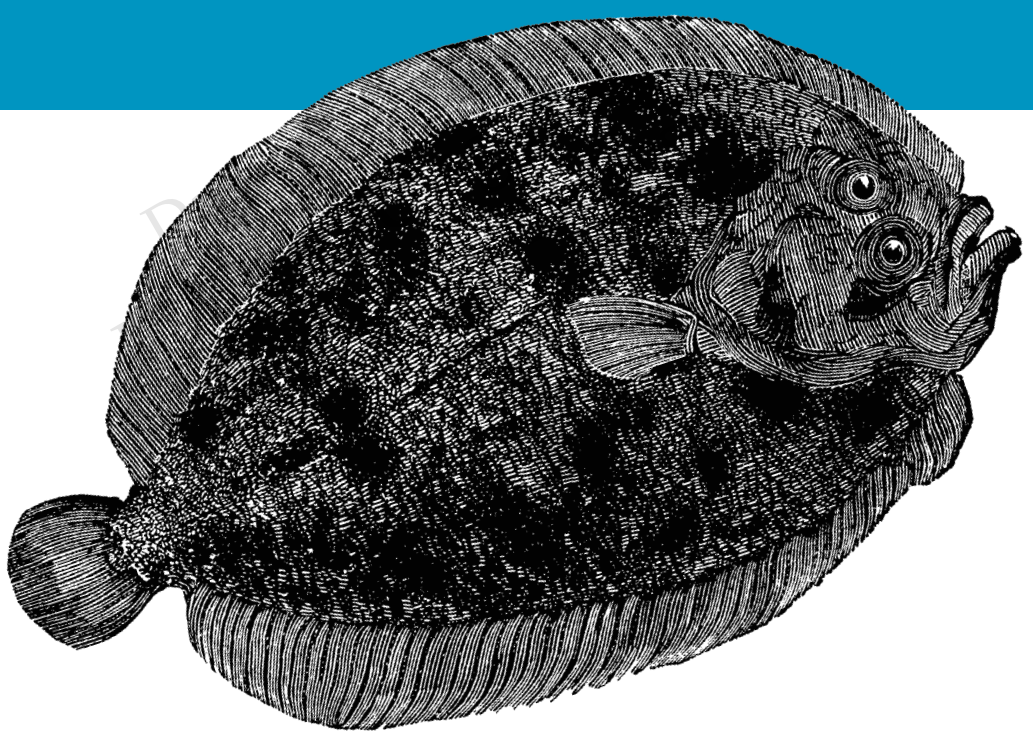
2 authors, including:

Ralf C. Staudemeyer
University of Applied Sciences Schmalkalden
34 PUBLICATIONS **414** CITATIONS

SEE PROFILE

Systemmonitoring vom Feinsten

Das Nagios/Icinga *Kochbuch*



Timo Kucza &
Ralf Staudemeyer

O'REILLY®

1. Auflage

Das Nagios/Icinga-Kochbuch

Timo Kucza & Ralf Staudemeyer

O'REILLY®
Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

Kommentare und Fragen können Sie gerne an uns richten:
O'Reilly Verlag
Balthasarstr. 81
50670 Köln
E-Mail: kommentar@oreilly.de

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

Lektorat: Volker Bombien, Köln
Fachliche Unterstützung: Michael Friedrich, Netways GmbH, Nürnberg
Korrektorat: Tanja Feder, Bonn
Satz: Tim Mergemeier, Reemers Publishing Services GmbH, Krefeld; *www.reemers.de*
Produktion: Andrea Miß
Belichtung, Druck und buchbinderische Verarbeitung:
Druckerei Kösel, Krugzell; *www.koeselbuch.de*

ISBN 978-3-86899-346-2

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Nicht der Beginn wird belohnt,
sondern einzig und allein das Durchhalten.

Katharina von Siena

Inhalt

Danksagungen XI

Einleitung XIII

1 Nagios/Icinga installieren und Hostsystem vorbereiten 1
1.1 Entscheidungshilfe für die Installation 3
1.2 Netzwerkanbindung des Monitoring-Servers planen 7
1.3 Hardware-Anforderungen abschätzen 9
1.4 Icinga aus den Quellen installieren 12
1.5 Icinga unter Debian installieren 29
1.6 Icinga unter Ubuntu installieren 44
1.7 Icinga unter SLES|openSUSE installieren 56
1.8 Icinga unter RHEL|CentOS|Fedora installieren 69
1.9 Nagios aus den Quellen installieren 78
1.10 Nagios unter Debian installieren 93
1.11 Nagios unter Ubuntu installieren 104
1.12 Nagios unter SLES|openSUSE installieren 113
1.13 Nagios unter RHEL|CentOS|Fedora installieren 123
1.14 Migrationen und Aktualisierungen von Nagios/Icinga 133
1.15 Icinga unter Debian deinstallieren 136
1.16 Icinga unter Ubuntu deinstallieren 138
1.17 Icinga unter SLES|openSUSE deinstallieren 140
1.18 Icinga unter RHEL|CentOS|Fedora deinstallieren 141
1.19 Nagios unter Debian deinstallieren 142
1.20 Nagios unter Ubuntu deinstallieren 144
1.21 Nagios unter SLES|openSUSE deinstallieren 146
1.22 Nagios unter RHEL|CentOS|Fedora deinstallieren 147

2 Basiskonfiguration der Systeme 149
2.1 Absichern der Webschnittstelle zum Monitoring-Server mit SSL 149
2.2 Plugins zur Ausführung mit administrativen Rechten vorbereiten 155
2.3 SNMP auf Linux-Maschinen vorbereiten 158
2.4 SNMP mit MIBs verwenden 162
2.5 Plugins unter Linux über SNMP ausführen 166
2.6 SNMP auf Windows-Maschinen vorbereiten 168
2.7 NRPE auf Unix-Maschinen vorbereiten 174
2.8 NRPE und NSClient auf Windows-Maschinen vorbereiten 179
2.9 SSH auf Linux-Maschinen vorbereiten 187

3 Die Arbeit an und mit Nagios/Icinga 194
3.1 Benutzerverwaltung für Zugriffe auf die Weboberfläche 194
3.2 Das Konzept von aktiven und passiven Prüfungen 198
3.3 Nutzung der Weboberflächen von Nagios und Icinga-Classic 203
3.4 Nutzung der Weboberfläche Icinga-Web 220
3.5 Vereinfachen der Konfiguration mit dem Vorlagen-System 233
3.6 Aufbau und strukturierte Ablage der Konfigurationsdateien 237
3.7 Konfigurationsdateien ändern und nach Abhängigkeiten durchsuchen . . 243
3.8 Prüfen der Konfigurationsdateien auf Fehler 247
3.9 Sicherungen der Konfigurationsdateien anlegen 248
3.10 Fehlern mit Debug- und Logdateien nachgehen 254
3.11 Maßgeschneiderte Benachrichtigungen einrichten 257
3.12 Den Flattern-Status verstehen und nutzen 261
3.13 Kommentare, Acknowledgments und Downtimes nutzen 264

4 Die Konfigurationsobjekte von Nagios/Icinga 268
4.1 Maschinen einbinden (host) 270
4.2 Dienste: Befehle mit Maschinen verknüpfen (service) 276
4.3 Befehle hinzufügen (command) 281
4.4 Kontakte definieren (contact) 286
4.5 Zeitperioden definieren (timeperiod) 290
4.6 Maschinen zu Gruppen zusammenfassen (host-group) 293
4.7 Services zu Gruppen zusammenfassen (service-group) 295
4.8 Kontakte in Gruppen zusammenfassen (contact-group) 296
4.9 Abbilden von Abhängigkeiten (host- und service-dependencies) 298
4.10 Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen 303
4.11 Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo) 305

5 Lokale Parameter mit Standard-Plugins überwachen 311
5.1 Anzahl angemeldeter Benutzer überwachen (check_users) 311
5.2 Prozessorauslastung überwachen (check_load) 314
5.3 Prozesse überwachen (check_procs) 317
5.4 Dateisysteme überwachen (check_disk) 322
5.5 Festplatten-Status überwachen (check_ide_smart) 327
5.6 Sensor-Daten überwachen (check_sensors) 330
5.7 Log-Dateien überwachen (check_log) 336
5.8 Überwachung einer Maschine mit NRPE (check_nrpe) 339
5.9 Überwachung einer Windows-Maschine mit NSClient (check_nt) 344
5.10 Entfernte Ausführung über SSH (check_by_ssh) 350
5.11 Überwachen auf Aktualisierungen des Systems (check_apt) 355

6 Netzwerk-Dienste mit Standard-Plugins überwachen 359
6.1 Erreichbarkeit überwachen mit Ping (check_ping) 359
6.2 Erreichbarkeit überwachen mit ICMP (check_icmp) 363
6.3 Überwachung beliebiger IP-Serviceports (check_tcp/check_udp) 366
6.4 Überwachung eines Jabber-Servers (check_jabber) 372
6.5 Das DNS überwachen (check_dns) 376
6.6 DHCP überwachen (check_dhcp) 380
6.7 Erstellung generischer SNMP-Abfragen (check_snmp) 384
6.8 Schnittstellen über SNMP allgemein prüfen (check_ifstatus) 391
6.9 Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus) 396
6.10 Überwachung eines FTP-Servers (check_ftp) 400
6.11 Überwachung eines Webservers (check_http) 403
6.12 Überwachung eines SSH-Dienstes (check_ssh) 409
6.13 Überwachung des E-Mail-Versanddienstes (check_smtp) 412
6.14 Überwachung der E-Mail-Auslieferungsdienste (check_pop, check_imap) 415
6.15 Verzeichnisdienst LDAP überwachen (check_ldap) 418
6.16 SMB-Dateidienst überwachen (check_smb) 421
6.17 MySQL-Datenbanken überwachen (check_mysql) 425

7 Erstellung eigener und Einbindung externer Plugins 432
7.1 Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API) 433
7.2 Ein einfaches Beispiel eines benutzerdefinierten Plugins 437
7.3 Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen 440
7.4 Einbinden des PNP4Nagios-Plugins (check_pnp_rrds) 445
7.5 Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins 448

8 Überwachung von Maschinen, Diensten und Infrastruktur 457
8.1 Überwachung von Unix-/Linux-Servern 458
8.2 Überwachung von Windows-Maschinen 470
8.3 Überwachung von Hypervisoren und virtuellen Maschinen 486
8.4 Überwachung der aktiven Infrastruktur 491
8.5 Dateiserver überwachen 498
8.6 E-Mail-Server überwachen 501
8.7 Webserver überwachen 505
8.8 Datenbankserver überwachen 512
8.9 Zentrale Netzwerk-Dienste überwachen (DNS, DHCP, LDAP, AD) 518

9 Datenvisualisierung mit PNP4Nagios und NagVis 525
9.1 PNP4Nagios verwenden 526
9.2 Migration von Performancedaten bei Umbenennungen 532
9.3 Alte Graphen und Daten aus PNP4Nagios entfernen 535
9.4 Vereinen von PNP4Nagios-Graphen über Special Templates 537
9.5 Benutzerdefinierte Zusammenfassung von PNP4Nagios-Graphen 542
9.6 Verwendung von NagVis und Erstellung von Karten 545
9.7 Erweiterte Karten in NagVis (Wetterlinien und Gadgets) 552
9.8 Karten in NagVis zu Rotationspools zusammenfassen 557
9.9 Ausblick auf das kommende Icinga2 559

Index 561

Danksagungen

Wir möchten uns zunächst ganz herzlich bei all den Personen bedanken, die uns bei der Erstellung dieses Buches unterstützt haben. Dabei gab es ganz unterschiedliche Formen der Hilfe: von Inspiration über konstruktive Kritik bis hin zu Geduld und Verständnis. Gerade Familien und Freunde haben ertragen müssen, dass wir selbst im Urlaub bei Sonne und Pool in Diskussionen zu diesem Projekt verstrickt waren. Aufgrund der Vielzahl der betroffenen Personen verzichten wir auf eine namentliche Aufführung.

Danke Euch allen für Eure Geduld, Zeit und Energie!

Einleitung

Mit der zunehmenden Bedeutung der Informationstechnik in allen Lebensbereichen steigt auch die Bedeutung der Prüfung auf deren Verfügbarkeit und Funktionsfähigkeit. Wir haben uns mehrere Jahre mit dem Einsatz von Nagios im Umfeld größerer Organisationen beschäftigt, in denen Monitoring-Lösungen bereits seit vielen Jahren Standard sind. Heute setzen wir das dabei gewonnene Wissen aber auch für private Infrastruktur ein. Der eigene Server ist immer weniger ein Spezialfall von Informatikern, sondern auch für Privatleute zunehmend Standard. In kleineren Unternehmen, aber auch innerhalb größerer Organisationen wird das Thema der Infrastruktur-Überwachung häufig noch stark vernachlässigt. Im Bereich der privaten Infrastruktur ist es außer in Ausnahmefällen sogar noch völlig unbekannt.

Nachdem wir regelmäßig auf Unwissenheit gestoßen sind, was Netzwerkmonitoring allgemein und speziell das zunehmend von uns verwendete Icinga betrifft, haben wir uns deshalb entschlossen, dieses Buch zu schreiben. Wir berücksichtigen dabei sowohl Nagios als auch Icinga und vermitteln Ihnen das Wissen, welches Sie zur Inbetriebnahme und zum bedarfsgerechten Ausbau eines Monitoring-Systems benötigen. Das Buch richtet sich sowohl an Anfänger als auch fortgeschrittene Benutzer.

Aufbau und Verwendung des Buches

Das Format der Kochbücher ist zum Lesen von Anfang bis Ende nur begrenzt geeignet. Wir möchten Ihnen deshalb einleitend eine Hilfestellung geben, wie Sie am besten mit diesem Buch umgehen sollten. Der ursprünglichen Idee eines Kochbuches folgend, soll es insbesondere ein Nachschlagewerk sein, in dem Sie nach Bedarf eine Lösung für ein bestimmtes Problem nachschlagen. Als Anfänger werden Sie eher mit den Rezepten aus den ersten vier Kapiteln beginnen. Als fortgeschrittene Anwender werden Sie es bevorzugen, mit den Kapiteln 8 und 9 zu beginnen und fehlendes Wissen bei Bedarf nachzuschlagen. Im Folgenden stellen wir Ihnen die Kapitel zunächst im Überblick vor.

Die Kapitel im Überblick

Kapitel 1: Nagios/Icinga installieren und Hostsystem vorbereiten

Dieses Kapitel beginnt mit Rezepten zur Auswahl eines Systems für das Monitoring sowie Überlegungen zur Anbindung an das Netzwerk. Diese einleitenden Rezepte sollten Sie auf jeden Fall lesen, wenn Sie eine neue Installation planen. Im Rest des Kapitels führen Sie die einzelnen Rezepte dann durch die Installation und Deinstallation von Nagios und Icinga unter unterschiedlichen Distributionen sowie die Migration von bestehenden Installationen. Beachten Sie, dass dieser Teil nicht für ein Durchlesen von Anfang bis Ende geeignet ist. Entscheiden Sie sich mit Hilfe der einleitenden Rezepte für eine Distribution und lesen Sie dann nur die für Sie zutreffenden Rezepte.

Kapitel 2: Basiskonfiguration der Systeme

In diesem Kapitel stehen Ihnen Rezepte zur Vorbereitung der Systeme zur Verfügung. Wir gehen dabei sowohl auf die weitergehende Konfiguration des Monitoring-Servers ein als auch auf die Vorbereitung der zu überwachenden Maschinen unter Windows und Unix/Linux. Hier sollten Sie einmal querlesen, ansonsten werden wir Sie aus den aufbauenden Rezepten heraus aber auch auf die jeweiligen Rezepte hinweisen. Wenn Sie es eilig haben, können Sie dieses Kapitel auch überspringen und dann bei Bedarf auf die entsprechenden Rezepte zurückgreifen.

Kapitel 3: Die Arbeit an und mit Nagios/Icinga

Dieses Kapitel stellt Ihnen in einzelnen Rezepten die Arbeit mit Nagios und Icinga vor. Sie erhalten hier Hilfe über recht unterschiedliche Themen von der Einrichtung von Benutzern über die Nutzung der Oberflächen bis zur Sicherung der Daten. Manche Rezepte sind dabei nur für fortgeschrittene Benutzer wirklich wichtig, während andere eher auf den Einsteiger abzielen. Hier lohnt es sich für Sie, die Rezepte im Inhaltsverzeichnis zur Kenntnis zu nehmen und gegebenenfalls die Ihnen interessant erscheinenden zu lesen. Ansonsten gilt auch hier, dass wir an den relevanten Stellen auf die Rezepte verweisen, so dass Sie sie im Zweifel bei Bedarf nachlesen können.

Kapitel 4: Die Konfigurationsobjekte von Nagios/Icinga

Was Ihr Nagios-/Icinga-Monitoringsystem wie überwacht, teilen Sie dem System über Konfigurationsobjekte mit. Diese erklären Ihnen die Rezepte in diesem Kapitel. Als Einsteiger sollten Sie sich von der Anzahl der möglichen Objekte nicht abschrecken lassen, denn nicht alle davon müssen von Ihnen genutzt werden. Die Einleitung des Kapitels erklärt Ihnen, welche Objekte Sie sich zunächst anschauen müssen, um überhaupt mit dem System arbeiten zu können. Die meisten Objekte können Sie dann aber zunächst außen vor lassen und sich erst zu Gemüte führen, wenn Sie auf die entsprechende weitergehende Verwendung gestoßen sind. Wie immer werden wir Sie auf die Rezepte zu den jeweiligen Objekttypen hinweisen, wenn andere Rezepte auf diesen aufbauen.

Kapitel 5: Lokale Parameter mit Standard-Plugins überwachen

Die im Rahmen der Installation mit-installierten Standard-Plugins (genannt Nagiosplugins) bieten Ihnen ein breites Spektrum von Prüfmöglichkeiten für die bei Ihnen vorhandenen Geräte. In diesem Kapitel finden Sie zunächst Rezepte zu dem Teil der Plugins, die nur lokal auf einer Maschine ausgeführt werden können. Dabei haben wir Ihnen jeweils die Funktionsweise, Anpassungsmöglichkeiten und die Einbindung in das Monitoring-System beschrieben. Die entsprechenden Prüfungen sind in der Regel aber nicht nur für die Monitoring-Maschine selbst interessant. Deshalb finden Sie hier auch Rezepte für Plugins, die es Ihnen erlauben solche lokalen Tests auf entfernten Maschinen durchzuführen. Sie sollten hier einmal querlesen und die für Sie besonders interessante Prüfungen betrachten. In Kapitel 8 erklären wir Ihnen, was Sie auf bestimmten Servertypen und Geräten wie überwachen sollten. Dabei verweisen wir Sie dann auf die jeweils dahinter stehenden Plugins, so dass Sie im Zweifel später auf diesem Wege zu den entsprechenden Rezepten gelangen.

Kapitel 6: Netzwerk-Dienste mit Standard-Plugins überwachen

Neben den in Kapitel 5 behandelten lokal ausführbaren Plugins enthalten die mitinstallierten Plugins eine Vielzahl von Abfragen und Funktionsprüfungen entfernter Dienste. Für die meisten dieser Plugins steht Ihnen in diesem Kapitel ein entsprechendes Rezept zur Verfügung. Auch hier finden Sie dann jeweils eine Funktionsbeschreibung, die Erläuterung der wichtigsten Anpassungsmöglichkeiten sowie die Beschreibung der Einbindung in das Monitoring-System. Wenn Sie sich für bestimmte Prüfungen besonders interessieren, können Sie gezielt auf diese Rezepte zurückgreifen. Ansonsten beschreiben wir Ihnen in Kapitel 8, was Sie bei bestimmten Server- und Gerätetypen überwachen sollten und verweisen Sie dabei dann auf die jeweiligen Rezepte zu den dahinter liegenden Plugins. Sie können also alternativ auch in Kapitel 8 ansetzen und dann den Verweisen dort folgen.

Kapitel 7: Erstellung eigener und Einbindung externer Plugins

Auch wenn die in den vorhergehenden Kapiteln beschriebenen mitinstallierten Plugins Ihnen bereits eine große Zahl unterschiedlicher Prüfungen ermöglichen, so gibt es typischerweise dennoch Anforderungen, die von diesen nicht oder nicht ausreichend abgedeckt werden. Für diese Fälle bietet Ihnen Nagios/Icinga eine Programmierschnittstelle, über die Sie eigene Programme als Plugins einbinden können. In diesem Kapitel finden Sie Rezepte zur Einführung in diese Schnittstelle und Beispiele dafür, wie Sie diese nutzen können. Darüber hinaus stellen Ihnen andere Nutzer von Nagios/Icinga über das Internet ihre Erweiterungen zur Verfügung. In eigenen Rezepten zeigen wir Ihnen, wie Sie solche externen Plugins einbinden können und was Sie dabei beachten müssen.

Kapitel 8: Überwachung von Maschinen, Diensten und Infrastruktur

In diesem Kapitel zeigen wir Ihnen, wie Sie bestimmte Geräte-Typen in das Monitoring-System einbinden können. Dabei fließen die Informationen aus den vorangegangenen Kapiteln zusammen. Hier erfahren Sie im Überblick, wie Sie beispielsweise Windows-Server generell einbinden und welche Prüfungen Sie bei einem E-Mail-Server durchfüh-

ren sollten. Aber auch die Einbindung der aktiven Netzwerk-Infrastruktur, also die von Switches, Routern und unterbrechungsfreien Stromversorgungen, stellen wir Ihnen vor. Und keine Sorge, Sie müssen sich nicht vorher durch alle anderen Kapitel durcharbeiten, da wir jeweils auf die relevanten Rezepte verweisen.

Kapitel 9: Datenvisualisierung mit PNP4Nagios und NagVis

Nachdem wir in den bisherigen Kapitel die einzelnen Aspekte des Auf- und Ausbaus des Monitorings selbst behandelt haben, stehen Ihnen in diesem abschließenden Kapitel Rezepte für die Auswertung der damit zur Verfügung stehenden Daten zur Verfügung. Wir behandeln dabei zunächst PNP4Nagios als Lösung für die Erstellung von Graphen. Sie erhalten hiermit insbesondere Zugang zur zeitlichen Entwicklung von Daten, die Ihnen als Grundlage für Analysen und Prognosen dienen. Als alternative Form der Darstellung der aktuellen Zustände steht Ihnen NagVis zur Verfügung. In weiteren Rezepten erklären wir Ihnen, wie Sie sich eigene Darstellungen auf der Basis von Karten, Grundrissen oder Fotos erstellen. Diese sind besonders gut für die ständige Darstellung in Ihrem Überwachungszentrum oder auf öffentlichen Anzeigen geeignet.

Die Webseite zum Buch

Unter *http://www.monilog.info* stellen wir Ihnen zusätzliche Informationen zur Verfügung. Als Nutzer des Buches sollten Sie hier vorbeischauen. Zum einen möchten wir Ihnen ersparen, die längeren Quellcode-Teile abzutippen. Und wenn es Errata gibt oder wir Aktualisierungen vornehmen, werden wir diese ebenfalls hier veröffentlichen.

Verwendete Konventionen

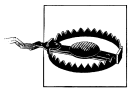
In diesem Buch werden die folgenden typographischen Konventionen verwendet:

Kursivschrift
: für Datei- und Verzeichnisnamen, Menüs, Code im Text, E-Mail-Adressen und URLs, aber auch bei der Definition neuer Fachbegriffe und für Hervorhebungen

`Nichtproportionalschrift`
: für die Codebeispiele

`Nichtproportionalschrift fett`
: zur Hervorhebung einzelner Zeilen oder Abschnitte in den Codebeispielen

Dieses Symbol kennzeichnet einen Tipp, einen Vorschlag oder eine allgemeine Anmerkung.



Dieses Symbol kennzeichnet eine Warnung.

KAPITEL 1

Nagios/Icinga installieren und Hostsystem vorbereiten

In diesem Kapitel bieten wir Ihnen zunächst eine Übersicht der Installation von Nagios/Icinga. Wir unterstützen Sie bei der Auswahl aus den beiden Varianten und den verfügbaren Versionen (siehe Rezept 1.1, *Entscheidungshilfe für die Installation*). Anschließend geben wir Ihnen Hinweise zur Planung des Aufbaus (siehe Rezept 1.3, *Hardware-Anforderungen abschätzen*) und der Anbindung des Monitoring-Servers (siehe Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*).

Die Installation selbst zeigen wir Ihnen dann für eine Auswahl häufig im Serverumfeld genutzter Linux-Distributionen. Besonders ausführlich werden wir die Installation aus den Quellen behandeln, die wir Ihnen am Beispiel der Distribution Debian zeigen werden (siehe Rezept 1.9, *Nagios aus den Quellen installieren* und Rezept 1.4, *Icinga aus den Quellen installieren*). Dabei werden wir uns mit einer Auswahl von Erweiterungen beschäftigen, und zwar zunächst mit PNP4Nagios als Lösung zur Speicherung von Performancedaten und Auswertung in Graphen. In Abbildung 1-1 ist ein entsprechender Graph dargestellt.

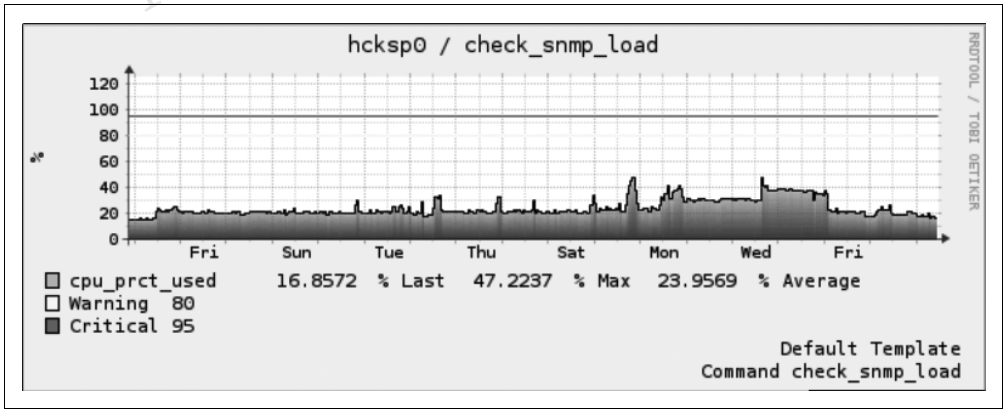


Abbildung 1-1: Beispiel-Graph der Prozessorlast mit PNP4Nagios

Des Weiteren werden wir Ihnen die Installation und Einrichtung der NDOUtils/IDOUtils für die Anbindung an eine Datenbank und das darauf aufbauende NagVis zur Anzeige von Status aus Nagios/Icinga in quasi beliebiger Anordnung und auf Hintergründen zeigen. Abbildung 1-2 ist ein Beispiel für eine entsprechende Visualisierung.

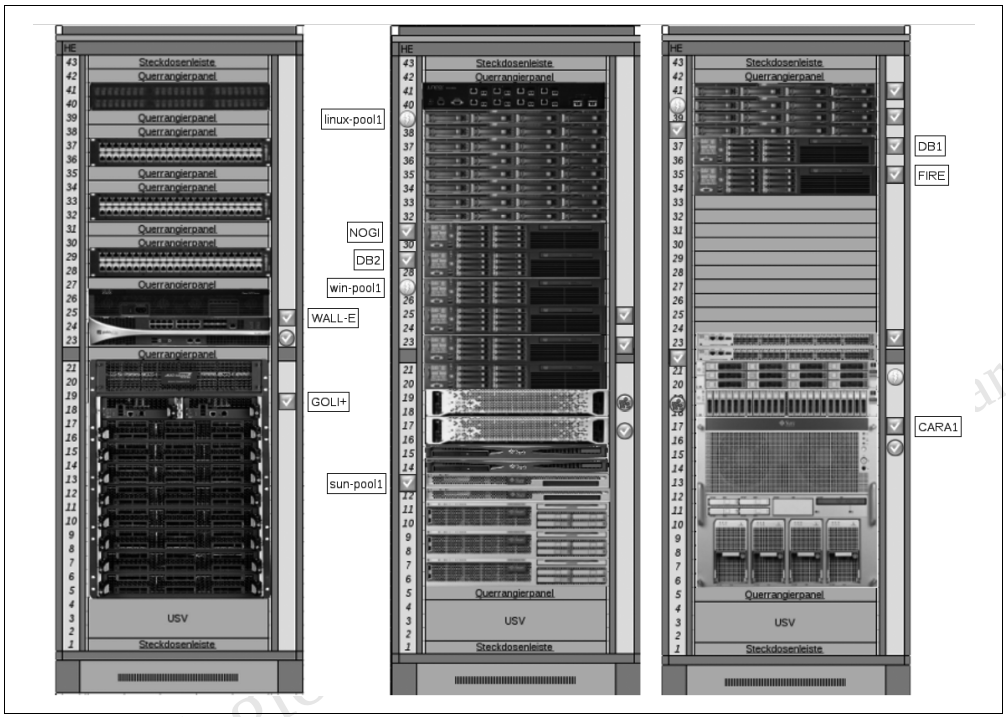


Abbildung 1-2: Beispiel für Visualisierung eines Zentralverteilers mit NagVis

Außerdem werden wir alternativ zur Installation aus entsprechenden Quellen auch die Installation über Pakete erläutern. Dabei haben wir uns für Sie mit den Distributionen Debian|Ubuntu, SLES|openSUSE und RHEL|CentOS|Fedora in den jeweils aktuellen Versionen befasst.

In Kapitel 2, *Basiskonfiguration der Systeme*, werden wir Sie durch grundlegende Vorbereitungen der Monitoring-Maschine selbst und der zu überwachenden Geräte führen. Das grundsätzliche Arbeiten mit Nagios/Icinga werden wir dann in Kapitel 3, *Die Arbeit an und mit Nagios/Icinga*, erläutern, das gefolgt wird von Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, in dem wir auf die einzelnen Konfigurationsobjekte eingehen. Dabei wird es dann letztendlich um den Einsatz von Plugins gehen, mit denen jeweils bestimmte Prüfungen durchgeführt werden. Wichtige Standard-Plugins für lokale Prüfungen werden wir in Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*, erläutern und mit solchen für Prüfungen über das Netzwerk werden wir uns dann anschließend in Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*, befassen. Anleitungen für die Erstellung eigener und Einbindung weiterer Plugins aus externen Quellen werden Sie in Kapitel 7,

Erstellung eigener und Einbindung externer Plugins, erhalten. In Kapitel 8, *Überwachung von Maschinen, Diensten und Infrastruktur*, werden wir Ihnen dann das Monitoring spezifischer Geräteklassen vorstellen und Ihnen konkrete Vorschläge zum Einsatz von Plugins und deren Optionen machen. Hier werden dann also die in den vorhergehenden Kapiteln beschriebenen Schritte zusammengeführt. Abschließend werden wir in Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*, auf die Datenvisualisierung mit Hilfe von PNP4Nagios und NagVis eingehen.

1.1 Entscheidungshilfe für die Installation

Problem

Sie möchten ein Monitoring-System auf Basis von Nagios aufsetzen. Dabei bieten sich Ihnen zunächst zwei Alternativen, nämlich das ursprüngliche Nagios und das daraus entstandene Icinga. Anschließend müssen Sie eine Entscheidung bezüglich der zu installierenden Version treffen. Hier haben Sie grundsätzlich die Wahl zwischen bereits im Paketmanagement-System der jeweiligen Distribution enthaltenen älteren, aber dafür entsprechend angepassten Versionen und der jeweils letzten als Quellcode veröffentlichten stabilen Version.

Lösung

Die Unterschiede, die sich derzeit aus den verschiedenen bei den Linux-Distributionen enthaltenen Versionen von Nagios und Icinga ergeben, sind bezüglich der Kernaufgabe des Monitorings eher marginal. Wenn Sie aber beispielsweise auch Graphen zeichnen möchten, bieten die unterschiedlichen Distributionen doch unterschiedliche Möglichkeiten im Hinblick auf die verfügbaren Pakete.

Der Vorteil von aktuellen Versionen aus Quelltexten ist generell der, dass sie Fehlerbehebungen, Erweiterungen und neue Funktionen enthalten. Demgegenüber sind bei entsprechend älteren Versionen aus einem Paketmanagement die distributionsspezifischen Anpassungen und die im Rahmen der Aufnahme durchgeführte Überprüfung zu nennen. Da die über Paketmanagement-Systeme verfügbaren Versionen in jeder Distribution anders sind, bereitet es hier jedoch zunächst einige Schwierigkeiten, einen entsprechenden Überblick zu gewinnen.

Icinga vs. Nagios

Icinga ist im 2. Quartal 2009 aus einer Kopie des Quellcodes von Nagios, also als sogenannter Fork, entstanden. Hintergrund war der zögerliche Fortschritt des Nagios-Projekts und die Abhängigkeit bei Entscheidungen von einem Einzelnen sowie Auseinandersetzungen um eingetragene Warenzeichen (siehe hierzu *https://www.icinga.org/faq/why-a-fork/*). Zum Zeitpunkt der Entstehung dieses Buches ist Icinga (noch, siehe hierzu auch Rezept 9.9, *Ausblick auf das kommende Icinga2*) vollständig kompatibel mit Nagios

im Hinblick auf zum Beispiel die Programmierschnittstelle und damit die Verwendbarkeit von Plug-ins und APIs. Die von uns betrachteten als stabil bewerteten Releases sind Nagios ab Version 3.5.0 und Icinga ab Version 1.9.1. Bei den paketbasierenden Installation haben wir auf die jeweils aktuell verfügbaren Pakete zurückgegriffen.

Beide Varianten, also Nagios wie auch Icinga, werden weiterentwickelt. Die Veröffentlichungszyklen des neueren Projektes sind erwartungsgemäß kürzer, als die des etablierten.

Kommerzielle Version: Von Nagios gibt es neben der frei verfügbaren Open-Source-Version `Nagios Core` seit dem 4. Quartal 2009 eine kommerzielle Version `Nagios XI` des Anbieters Nagios Enterprises LLC. `Nagios Core` (das betrifft nur den Kernteil der Anwendung) wird als sogenannte Open Core-Lösung weiterentwickelt, auf Basis dessen verschiedene kommerzielle Lösungen existieren (`Nagios XI, Opsview, op5, Groundwork`, etc). Betrachtungen zur kommerziellen Lösung `Nagios XI` sind nicht Bestandteil dieses Buches, auch wenn sich vieles übertragen lässt. Icinga folgt im Gegensatz dazu nicht dem Open Core-Modell, sondern umfasst seinerseits zusätzliche Komponenten (`Icinga Web, Icinga Reporting`, etc), die frei verfügbar sind.

Versionswahl

Prinzipiell bietet sich die Nutzung der als stabil bewerteten Versionen an. Nur diese halten üblicherweise auch Einzug in die Paketmanagement-Systeme. Ansonsten ist es in den meisten Einsatzszenarien nicht wirklich relevant, ob die eingesetzte Version etwas älter ist. Und im Zweifel ist es immer möglich, auf eine Installation aus den Quellen umzusteigen (zur Migration siehe Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*). Aus diesem Grund spricht nichts prinzipiell dagegen, die in Ihrer Umgebung im Paketmanagementsystem verfügbare Version einzusetzen.

Sie sollten allerdings wissen, dass sich die bei der Installation verwendeten Verzeichnisse bei der Installation aus Quellcode gegenüber denen bei der Installation aus Paketen unterscheiden. Sie ersparen Sie sich also Arbeit, wenn Sie Nagios/Icinga und alle Erweiterungen (die sogenannten Addons) einheitlich entweder aus dem Quellcode oder über Pakete installieren.

Auswahl an Erweiterungen (Addons)

Die verfügbaren Erweiterungen können die Arbeitsweise des Monitorings erheblich erweitern beziehungsweise die Darstellung und Auswertung der Ergebnisse verändern. Nicht selten sind Anforderungen von neu erschienenen Erweiterungen auch der Grund für durchgeführte Aktualisierungen. Zum Verständnis der folgenden Ausführungen hier einige Erweiterungen mit einer kurzen Beschreibung Ihrer Funktionalität:

- PNP4Nagios: Erzeugt Graphen aus Monitoring-Werten wie beispielsweise CPU-Auslastung oder Bandbreitennutzung (siehe *http://www.pnp4nagios.org/*).
- NagVis: Visualisierungs-Werkzeug, das die Anordnung von graphischen Status-Indikatoren auf Hintergründen wie etwa Karten oder Fotos ermöglicht (siehe *http://www.nagvis.org/*).

- NRPE (Nagios Remote Probe Executor): Ein Dienst, der auf Anforderung eines externen Nagios/Icinga Plugins ausführt und die Ergebnisse zurückliefert (siehe *http://sourceforge.net/projects/nagios/files/*).
- NSCA (Nagios Service Check Acceptor): Komponente, die Ergebnisse von auf anderen Maschinen ausgeführten Plugins entgegennimmt (siehe *http://sourceforge.net/projects/ nagios/files/*).

Tabelle 1-1 bietet eine Übersicht darüber, welche dieser Erweiterungen über die Paketverwaltung in den verschiedenen derzeit aktuellen Versionen der häufig genutzten Distributionen verfügbar sind.

Tabelle 1-1: Distributionen und enthaltene Pakete

Distribution	Als Pakete verfügbare Addons
Seit Debian 7.0	PNP4Nagios, NagVis, NSCA, NRPE
Seit Ubuntu 12.04 LTS	PNP4Nagios, NagVis, NSCA, NRPE
Seit openSUSE 12.2 + server:monitoring + server:php:applications Repositories	PNP4Nagios, NagVis, NSCA, NRPE
Seit CentOS 6.3 + EPEL Repository	PNP4Nagios, NSCA, NRPE
Seit CentOS 6.3 + RepoForge Repository	NSCA, NRPE

Alle Erweiterungen sind prinzipiell auf allen Plattform über die Installation aus Quellen verfügbar, weshalb diese Übersicht nicht die Entscheidungsgrundlage für ein Produktivsystem sein sollte. Aber insbesondere für den Aufbau einer Test-Installation kann die Auswahl einer diesbezüglich passenden Distribution Ihnen einigen Aufwand ersparen.

Diskussion

Wir haben selbst lange Jahre mit Nagios gearbeitet, sind mittlerweile allerdings auf Icinga umgestiegen. Aus unserer Erfahrung geschehen hier deutlich schneller Reaktionen auf Fehler- und Rückmeldungen (teilweise in Stunden), während von uns geschriebene Fehlerberichte bei Nagios auch nach Jahren nicht bearbeitet worden sind. Durch diesen Hintergrund sind wir Icinga-Nutzer geworden. Für eine produktive Umgebung lassen sich nach derzeitigem Stand durchaus beide Projekte verwenden und es gibt auch für beide Support-Verträge von Firmen. Schon jetzt bietet Icinga allerdings Funktionalität, die es für Nagios nicht oder eben nur gegen Bezahlung gibt – insbesondere Unterstützung weiterer Datenbanken wie Postgresql und Oracle und eine neue Oberfläche. Einen Ausblick auf die anstehende Entwicklung in Icinga2 geben wir Ihnen in Rezept 9.9, *Ausblick auf das kommende Icinga2*. Wir beschreiben in diesem Buch beide Varianten, aber wenn Sie eine Empfehlung von uns möchten so ist dies Icinga.

Gegenüber der Installation aus den Quellcodes bietet die Installation aus Paketen einige Vorteile. Die von der jeweiligen Distribution ausgewählte Nagios/Icinga-Version ist meist

von hoher Stabilität und an die Distribution angepasst. Bei der Installation und Konfiguration werden große Teile automatisch abgearbeitet, wodurch sich die Anzahl der von Ihnen selbst durchzuführenden Arbeitsschritte erheblich reduziert. Ein weiterer Vorteil besteht in der Möglichkeit einer automatischen Aktualisierung. Dies ist besonders dann hilfreich, wenn Sie beispielsweise auf ein automatisches Einspielen von Sicherheitspatches Wert legen.

Bei der Installation aus Paketen sind Sie dann allerdings auch auf die in den Paketquellen verfügbare Version beschränkt, die allerdings gegenüber den Veröffentlichungen der Quellcodes meist nicht unerheblich veraltet ist. Bei einer Installation aus Quellcode können Sie sich hingegen normalerweise für jede beliebige der veröffentlichten Versionen entscheiden.

Je nachdem, ob Sie ein System für den Produktivbetrieb oder für Testzwecke aufsetzen möchten, sollten Sie sich auch auf eine Ihren Bedürfnissen entsprechende Umgebung festlegen. Falls Sie eine Empfehlung für eine Distribution suchen, würden wir folgende Empfehlungen aussprechen:

- für den Produktionsbetrieb die Installation auf Debian Stable (Rezept 1.5, *Icinga unter Debian installieren*)
- für Testzwecke die Installation der jeweils aktuellsten Ubuntu-Servervariante (Rezept 1.6, *Icinga unter Ubuntu installieren*) – aber nicht unbedingt in der hier gezeigten LTS-Version

Sofern Sie die Installation aus Quellcode allerdings nicht scheuen (Rezept 1.9, *Nagios aus den Quellen installieren* und Rezept 1.4, *Icinga aus den Quellen installieren*), erhalten Sie damit die größtmögliche Flexibilität im Hinblick auf Veränderungen an dem System und die Integration mit Erweiterungen, so dass dies insbesondere für Produktionssysteme durchaus eine Alternative darstellt.

Allerdings kann auch eine Installation aus den Quellen problematisch werden. Wir sind vor ein paar Jahren zumindest bei der Installation von Nagios unter Solaris an einen Punkt gelangt, an dem es ohne Anpassungen im Quellcode schlicht nicht weiter ging. Für den Fall, dass Sie also ein etwas exotischeres System verwenden möchten, empfehlen wir Ihnen sicherheitshalber, entsprechend Zeit für eine Testinstallation einzuplanen.

Siehe auch

- Webseite des Nagios-Projekts: *http://nagios.org/*
- Webseite des Icinga-Projekts: *https://www.icinga.org/*
- Erklärung des Icinga-Projekts zur Notwendigkeit der Abspaltung: *https://www.icinga.org/faq/why-a-fork/*
- Rezept mit Ausblick auf das kommende Icinga2: Rezept 9.9, *Ausblick auf das kommende Icinga2*
- Webseite des Debian Projekts: *http://www.debian.org/*
- Webseite von Ubuntu: *http://www.ubuntu.com/*

- Webseite von openSUSE: *http://de.opensuse.org/*
- Webseite von CentOS: *http://www.centos.org/*
- Rezepte zur Installation von Nagios und Icinga aus den Quellen: Rezept 1.9, *Nagios aus den Quellen installieren* und Rezept 1.4, *Icinga aus den Quellen installieren*
- Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*
- Webseite des PNP4Nagios Projekts: *http://www.pnp4nagios.org/*
- Webseite des NagVis Projekts: *http://www.nagvis.org/*
- Sourceforge-Seite mit den Quellen von NRPE: *http://sourceforge.net/projects/nagios/files/ nrpe-2.x/*

1.2 Netzwerkanbindung des Monitoring-Servers planen

Problem

Sie planen den Aufbau eines Monitoring-Systems. Der Monitoring-Server soll dabei Zugriff auf alle zu überwachenden Maschinen haben, und das möglichst auch dann, wenn das eigentliche Netzwerk gestört ist.

Lösung

Als Lösung können Sie für die Netzwerkanbindung getrennte Netzwerke, also neben dem Kern-Netzwerk ein zusätzliches, sogenanntes Management-Netzwerk verwenden. Abbildung 1-3 zeigt Ihnen den herkömmlichen Aufbau eines einfachen Sternnetzwerks, ergänzt durch ein Management-Netzwerk mit einem zentralen Management-Switch.

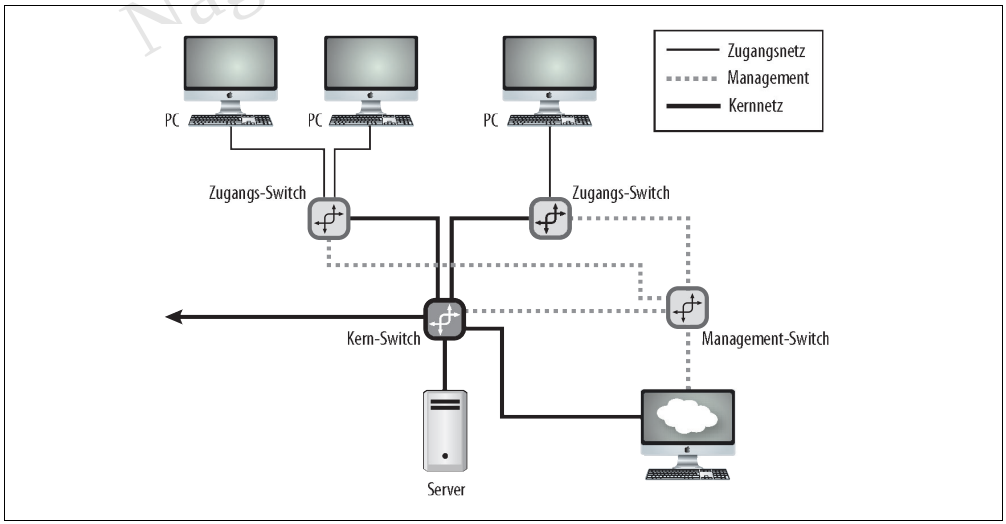


Abbildung 1-3: Duales Kern- und Management-Netzwerk

Die zu favorisierende Lösung besteht in der Realisierung des Management-Netzwerkes als separates physikalisches Netzwerk (ein sogenanntes Outband-Management-Netzwerk). Dieses wird über eine parallel zum Produktiv-Netzwerk aufgebaute physische Infrastruktur betrieben. Dafür benötigen Sie separate Switches und passive Infrastruktur (meist Kupfer- oder Glasfaserverkabelung) beziehungsweise Leitungen (zum Beispiel über Modemeinwahl, ISDN oder VPN über DSL). Aufgrund der viel geringeren Leistungsanforderungen ist die Management-Infrastrukter sehr viel kostengünstiger als die des produktiv genutzten Kern-Netzwerks. Innerhalb von Serverräumen oder -gebäuden lässt sich in vielen Fällen mit wenig Aufwand eine entsprechende Nachrüstung durchführen. Wenn allerdings zwischen Gebäuden oder gar über größere Distanzen keine zweite Infrastruktur mit ausreichend Leitungskapazitäten zur Verfügung steht, dann kann der Aufwand für diese Lösung schnell zu groß werden. Sie können dann auf eine Lösung über VLANs (ein sogenanntes Inband-Management-Netzwerk) zurückgreifen. Dafür richten Sie ein separates VLAN ein, das jedoch aus dem Kern-Netzwerk nicht geroutet wird, also aus diesem nicht erreicht werden kann.

Wenn Sie über ein solches Management-Netzwerk verfügen, können Sie wichtige Maschinen über eine zweite Netzwerkschnittstelle (physisch oder virtuell) mit diesem verbinden. Auf diese Weise lassen sich Prüfungen durchführen, selbst wenn die primäre Netzwerkinfrastruktur fehlerhaft oder gar ausgefallen ist. Dies kann bei der Fehlerdiagnose sehr hilfreich sein. Wenn nämlich ein solcher Server über das Management-Netzwerk normal reagiert, aber über das normale Netzwerk nicht oder nur eingeschränkt zu erreichen ist, dann können Sie auf ein Netzwerkproblem (im Gegensatz zu einem grundsätzlichen Serverproblem) schließen.

Das Management-Netzwerk können Sie nicht nur für das Monitoring, sondern eben auch bei Zugriffen für Wartungsarbeiten und Ähnliches verwenden. Dadurch werden Messungen wie etwa der Netzwerkauslastung (Bandbreite, Anzahl Verbindungen u.ä.) dann auch weniger durch das Monitoring und Management selbst verfälscht.

Bei der Anbindung das Monitoring-Systems müssen Sie allerdings beachten, dass Sie die Funktionsfähigkeit der Netzwerk-Dienste natürlich aus den Netzen testen müssen, aus denen diese erreichbar sein sollen. Denn ohne das Testen der tatsächlich genutzten Netzwerk-Infrastruktur ließe sich natürlich auch ein entsprechender Ausfall nicht in allen Fällen feststellen. Da viele der üblichen Prüfungen auf interne Abfragen wie etwa nach CPU-, Speicher- und Netzwerk-Auslastung hinauslaufen, macht der in einem Management-Netzwerk auslagerbare Verkehr allerdings meist den größten Teil des insgesamt durch das Monitoring verursachten Netzwerkverkehrs aus.

Hinzu kommt für Sie möglicherweise ein Sicherheitsgewinn, wenn Sie Geräte mit Hilfe von SNMP (Simple Network Management Protocol) überwachen, diese aber SNMP nur in den älteren, unverschlüsselten Versionen 1 und 2 beherrschen. In einem isolierten Management-Netzwerk sind entsprechende Abfragen weniger problematisch. Demgegenüber ist unverschlüsselter Management-Verkehr mit Klartext-Passwörtern in einem Produktivnetz zumindest sehr bedenklich. Dasselbe gilt selbstverständlich auch für die Nutzung von Protokollen wie `telnet` und `rsh` anstelle von `ssh` für das Management.

Wenn Sie eine sehr große, verteilte Nagios/Icinga-Installation (hunderte von Maschinen) planen, bei der die Last des Monitoringsystems auf unterschiedliche Maschinen (mehrere verteilte Monitoring-Server, Datenbankserver und Webserver) verteilt werden soll, dann sollten Sie diese mit einer dedizierten Infrastruktur miteinander zu verbinden (sozusagen ein Monitoring-Backbone). Damit können schnelle Zugriffe, etwa zwischen Datenbank und Monitoring, garantiert werden, ohne dass andere Teile des Netzwerkes dadurch belastet würden.

Diskussion

Ein Outband-Management-Netzwerk bietet nicht nur für das Monitoring erhebliche Vorteile, sondern eben auch für das Management. Gleichwohl ist die eigene Infrastruktur mit Kosten verbunden, auch wenn für ein solches Low-Traffic-Netz deutlich günstigere Komponenten verwendet werden können als im Produktivnetz. Auch mit einem solchen Management-Netzwerk benötigt eine Monitoring-Maschine Zugang zum eigentlichen Netzwerk, da nur hier tatsächlich die Funktionsfähigkeit von Diensten getestet werden kann.

Wenn eine physisch getrennte Netzwerk-Infrastruktur nicht möglich ist, können Sie überlegen, stattdessen ein logisch getrenntes Netzwerk über ein VLAN zu realisieren. Den entscheidenden Vorteil, dass Sie die überwachten Maschinen auch bei einem Ausfall der Infrastruktur noch erreichen zu können, bietet die Inband-Lösung nicht. Aber durch die Trennung des Netzwerkverkehrs erhöhen Sie zumindest die Sicherheit für nicht verschlüsselte Verbindungen, wie dies bei SNMP Version 1 und 2, `telnet` und `rsh` der Fall ist.

Routing: Bei einem Management-Netzwerk sollten Sie unabhängig davon, ob Sie sich für ein Inband- oder ein Outband-Netzwerk entscheiden, immer darauf achten, dass Sie den Verkehr nicht in das Produktivnetz routen.

1.3 Hardware-Anforderungen abschätzen

Problem

Sie möchten ein Monitoring-System aufbauen und müssen dazu die Hardware-Anforderungen abschätzen.

Lösung

Das reine Monitoring mit Nagios oder Icinga an sich wird in der Regel auch bei größeren Netzwerken noch auf jedem aktuellen Standard-PC laufen. Die Abschätzung der Anforderungen bezüglich der Leistung hängt jedoch weniger von der Anzahl der zu überwachenden Geräte, sondern eher von Anzahl der zu überwachten Dienste, von der Art des Checks und insbesondere von der Häufigkeit der Tests ab.

Grundsätzlich benötigt Nagios/Icinga also erstmal nicht viele Ressourcen. Erst bei Tausenden Service-Checks pro Minute entsteht eine signifikante Systemlast. Werden der

Abruf und damit die Berechnung der entsprechenden Graphen in Weboberflächen auch auf derselben Maschine durchgeführt und intensiv von mehreren Mitarbeitern genutzt, dann werden schnell mehrere CPUs, beziehungsweise Kerne, notwendig. Wenn Sie die Daten aber auch auf demselben Rechner in einer Datenbank speichern, dann steigen die Schreibraten auf den Datenträger schnell an, so dass unter Umständen mehrere Festplatten und ein RAID0 oder RAID10-Array nötig werden.

Eine genaue Berechnung der zu erwartenden Belastung ist nach unserer Erfahrung nicht sinnvoll möglich, da es einfach zu viele und nur schwer einschätzbare Faktoren zu beachten gilt. Aber gehen Sie vorsichtshalber von mehr durchgeführten Prüfungen aus, als Sie aktuell konkret planen. Überlegen Sie außerdem, wie häufig die Tests durchgeführt werden sollen. Diese Überlegung ist sehr entscheidend für die entstehende Last, aber eben auch für die Auflösung von Graphen. Unserer Erfahrung nach besteht hierin der häufigste Grund für kleine Prüfintervalle von unter 10 Minuten.

Ausgehend von einem durchschnittlichen Prüfintervall von 5 Minuten, können Sie sich an den Werten in Tabelle 1-2 orientieren. Damit sollten Sie für die meisten Fälle ausreichend Reserven haben:

Tabelle 1-2: Nagios/Icinga Hardware-Anforderungen

Dienste-checks	CPU Cores (min. 2Ghz)	Arbeits-speicher	Festplatten-speicher	Netzwerk
< 100	1	4 GB	40 GB	100Mbit
100 – 1000	4	8 GB	100 GB	100Mbit
> 1000	16	16 GB	400 GB	100Mbit

Diskussion

Außer bei sehr kleinen Prüfintervallen von weniger als 5 Minuten oder einer sehr umfangreichen Umgebung mit mehr als 5000 Dienste-Checks sollte ein Monitoring-Server ausreichend sein. Falls Sie mit dem gewählten Rechner dann an Lastgrenzen stoßen sollten, bleiben Ihnen immer noch einige Optionen zur Optimierung. Hierzu geben wir Ihnen folgende Hinweise:

- Zunächst sollte Sie Ihr Monitoringsystem auch selber überwachen. Achten Sie auf Werte wie CPU-Last, Datenträger-I/O und Speicherverbrauch (Arbeits- und Festplattenspeicher).
- Beobachten Sie die durchschnittlichen Latenzzeiten bei der Überprüfung von Hosts und Services. Durchgehend hohe Latenzen können auch auf eine Überlastung hinweisen. Nagios/Icinga bieten hierfür das Tool `nagiosstats` beziehungsweise `icinga-stats`.
- Erwägen Sie, die Prüfintervalle Ihres Monitoring-System soweit wie möglich hochzusetzen. Die Auswirkung der Intervalle ist insbesondere bei vielen Prüfungen erheblich.

- Dann sollten Sie die von Ihnen genutzten Plugins betrachten.
  - Wenn Sie `check_ping` (siehe Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*) häufig einsetzen, erwägen Sie einen Umstieg auf `check_icmp` (siehe Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)*).
  - Wenn Sie viele interpretierte Plugins wie Shell-Scripte oder Perl-Programme ausführen, erwägen Sie, soweit möglich, den Umstieg auf kompilierte Plugins (meist C/C++). Auf diese Weise können Sie unter Umständen große Performance-Einsparungen erzielen.
  - Die Nutzung von `check_by_ssh` (siehe Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*) erfordert wesentlich mehr Resourcen als die von `check_snmp` (siehe Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*) und `check_nrpe` (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*). Wir empfehlen Ihnen, sofern möglich, SNMP in Version 3 einzusetzten (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* für Linux und Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten* für Windows).
- Wenn Sie PNP4Nagios und NagVis intensiv Nutzen, dann prüfen Sie bei Performanceproblemen die Auslastung der Festplatten. Sie können dazu das Tool `iostat` nutzen. Bei umfangreichen Installationen müssen Sie eventuell die Schreibperformance auf den Datenträger verbessern. Erwägen Sie in diesem Fall die Anschaffung eines Raid-Controllers und die Konfiguration eines RAID0 oder RAID10-Arrays. Alternativ können Sie den Festspeicher entlasten, indem Sie die temporären Performancedaten und Prüfungsergebnisse in eine RAMDisk auslagern.
- Des Weiteren lassen sich auch ganze Teile Ihres Monitoring-Systems auf andere Maschinen auslagern. Sie können die Last verteilen, indem Sie Monitoring, Datenbank und Benutzerschnittstellen jeweils auf eigenen Maschinen laufen lassen. Neben den entsprechenden Anpassungen in der Konfiguration ist dann allerdings auch eine ausreichende Netzwerkanbindung zwischen diesen Maschinen erforderlich.
- Es gibt noch diverse Möglichkeiten zur Optimierung der Konfiguration, wozu wir Sie auf die entsprechenden Seiten der Projekte verweisen möchten: Für Nagios *http://nagios.sourceforge.net/docs/3_0/tuning.html* und *http://docs.icinga.org/latest/de/tuning.html* für Icinga.
- Wenn Ihr Rechner trotz Optimierung weiterhin einer Überlastung ausgesetzt ist, können Sie die Tests auf andere, dezentral installierte Nagios/Icinga-Instanzen auslagern. Die Resultate werden häufig mit NSCA (Nagios Service Check Acceptor) an den zentralen Monitoring-Server übermittelt. Dieser führt dann nur noch die Ergebnisse zusammen.

Auf die Optimierung von umfangreichen Installationen gehen wir in diesem Buch nicht weiter ein. Bitte konsultieren Sie dazu die Nagios/Icinga-Dokumentation.

Siehe auch

- Rezepte zur Überwachung der Erreichbarkeit mit `ping` und `icmp`: Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)* und Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)*
- Rezept zur Remote-Ausführung von Plugins über SSH: Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*
- Rezept zur Erstellung generischer Abfragen mit SNMP: Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*
- Webseiten zur Optimierung eines Nagios/Icinga-Systems für das Monitoring: *http://nagios.sourceforge.net/docs/3_0/tuning.html* beziehungsweise *http://docs.icinga.org/latest/de/tuning.html*

1.4 Icinga aus den Quellen installieren

Problem

Sie möchten ein aktuelles Icinga aus den Quellen installieren.

Lösung

Im Folgenden zeigen wir zunächst die quellenbasierte Installation von Icinga ab Version 1.9 mit dem Datenbank-Addon IDOUtils (Icinga Data Out Utilities) auf Basis der Distribution Debian 7.0 (Codename Wheezy). Darauf aufbauend erläutern wir dann, wie Sie die Addons PNP4Nagios (Graphen) und NagVis (Karten) installieren und in Icinga integrieren. Ausgangspunkt ist dabei ein frisch installiertes System in Minimalkonfiguration, jedoch mit bereits installiertem SSH.

Vorbereitung der Installation

Der Zugriff auf die Quellen ist über die Seiten des Projektes möglich. In dieser Anleitung werden wir wir uns die Installationsquellen über Git kopieren. Installieren Sie zunächst als benötigte Komponenten den Apache2-Webserver, den MySQL-Datenbankserver sowie die weiteren für Icinga erforderlichen Pakete:

```
root@moni-wheezy-icisrc:~# aptitude install git apache2 build-essential libgd2-xpm-dev
libjpeg8 libjpeg8-dev libpng12-0 libpng12-dev mysql-server mysql-client libdbi1
libdbi-dev libdbd-mysql
```

Während der Installation werden Sie nach einem Passwort für MySQL-Benutzer `root` gefragt.

Administrator-Passwort für MySQL: Wenn Sie das hier gesetzte Passwort für den MySQL-Benutzer `root` später ändern möchten, verwenden Sie folgendes Kommando:

```
root@moni-wheezy-nagios:~# mysqladmin -u root -p password
```

Legen Sie anschließend einen System-Benutzer `icinga` an, unter dessen Namen und mit dessen Rechten Icinga laufen soll, und weisen Sie diesem ein Passwort zu:

```
root@moni-wheezy-icisrc:~# useradd -m icinga
root@moni-wheezy-icisrc:~# passwd icinga
```

Eine Benutzergruppe mit dem gleichen Namen wird dabei unter Debian automatisch angelegt und dem Benutzer zugewiesen. Icinga setzt zur Verwaltung der Steuerungsrechte eine weitere Benutzergruppe voraus, die manuell angelegt und sowohl dem Konto für Icinga als auch dem Konto des Webservers zugewiesen werden muss:

```
root@moni-wheezy-icisrc:~# groupadd icinga-cmd
root@moni-wheezy-icisrc:~# usermod -a -G icinga-cmd icinga
root@moni-wheezy-icisrc:~# usermod -a -G icinga-cmd www-data
```

Mit folgenden Anweisungen können Sie sich vergewissern, dass Apache2 und MySQL bereits automatisch nach jedem Bootvorgang gestartet werden:

```
root@moni-wheezy-icisrc:~# find /etc/rc?.d/ -iname "*apache2"
/etc/rc0.d/K01apache2
/etc/rc1.d/K01apache2
/etc/rc2.d/S16apache2
/etc/rc3.d/S16apache2
/etc/rc4.d/S16apache2
/etc/rc5.d/S16apache2
/etc/rc6.d/K01apache2
root@moni-wheezy-icisrc:~# find /etc/rc?.d/ -iname "*mysql"
/etc/rc0.d/K02mysql
/etc/rc1.d/K02mysql
/etc/rc2.d/S17mysql
/etc/rc3.d/S17mysql
/etc/rc4.d/S17mysql
/etc/rc5.d/S17mysql
/etc/rc6.d/K02mysql
```

`insserv`: Wenn ein Service automatisch gestartet werden soll, können Sie dies meist mit dem Kommando `insserv <service>` entsprechend konfigurieren.

Installation von Icinga

Nach diesen vorbereitenden Schritten wechseln Sie zunächst beispielsweise in das Verzeichnis `/usr/src` und fertigen dann mit Hilfe von `git` eine lokale Kopie des `icinga-core`-Repositories an. Anschließend wählen Sie Icinga Version 1.9 als lokale Arbeitskopie:

```
root@moni-wheezy-icisrc:~# cd /usr/src/
root@moni-wheezy-icisrc:/usr/src# git clone git://git.icinga.org/icinga-core.git
root@moni-wheezy-icisrc:/usr/src# cd icinga-core/
root@moni-wheezy-icisrc:/usr/src/icinga-core# git checkout  support/1.9
```

`git`: In der Versionierungsanwendung Git können Versionen mit Hilfe von sogenannten Tags markiert werden. Mittels des Kommandos `git tag` können Sie sich sich alle vorhandenen Tags eines Repositories anzeigen lassen. Auf diese Weise können Sie leicht prüfen, welche Versionen verfügbar sind.

Tarball: Alternativ können Sie auch Icinga aus dem offiziellen Tarball installieren. Laden Sie dazu zuerst die aktuelle Version aus dem Verzeichnis *https://sourceforge.net/projects/icinga/files/icinga* herunter. Anschließend entpacken Sie diese Version und kopieren sie beispielsweise in das Verzeichnis */usr/src* (zum Beispiel mit dem Kommando `tar xzf icinga-1.9.1.tar.gz && mv icinga-1.9.1 /usr/src`).

Jetzt wechseln Sie in das neu angelegte Verzeichnis *icinga-core* beziehungsweise *icinga-VERSION* bei Installation aus dem Tarball, und stoßen die automatische Konfiguration an. Dabei stellt der Parameter `--enable-idoutils` sicher, dass die IDOUtils als Backend ebenfalls kompiliert werden. Sofern hier keine Fehler gemeldet wurden, können Sie dann mit dem Kompilieren der Quellen und der Installation von Icinga fortfahren:

```
root@moni-wheezy-icisrc:/usr/src/icinga-core# ./configure --with-command-group=icinga-cmd
--enable-idoutils
root@moni-wheezy-icisrc:/usr/src/icinga-core# make all
root@moni-wheezy-icisrc:/usr/src/icinga-core# make fullinstall
root@moni-wheezy-icisrc:/usr/src/icinga-core# make install-config
```

Hinweise zu neuen Versionen: (sogenannte Release Announcements): Wenn Sie Icinga über den Quellcode installieren, entfällt die automatische Versorgung mit Aktualisierungen. Tragen Sie sich daher in die entsprechende Mailing-Liste ein (*https://lists.sourceforge.net/lists/listinfo/icinga-users*) oder verfolgen Sie die Release-Blogposts im Icinga-Blog (*https://www.icinga.org/blog/*), um sich über Neuerungen auf dem Laufenden zu halten.

Installation der Standard-Plugins

Icinga führt Tests mit Hilfe von sogenannten Plugins durch. Das Standard-Set besteht derzeit aus den Nagios-Plugins, die – wie Icinga selbst – sowohl über Git als auch als Tarball-Releases erhältlich sind.

Installieren Sie zunächst folgende Pakete, die für die Kompilierung aller Plugins erforderlich sind:

```
root@moni-wheezy-icisrc:~# aptitude install m4 gettext automake autoconf
libradiusclient-ng-dev libgnutls-dev libssl-dev libldap2-dev libmysqlclient-dev
smbclient libnet-snmp-perl snmp libsnmp-dev fping
```

Die Standard-Arbeitsgruppe für Samba können Sie für die Abfrage im Zweifel bei workgroup belassen. Anschließend kopieren Sie sich die Quelltexte dieser Plugins wie folgt und wählen Version 1.4.16 als lokale Arbeitskopie:

```
root@moni-wheezy-icisrc:~# cd /usr/src
root@moni-wheezy-icisrc:/usr/src# git clone https://github.com/nagios-plugins/
nagios-plugins
root@moni-wheezy-icisrc:/usr/src# cd nagiosplug
root@moni-wheezy-icisrc:/usr/src/nagiosplug# git checkout release-1.4.16
```

Sie können diese dann direkt kompilieren, allerdings werden dabei gegebenenfalls Plugins ausgelassen, wenn deren Abhängigkeiten nicht erfüllt sind:

```
root@moni-wheezy-icisrc:/usr/src/nagiosplug# ./tools/setup
root@moni-wheezy-icisrc:/usr/src/nagiosplug# ./configure --prefix=/usr/local/icinga
--with-cgiurl=/icinga/cgi-bin --with-nagios-user=icinga --with-nagios-group=icinga
root@moni-wheezy-icisrc:/usr/src/nagiosplug# make
root@moni-wheezy-icisrc:/usr/src/nagiosplug# make install
```

Abhängigkeiten: Bei welchen Plugins dabei gegebenenfalls Warnungen zu nicht erfüllten Abhängigkeiten vorliegen, können Sie sich anschließend mit dem folgenden Befehl anzeigen lassen:

```
root@moni-wheezy-icisrc:/usr/src/nagiosplug# cat config.log | grep WARNING
```

Installation der Weboberfläche Icinga-Classic

Damit ein Zugriff auf die Installation möglich wird, sollten Sie darüber hinaus das in den Quellen enthaltene Web-Interface installieren. Rufen Sie dazu erneut das Kommando `make` wie folgt auf:

```
root@moni-wheezy-icisrc:/usr/src/icinga-core# make cgis
root@moni-wheezy-icisrc:/usr/src/icinga-core# make install-cgis
root@moni-wheezy-icisrc:/usr/src/icinga-core# make install-html
root@moni-wheezy-icisrc:/usr/src/icinga-core# make install-webconf
```

Richten Sie sich bei der Gelegenheit auch gleich ein Passwort für den in der Konfiguration des Webservers vorgesehenen Schutz durch `http-auth` (siehe Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*) ein:

```
root@moni-wheezy-icisrc:/usr/src/icinga-core# htpasswd -c
/usr/local/icinga/etc/htpasswd.users icingaadmin
```

Anschließend starten Sie den Apache Webserver und Icinga neu:

```
root@moni-wheezy-icisrc:~# service apache2 restart
root@moni-wheezy-icisrc:~# service icinga restart
```

Icinga führt nun bereits erste lokale Checks auf Ihrem Monitoring-System durch. Über Zugriff mit dem Browser auf den URL *http://127.0.0.1/icinga* beziehungsweise über Ihre systemspezifische IP-Adresse sollte sich jetzt die Oberfläche von Icinga-Classic aufbauen. Verschaffen Sie sich einen ersten Eindruck.

Oberflächentest von der Kommandozeile: Sollte der entfernte Webzugriff auf Ihr System (noch) nicht möglich sein, beispielsweise aufgrund einer Firewall, dann können Sie auch über *http://127.0.0.1/nagios/* auf Ihr frisch installiertes Nagios zugreifen. Wenn Sie über keine graphische Oberfläche und bzw. oder keinen lokalen Webbrowser verfügen, können Sie die Erreichbarkeit der Oberfläche durch folgende Anweisung mit dem Kommandozeilen-Werkzeug `telnet` testen:

```
root@moni-wheezy-icisrc:~# telnet localhost 80
[...]
GET /icinga
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
[...]
```

Entsprechend der automatisch generierten Konfiguration wird die lokale Maschine bereits mit einigen Diensten überwacht, wobei es nach dem Start gegebenenfalls einige Minuten dauert, bevor hierzu Ergebnisse angezeigt werden. Stellen Sie nun abschließend noch sicher, dass Icinga bei einem Neustart automatisch gestartet wird:

```
root@moni-wheezy-icisrc:~# insserv icinga
root@moni-wheezy-icisrc:~# find /etc/rc?.d/ -iname "*icinga"
/etc/rc0.d/K01icinga
/etc/rc1.d/K01icinga
/etc/rc2.d/S19icinga
/etc/rc3.d/S19icinga
/etc/rc4.d/S19icinga
/etc/rc5.d/S19icinga
/etc/rc6.d/K01icinga
```


Installation der Datenbankanbindung über IDOUtils

Wir zeigen im Folgenden, wie Sie die Installation um das Addon IDOUtils erweitern und an den im vorhergehenden Schritt bereits installierten MySQL-Datenbankserver anbinden. Diese Anbindung benötigen Sie unter anderem zur Nutzung der Erweiterungen Icinga-Web und NagVis, auf die wir weiter unten eingehen. Als ersten Schritt legen Sie dazu eine leere Datenbank mit dem Namen `icinga` an und erstellen einen MySQL-Benutzer mit den nötigen Rechten (hier `sql_icinga`) und einem Passwort (hier `DBUSERPW`):

Postgresql und Co.: Sie können mit den IDOUtils, und ebenfalls mit Icinga-Web, auch MariaDB, Postgresql, oder Oracle als Datenbankbackend einsetzen. Die Konfiguration funktioniert dann weitestgehend analog.

```
root@moni-wheezy-icisrc:~# mysql -u root -p
[...]
mysql> CREATE DATABASE icinga;
mysql> GRANT USAGE ON icinga.* TO 'sql_icinga'@'localhost' IDENTIFIED BY 'DBUSERPW' WITH
```

```
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0;
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, DROP, CREATE VIEW, INDEX, EXECUTE ON
icinga.* TO 'sql_icinga'@'localhost';
mysql> FLUSH PRIVILEGES;
mysql> exit
```

MySQL-Passwörter: Wenn Sie das hier gesetzte Passwort für den root-Nutzer von MySQL ändern möchten, funktioniert das folgendermaßen:

```
root@moni-wheezy-icisrc:~# mysqladmin -u root -p password
```

Um das Passwort eines nicht-administrativen MySQL-Benutzers zu ändern, können Sie sich von der Kommandozeile aus als Benutzer root auf dem Server anmelden und dann das hier nachfolgend angeführte Kommando nutzen:

```
root@moni-wheezy-icisrc:~# mysql -u root -p
mysql> use mysql;
mysql> update user set Password=DBUSERPW('NEWDBUSERPW') WHERE
User='DBUSER';
mysql> exit
```

Importieren Sie dann die in den Quellen enthaltene Struktur in diese Datenbank:

```
root@moni-wheezy-icisrc:~# mysql -u root -p icinga < /usr/src/icinga-core/module/
idoutils/db/mysql/mysql.sql
```

Datenbank-Updates: Wenn Sie später die IDOUtils auf eine aktuelle Version aktualisieren möchten, finden Sie im Unterverzeichnis /usr/src/icinga-core/module/idoutils/db/mysql/upgrade die entsprechenden SQL-Skripte, die Sie dann inkrementell einzuspielen müssen.

Bei der Installation wurde für Sie eine Datei zur Konfiguration der Zugangsdaten als Vorlage erstellt. Kopieren Sie diese zunächst und entfernen Sie dabei die Endung »-sample«:

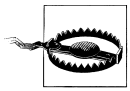
```
root@moni-wheezy-icisrc:~# cp /usr/local/icinga/etc/ido2db.cfg-sample
/usr/local/icinga/etc/ido2db.cfg
```

Die Zugangsdaten für die Datenbank müssen Sie dann in dieser Datei hinterlegen:

```
[...]
db_user=sql_icinga
db_pass=DBUSERPW
[...]
```

Wenn Sie einen Datenbankserver auf einer anderen Maschine verwenden möchten, müssten Sie hier weitere Angaben machen. Damit sind Icinga und IDO2DB ansonsten installiert. Sie müssen Icinga allerdings noch darüber informieren, dass die Daten an das dafür vorgesehene Modul idomod übergeben werden sollen. Aktivieren Sie dazu zunächst die mitgelieferten Konfigurationsvorlage für das Broker-Modul durch Kopieren und Umbennenen der entsprechenden Dateien:

```
root@moni-wheezy-icisrc:~# cp /usr/local/icinga/etc/idomod.cfg-sample
/usr/local/icinga/etc/idomod.cfg
root@moni-wheezy-icisrc:~# cp /usr/local/icinga/etc/modules/idoutils.cfg-sample
/usr/local/icinga/etc/modules/idoutils.cfg
```



Eventbroker-Module: Seit IDOUtils Version 1.4 sollte das Eventbroker-Modul `idomod` über eine Modules-Definition im *modules*-Unterverzeichnis */etc/icinga/modules/* eingebunden werden. Sie sollten daher nicht zusätzlich den entsprechenden Eintrag `broker_modul` in der Hauptkonfigurationsdatei `/etc/icinga/icinga.cfg` aktivieren, da es in diesem Fall, verursacht durch die Doppeldefinition, zu schwer zu analysierenden Fehlern kommen kann.

Für einen ersten Test können Sie beide Teile starten (IDO2DB muss vor Icinga gestartet werden):

```
root@moni-wheezy-icisrc:~# service ido2db start
root@moni-wheezy-icisrc:~# service icinga restart
```

Nun läuft Icinga und die Daten werden über das Modul IDO2DB in die neu angelegte Datenbank geschrieben.

Abschließend müssen Sie dem System nur noch mitteilen, dass dieser Dienst automatisch gestartet werden soll, und überprüfen, ob die entsprechenden Verweise auf die Startdateien korrekt eingerichtet wurden:

```
root@moni-wheezy-icisrc:~# insserv ido2db
root@moni-wheezy-icisrc:~# find /etc/rc?.d/ -iname "*ido2db"
/etc/rc0.d/K02ido2db
/etc/rc1.d/K02ido2db
/etc/rc2.d/S18ido2db
/etc/rc3.d/S18ido2db
/etc/rc4.d/S18ido2db
/etc/rc5.d/S18ido2db
/etc/rc6.d/K02ido2db
```

Damit ist die grundlegende Installation von Icinga abgeschlossen. Die Dienste laufen jetzt nach jedem Start automatisch.

Prüfen Sie nun, ob auch tatsächlich Daten in die angelegte Datenbank geschrieben werden. Sie können sich hierzu vergewissern, dass die Icinga-Datenbank kürzlich aktualisiert wurde. Lesen Sie hierzu den Eintrag `status_update_time` aus der Tabelle `icinga_programstatus` aus:

```
root@moni-wheezy-icisrc:~# mysql -u sql_icinga -p
mysql> USE icinga;
mysql> SELECT status_update_time FROM icinga_programstatus;
+---------------------+
| status_update_time |
+---------------------+
| 2012-12-02 08:59:21 |
+---------------------+
1 row in set (0.00 sec)
mysql> exit
```

Damit läuft Icinga und die Daten werden über das Modul IDO2DB in die neu angelegte Datenbank geschrieben.

Installation der Weboberfläche Icinga-Web

Neben der klassischen Nutzerschnittstelle Icinga-Classic bietet Icinga eine zweite, neu entwickelte Weboberfläche names Icinga-Web. Sie können beide Oberflächen problemlos parallel betreiben. Im Gegensatz zu Icinga-Classic, das die aus Nagios bekannten `status.dat/objects.cache`-Dateien verwendet, verwendet Icinga-Web die IDOUtils-Datenbank.

Installieren Sie zunächst die folgenden, zusätzlich erforderlichen Pakete:

```
root@moni-wheezy-icisrc:~# aptitude install php5 php5-cli php-pear php5-xmlrpc php5-xsl
php5-gd php5-ldap php5-mysql
```

In unserem Falle war hier zuvor der Apache über das Paket `apache2-mpm-worker` installiert. Da PHP5 an dieser Stelle jedoch stattdessen die Installation des Pakets `apache2-mpm-prefork` bedingt, wird gegebenenfalls eine entsprechende Ersetzung vorgenommen. Die Quellen erhalten Sie am einfachsten wieder über Git oder über die Icinga-Webseite als Tarball-Release. Für die Installation über Git legen Sie sich als erstes in */usr/src* eine Kopie des Icinga-Web-Repositories an:

```
root@moni-wheezy-icisrc:~# cd /usr/src/
root@moni:/usr/src# git clone git://git.icinga.org/icinga-web.git
```

Um die Dateien auf den Stand von Version 1.9 zu bringen, führen Sie in dem neu erstellen Verzeichnis *icinga-web* die folgende Anweisung aus:

```
root@moni-wheezy-icisrc:/usr/src# cd icinga-web/
root@moni-wheezy-icisrc:/usr/src/icinga-web# git checkout r1.9
```

Lassen Sie dann die systemspezifischen Anpassungen unter Nutzung der vom Projekt vorgeschlagenen Optionen laufen:

```
root@moni-wheezy-icisrc:/usr/src/icinga-web# ./configure
--with-api-cmd-file=/usr/local/icinga/var/rw/icinga.cmd
--with-conf-dir=/etc/icinga-web --with-log-dir=/var/log/icinga-web
--with-cache-dir=/var/cache/icinga-web
```

Dabei werden möglicherweise Warnungen angezeigt, die Sie in den meisten Fällen aber ignorieren können. Starten Sie die Übersetzung und die Installation der einzelnen Komponenten wie folgt. Beginnen Sie mit den eigentlichen Webseiten und installieren Sie anschließend die Konfigurationsdatei für den Apache-Server:

```
root@moni-wheezy-icisrc:/usr/src/icinga-web# make install
root@moni-wheezy-icisrc:/usr/src/icinga-web# make install-apache-config
```

Unter Debian muss dann zunächst das Modul `rewrite` für den Apache aktiviert werden. Anschließend müssen Sie den Webserver neu starten. Dies erledigen Sie folgendermaßen:

```
root@moni-wheezy-icisrc:~# a2enmod rewrite
root@moni-wheezy-icisrc:~# service apache2 restart
```

Die Weboberfläche ist darauf ausgelegt, Daten (Benutzer, Sitzungen und Einstellungen) in einer eigenen Datenbank zu speichern. Sie benötigen diese Datenbank zusätzlich zu der für IDOUtils installierten. Legen Sie die neue Datenbank zunächst unter Nutzung des MySQL-Benutzers `root` an und richten Sie dabei weiter einen entsprechenden MySQL-Benutzer `sql_icinga-web` für den Zugriff ein (verwenden Sie dabei statt des hier genutzten Passworts `DBUSERPW` bitte ein eigenes):

```
root@moni-wheezy-icisrc:/usr/src/icinga-web# mysql -u root -p
[...]
mysql> CREATE DATABASE icinga_web;
mysql> GRANT USAGE ON icinga_web.* TO 'sql_icinga-
web'@'localhost' IDENTIFIED BY 'DBUSERPW' WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_
PER_HOUR 0 MAX_UPDATES_PER_HOUR 0;
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, INDEX ON icinga_web.*
TO 'sql_icinga-web'@'localhost';
mysql> FLUSH PRIVILEGES;
mysql> EXIT;
```

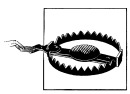
In die erstellte aber noch leere Datenbank können Sie jetzt die vorgegebene Struktur importieren (hier mit dem oben angelegten MySQL-Benutzer `sql_icinga-web`):

```
root@moni-wheezy-icisrc:~# mysql -u sql_icinga-web -p icinga_web < /usr/src/icinga-
web/etc/schema/mysql.sql
```

Datenbankupdates: Bei einer späteren Aktualisierung von Icinga-Web müssen Sie, wie auch bei den IDOUtils, die Datenbank von Icinga-Web auf eine aktuelle Version aktualisieren. Die entsprechenden SQL-Scripte finden Sie hier im Unterverzeichnis `/usr/src/icinga-web/etc/schema/updates`. Diese müssen Sie dann inkrementell einspielen.

Die Zugriffsdaten für diese Datenbank müssen Sie jetzt noch in der Datei */etc/icinga-web/databases.xml* hinterlegen. Dazu benötigen Sie zunächst die Zugangsdaten (Benutzername und Passwort) für die Datenbank `icinga_web` und anschließend die Zugangsdaten für die Datenbank `icinga`, die für IDOUtils erstellt wurde:

```
[...]
 <db:database name="icinga_web" class="AppKitDoctrineDatabase">
 <ae:parameter name="dsn">mysql://sql_icinga-web:
DBUSERPW@localhost:3306/icinga_web</ae:parameter>
[...]
 </ae:parameter>
 </db:database>
[...]
 <db:database xmlns="http://agavi.org/agavi/config/parts/databases/1.0" name="icinga"
class="IcingaDoctrineDatabase">
 <ae:parameter name="dsn">mysql://sql_icinga:DBUSERWD@localhost:3306/icinga</ae:parameter>
[...]
 </ae:parameter>
 </db:database>
[...]
```



Zugang verwehrt: Die betroffenen Konfigurationsabschnitte in der Datei *databases.xml* könnten durch die Zeichenketten `<!--` und `-->` auskommentiert sein. Achten Sie darauf, die entsprechenden Kommentarzeichen zu entfernen, anderenfalls kann Icinga-Web nicht auf die Zugangsdaten zugreifen und wird Ihnen den Zugang verwehren. Ein Editor mit Syntax-Highlightning, wie beispielsweise `vim` mit `syntax on`, hilft Ihnen bei der Identifizierung von auskommentierten Blöcken.

Nach Änderungen an den Konfigurationsdateien von Icinga-Web müssen Sie immer das Script zum Löschen der Cache-Dateien ausführen:

```
root@moni-wheezy-icisrc:~# /usr/local/icinga-web/bin/clearcache.sh
```

Damit ist die Installation der zusätzlichen Oberfläche abgeschlossen. Starten Sie jetzt den Webserver neu, damit er die hinzugekommene Konfigurationsdatei einliest:

```
root@moni-wheezy-icisrc:~# service apache2 restart
```

Sie können sich die neue Oberfläche jetzt unter *http://127.0.0.1/icinga-web* anschauen (siehe dazu Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*). Loggen Sie sich mit dem Benutzernamen `root` und dem Passwort `password` erstmalig ein. Als Erstes sollten Sie dann dieses Passwort ändern. Das entsprechende Menü finden rechts oben.

Keine Daten: Falls Sie vor dem Problem stehen, dass Icinga-Web keine Daten anzeigt, stellen Sie sicher, dass IDOUtils und Icinga korrekt funktionieren. Weitere Tests finden Sie im Icinga Wiki: *https://wiki.icinga.org/display/testing/Icinga+Web+Testing*.

Installation einer graphischen Auswertung mit PNP4Nagios

Bei PNP4Nagios handelt es sich um eine Erweiterung, mit der Performance-Daten aufgezeichnet werden, aus denen sich dann Graphen generieren lassen. Für die Speicherung der Daten verwendet dieses Addon ein Werkzeug namens RRDtool (Round Robin Database Tool). Installieren Sie dieses deshalb zunächst zusammen mit der ebenfalls erforderlichen Anbindung an Perl:

```
root@moni-wheezy-icisrc:~# aptitude install librrds-perl rrdtool
```

Am besten beziehen Sie die Quellen auch hier wieder über Git. Legen Sie sich zunächst in */usr/src* eine Kopie der PNP4Nagios-Quellen an:

```
root@moni-wheezy-icisrc:~# cd /usr/src/
root@moni-wheezy-icisrc:/usr/src# git clone
git://pnp4nagios.git.sourceforge.net/gitroot/pnp4nagios/pnp4nagios
```

Um die Dateien auf den Stand von Version 0.6.19 zu bringen, führen Sie in dem neu erstellten Verzeichnis *pnp4nagios* die folgende Anweisung aus:

```
root@moni-wheezy-icisrc:/usr/src# cd pnp4nagios/
root@moni-wheezy-icisrc:/usr/src/pnp4nagios# git checkout 0.6.19
```

Lassen Sie dann die Anpassung für Ihr System, die Übersetzung sowie die Installation laufen, um PNP4Nagios auf Ihrem System zu installieren:

```
root@moni-wheezy-icisrc:/usr/src/pnp4nagios# ./configure --with-nagios-user=icinga
--with-nagios-group=icinga
root@moni-wheezy-icisrc:/usr/src/pnp4nagios# make all
root@moni-wheezy-icisrc:/usr/src/pnp4nagios# make fullinstall
```

Bei der Option `fullinstall` installieren Sie dabei den NPCD (Nagios Perfdata C Daemon) mit. PNP4Nagios unterstützt eine ganze Reihe von Betriebsmodi (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/modes*). Der Modus mit NPCD wird vom PNP4Nagios-Projekt mit »Dies ist aus Nagios-Sicht die sauberste Art der Verarbeitung« beschrieben, weshalb wir diesen Modus im Folgenden beschreiben.

Für die Konfiguration der Aufbewahrungszeiten der Daten wurde eine Beispieldatei installiert, die Sie durch Kopieren übernehmen können:

```
root@moni-wheezy-icisrc:~# cp /usr/local/pnp4nagios/etc/rra.cfg-sample
/usr/local/pnp4nagios/etc/rra.cfg
```

In der Konfigurationsdatei */usr/local/pnp4nagios/etc/config.php* müssen Sie dann anschließend eine Anpassung für Icinga vornehmen:

```
$conf['nagios_base'] = "/icinga/cgi-bin";
```

Eine ähnliche Ersetzung ist auch in der Konfigurationsdatei für den Webserver unter */etc/apache2/conf.d/pnp4nagios.conf* erforderlich:

```
AuthUserFile /usr/local/icinga/etc/htpasswd.users
```

Damit Icinga überhaupt Performance-Daten weiterleitet, aktivieren Sie die entsprechende Option in */usr/local/icinga/etc/icinga.cfg* und richten Sie die Umleitung in eine Datei wie folgt ein:

```
[...]
# PROCESS PERFORMANCE DATA OPTION
[...]
process_performance_data=1
[...]
# HOST AND SERVICE PERFORMANCE DATA FILES
[...]
host_perfdata_file=/usr/local/pnp4nagios/var/host-perfdata
service_perfdata_file=/usr/local/pnp4nagios/var/service-perfdata
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE TEMPLATES
[...]
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::
  $TIMET$\tHOSTNAME::$HOSTNAME$\tSERVICEDESC::$SERVICEDESC$\tSERVICEPERFDATA::
  $SERVICEPERFDATA$\tSERVICECHECKCOMMAND::$SERVICECHECKCOMMAND$\tHOSTSTATE::
  $HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$\tSERVICESTATE::
  $SERVICESTATE$\tSERVICESTATETYPE::$SERVICESTATETYPE$
host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET::$TIMET$\tHOSTNAME::
  $HOSTNAME$\tHOSTPERFDATA::$HOSTPERFDATA$\tHOSTCHECKCOMMAND::
  $HOSTCHECKCOMMAND$\tHOSTSTATE::$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE MODES
[...]
host_perfdata_file_mode=a
service_perfdata_file_mode=a
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING INTERVAL
[...]
host_perfdata_file_processing_interval=30
service_perfdata_file_processing_interval=30
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING COMMANDS
[...]
host_perfdata_file_processing_command=process-host-perfdata-file
service_perfdata_file_processing_command=process-service-perfdata-file
[...]
```

Sie richten hiermit zunächst ein, dass Icinga Performancedaten wie in der Vorlage mit Makros spezifiziert in die angegebene Datei schreibt. Die unteren Zeilen veranlassen Icinga, weiter in den angegebenen Abständen die Kommandos `process-host-perfdata-file` und `process-service-perfdata-file` aufzurufen, um die eigentliche Verarbeitung durchzuführen. Diese beiden Kommandos definieren Sie dann noch wie folgt in der Datei */usr/local/icinga/etc/objects/commands.cfg* (zu Kommandodefinitionen und Makros siehe auch Rezept 4.3, *Befehle hinzufügen (command)*):

```
# 'process-host-perfdata' command definition
define command {
       command_name                            process-host-perfdata-file
       command_line                            /bin/mv /usr/local/pnp4nagios/var/ \
host-perfdata /usr/local/pnp4nagios/var/spool/host-perfdata.$TIMET$
}

# 'process-service-perfdata' command definition
define command {
       command_name                            process-service-perfdata-file
       command_line                            /bin/mv /usr/local/pnp4nagios/var/ \
service-perfdata /usr/local/pnp4nagios/var/spool/service-perfdata.$TIMET$
}
```

Auf diese Weise wurde eingerichtet, dass die von Icinga erstellten Dateien in regelmäßigen Abständen in das Verzeichnis */usr/local/pnp4nagios/var/spool* verschoben werden. Von dort wird sie dann der NPCD abrufen und verarbeiten. Dies ist wiederum über die folgende Konfigurationsoption in der Datei */usr/local/pnp4nagios/etc/npcd.cfg* voreingestellt:

```
perfdata_spool_dir = /usr/local/pnp4nagios/var/spool
```

Starten Sie jetzt den NPCD und führen Sie einen Neustart von Icinga und des Webservers durch, um die Änderungen zu übernehmen:

```
root@moni-wheezy-icisrc:~# service npcd start
root@moni-wheezy-icisrc:~# service apache2 restart
root@moni-wheezy-icisrc:~# service icinga restart
```

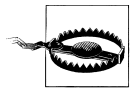
Rufen Sie nun zunächst die PNP4Nagios-Oberfläche unter *http://127.0.0.1/pnp4nagios* beziehungsweise der jeweiligen IP-Adresse auf. Nutzen Sie die für Icinga gesetzten Zugangsdaten, um sich anzumelden. Dabei werden Test zur Einrichtung ausgeführt. Wenn alle Tests den Status OK aufweisen, können Sie anschließend wie aufgefordert die Datei umbenennen, die diese Tests durchgeführt hat:

```
root@moni-wheezy-icisrc:~# mv /usr/local/pnp4nagios/share/install.php
/usr/local/pnp4nagios/share/install.php.orig
```

Sobald Tests von Icinga durchgeführt werden, für die in PNP4Nagios eine passende Konfiguration vorhanden ist, werden Sie dann nach einigen Minuten entsprechende Graphen vorfinden. Dies liegt daran, dass Icinga die zu verarbeitenden Daten erst einmal sammeln und weiterleiten muss.

Stellen Sie nun abschließend sicher, dass der NPCD nach einem Reboot automatisch gestartet wird:

```
root@moni-wheezy-nagsrc:~# insserv npcd
```



Keine Graphen: Wenn PNP4Nagios keine Performancedaten findet, können auch keine Graphen generiert werden. Prüfen Sie bei fehlenden Graphen zunächst, ob von den entsprechenden Check-Plugins tatsächlich Performancedaten erzeugt werden.

Damit haben Sie die Installation der Erweiterung PNP4Nagios abgeschlossen. Per Standardeinstellung werden nun für jedes Gerät und jeden Dienst Performancedaten verarbeitet, sofern diese vom Plugin geliefert werden.

Integration von PNP4Nagios in Icinga-Classic

Damit Sie unter der Weboberfläche Icinga-Classic ebenfalls auf die Graphen zugreifen können, müssen die betroffenen Konfigurationen angepasst werden. Legen Sie sich zunächst, wie im Folgenden beschrieben, Vorlagen in der Datei */usr/local/icinga/etc/objects/templates.cfg* an (siehe hierzu Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*) und referenzieren Sie diese dann in den vorhandenen Definitionen `generic-host` und `generic-service` (siehe Rezept 4.1, *Maschinen einbinden (host)* für die Definition von Maschinen, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* für die Definition von Services und Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System* für die Verwendung des Vorlagensystems):

```
[...]
# HOST TEMPLATES
[...]
define host{
       name                                    template-host-generic
       use                                     pnp-hst
[...]
define host {
       name                                    pnp-hst
       register                                0
       action_url                              /pnp4nagios/graph?host=$HOSTNAME$' \
class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=_HOST_
}
[...]
# SERVICE TEMPLATES
[...]
define service{
       name                                    generic-service
       ; The 'name' of this service template
       use                                     pnp-svc
[...]
define service {
       name                                    pnp-svc
       register                                0
       action_url                              /pnp4nagios/graph?host=$HOSTNAME$&srv= \
$SERVICEDESC$' class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=$SERVICEDESC$
}
[...]
```

Hierdurch werden entsprechende Graphiken als Icons für alle Geräte und Services angezeigt, die mit der entsprechenden PNP4Nagios-Seite verlinkt sind. Allerdings finden sich

entsprechend auch Icons für solche Dienste, die möglicherweise gar keine Performance-Daten liefern und zu denen es deshalb auch keine Graphen gibt. Sie können sich über die aufgerufene Seite dann aber zumindest schnell zu den vorhandenen Graphen der jeweiligen Maschine weiterklicken.

Differenzierung: Hierbei werden entsprechende Icons für alle Geräte und Services definiert, auch für solche ohne Graphen. Alternativ besteht an dieser Stelle auch die Möglichkeit, die Vorlage `pnp-svc` nur bei den Diensten einzubinden, für die Sie Graphen haben oder anzeigen möchten.

Damit die Graphen bereits dann als Vorschau angezeigt werden, wenn Sie mit der Maus darüberfahren, also beim sogenannten `mouse-over`-Ereignis, kopieren Sie die als Vorlage vorhandene Konfigurationsdatei und passen die Zugriffrechte an:

```
root@moni-wheezy-icisrc:~# cp /usr/src/pnp4nagios/contrib/ssi/status-header.ssi
/usr/local/icinga/share/ssi/
root@moni-wheezy-icisrc:~# chmod 644 /usr/local/icinga/share/ssi/status-header.ssi
```

Damit Icinga die geänderten Dateien einliest, ist noch ein Neustart erforderlich:

```
root@moni-wheezy-icisrc:~# service icinga restart
```

Nun weist die klassischen Oberfläche Symbole für die Graphen auf. Beim Darüberfahren mit der Maus sehen Sie eine Vorschau und durch Klicken gelangen Sie auf die Seite von PNP4Nagios mit den entsprechenden Graphen.

Integration von PNP4Nagios in Icinga-Web

Die Integration mit Icinga-Web ist deutlich einfacher. Kopieren Sie einfach die folgende Vorlage und führen Sie anschließend das Script zum Löschen der Cache-Dateien aus:

```
root@moni-wheezy-icisrc:~# cp /usr/src/icinga-web/contrib/PNP_Integration/
templateExtensions/* /usr/local/icinga-web/app/modules/Cronks/data/xml/extensions/
root@moni-wheezy-icisrc:~# /usr/local/icinga-web/bin/clearcache.sh
```

Damit enthält auch die Oberfläche von Icinga-Web Symbole für die Graphen und Sie können diese abrufen.

`Expander Icon`: Icinga-Web zeigt die PNP-Icons seit Version 1.8 im sogenannten. `Expander Icon` an, wo diese dann in die Hauptansicht verschoben werden können (`Move to Grid`). Weiter ist zu beachten, dass die Icons nur dann angezeigt werden, wenn in der Konfiguration des Host/Service `process_perf_data 1` gesetzt ist.

Installation des Visalisierungsaddons NagVis

Das Addon NagVis ermöglicht Ihnen, Symbole für Maschinen und Services beliebig anzuordnen. Durch die Wahl eines entsprechenden Hintergrundes erhalten die Symbole dabei einen Mehrwert, etwa durch die Platzierung auf Fotos oder Karten. Machen Sie sich zunächst mittels Git eine Kopie von NagVis:

```
root@moni-wheezy-icisrc:~# cd /usr/src/
root@moni-wheezy-icisrc:/usr/src# git
clone git://nagvis.git.sourceforge.net/gitroot/nagvis/nagvis
```

Wählen Sie dann die zu nutzende Version (hier beispielsweise 1.7.4):

```
root@moni-wheezy-icisrc:~# cd /usr/src/nagvis
root@moni-wheezy-icisrc:/usr/src/nagvis# git checkout nagvis-1.7.4
```

NagVis benötigt für den Zugriff auf die Daten das Paket `php5-sqlite` und für die Darstellung das Paket `graphviz`. Installieren Sie diese mit Hilfe des Paketmanagements:

```
root@moni-wheezy-icisrc:~# aptitude install php5-sqlite graphviz
```

Anschließend lassen sich die entsprechenden Dateien installieren. Sie können dabei alle abgefragten Vorgabewerte akzeptieren:

```
root@moni-wheezy-icisrc:/usr/src/nagvis# ./install.sh -s icinga -n /usr/local/icinga -p
/usr/local/nagvis -u www-data -g www-data -i ido2db -m /usr/local/icinga/bin/ido2db
```

Hinterlegen Sie dann zunächst die Zugangsdaten zu der oben über IDO2DB eingebundenen Datenbank in der Datei */usr/local/nagvis/etc/nagvis.ini.php*:

```
dbhost="localhost"
dbport=3306
dbname="icinga"
dbuser="sql_icinga"
dbpass="DBUSERPW"
dbprefix="icinga_"
dbinstancename="default"
```

Starten Sie im Anschluss daran den Webserver neu:

```
root@moni-wheezy-icisrc:~# service apache2 restart
```

Sie können sich nun bereits unter *http://127.0.0.1/nagvis* am System anmelden. Bei der Installation wurde dazu ein administratives Benutzerkonto `admin` eingerichtet, dessen Passwort `admin` Sie jetzt sofort ändern sollten. Für den weiteren Umgang mit PNP4Nagios und NagVis lesen Sie bitte die Rezepte zur Datenvisualisierung in (Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*).

Diskussion

Die Installation aus den Quellen ist im Vergleich zur Installation über das Paketmanagementsystem vergleichsweise aufwendig. Dafür bieten sich Möglichkeiten, die Installation anzupassen und etwa nicht benötigte Plugins wegzulassen. Sie können diese unterschiedlichen Installationsmethoden allerdings nicht ohne Weiteres mischen, um etwa zusätzliche im Paketmanagement-System enthaltene Plugins zu installieren.

Alternativ zu dieser Installation aus den Quellen haben wir für Sie die paketbasierte Installation von Icinga unter verschiedenen Distributionen beschrieben:

- Rezept 1.5, *Icinga unter Debian installieren*
- Rezept 1.6, *Icinga unter Ubuntu installieren*
- Rezept 1.7, *Icinga unter SLES|openSUSE installieren*
- Rezept 1.8, *Icinga unter RHEL|CentOS|Fedora installieren*

Icinga nutzte bei dieser Installation die in Tabelle 1-3 aufgeführten Speicherorte. Die Ablageorte für diverse Komponenten und Konfigurationsdateien sind für Sie von Bedeutung, damit Sie schnell die richtige Stelle für gewünschte Änderungen finden können.

Tabelle 1-3: Ablageorte der Komponenten von Icinga

Komponente	Pfad
Programm	/usr/local/icinga/bin/icinga
Haupt-Konfigurationsdatei	/usr/local/icinga/etc/icinga.cfg
Plugins	/usr/local/icinga/libexec/
Logdatei	/usr/local/icinga/var/icinga.log
Webserverkonfiguration	/etc/apache2/conf.d/icinga.conf
Webseiten	/usr/local/icinga/share/

Das Addon IDOUtils nutzt die in Tabelle 1-4 aufgeführten Speicherorte.

Tabelle 1-4: Pfade der wichtigsten Dateien von IDOUtils

Komponente	Pfad
Konfigurationsdatei ido2db	/usr/local/icinga/etc/ido2db.cfg
Konfigurationsdatei idomod	/usr/local/icinga/etc/idomod.cfg
Konfigurationsdatei Event-Broker-Modul	/usr/local/icinga/etc/modules/idoutils.cfg
Debug-Logs	/usr/local/icinga/var/*.debug

Das Addon Icinga-Web verwendet die in Tabelle 1-5 aufgeführten Speicherorte.

Tabelle 1-5: Pfade zu den Komponenten von Icinga-Web

Komponente	Pfad
Haupt-Konfigurationsdateien	/etc/icinga-web/
Webserver-Konfiguration	/etc/apache2/conf.d/icinga-web.conf
Webseiten	/usr/local/icinga-web/pub/

PNP4Nagios nutzt die in Tabelle 1-6 genannten Speicherorte.

Tabelle 1-6: Pfade zu den Komponenten von PNP4Nagios

Komponente	Pfad
Konfigurationsdateien	usr/local/pnp4nagios
Haupt-Konfigurationsdatei	/usr/local/pnp4nagios/etc/config.php
NPCD-Konfigurationsdatei	/usr/local/pnp4nagios/etc/npcd.cfg
Graphendefinitionen und Vorlagen	/usr/local/pnp4nagios/share/
Webserver-Konfiguration	/etc/apache2/conf.d/pnp4nagios.conf
Webseiten	/usr/local/pnp4nagios/share/
Perfdata-Script zur Verarbeitung	/usr/local/pnp4nagios/libexec/process_perfdata.pl
Perfdata-Konfigurationsdatei	/usr/local/pnp4nagios/etc/process_perfdata.cfg

Das Addon NagVis verwendet die in Tabelle 1-7 aufgeführten Speicherorte.

Tabelle 1-7: Pfade zu den Komponenten von NagVis

Komponente	Pfad
Haupt-Konfigurationsdatei	/usr/local/nagvis/etc/nagvis.ini.php
Webserver-Konfiguration	/etc/apache2/conf.d/nagvis.conf
Webseiten	/usr/local/nagvis/share/

Auf die Nagios-Plugins werden wir im Rahmen der Rezepte im Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen* und Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen* genauer eingehen. Die Erweiterung NRPE (Nagios Remote Plugin Executor) erklären wir in den Rezepten Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*.

Siehe auch

- Rezept mit Entscheidungshilfe zur Installation: Rezept 1.1, *Entscheidungshilfe für die Installation*
- Webseite des Icinga-Projekts: *https://www.icinga.org/*
- Download-Seite von Icinga auf Sourceforge: *http://sourceforge.net/projects/icinga/*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Download-Seite der Nagios-Plugins: *http://sourceforge.net/projects/nagiosplug/files/nagiosplug/*
- Quickstart-Dokumentation zur Installation: *http://docs.icinga.org/latest/en/quickstart-icinga.html*
- Dokumentation zur Installation mit IDOUtils: *http://docs.icinga.org/latest/en/quickstart-idoutils.html*
- Icinga Fehlersuche: *https://wiki.icinga.org/display/testing/Icinga+Web+Testing*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Dokumentation der Betriebsmodi von PNP4Nagios: *http://docs.pnp4nagios.org/de/pnp-0.6/modes*
- Webseite von NagVis: *http://www.nagvis.org/*
- Webseite des Debian Projekts: *http://www.debian.org/*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zum Aufbau und zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*

- Kapitel zur Konfiguration von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere das Rezept 4.3, *Befehle hinzufügen (command)*, Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* sowie Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Kapitel zur Datenvisualisierung mit PNP4Nagios und NagVis: Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*
- Rezepte zur paketbasierten Installation von Icinga: Rezept 1.5, *Icinga unter Debian installieren*, Rezept 1.6, *Icinga unter Ubuntu installieren*, Rezept 1.7, *Icinga unter SLES|openSUSE installieren* und Rezept 1.8, *Icinga unter RHEL|CentOS|Fedora installieren*
- Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezepte zur Installation von NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

1.5 Icinga unter Debian installieren

Problem

Sie möchten Icinga unter Debian paketbasiert installieren.

Lösung

Wir beschreiben Ihnen zunächst die Installation von Icinga selbst. Anschließend zeigen wir Ihnen die Installation der Datenbankanbindung mit Hilfe des Addons IDOUtils (Icinga Data Out Utilities). Die in einer Datenbank gesammelten Daten können Sie später von weiteren Addons analysieren und ausgewerten lassen. Als entsprechendes Beispiel zeigen wir Ihnen die Installation der Addons PNP4Nagios (Graphen) und NagVis (Karten).

Installation von Icinga

Die aktuelle stabile Version von Debian besitzt den Codenamen Wheezy (Debian 7). Debian Wheezy enthält Icinga als fertiges Paket ab Version 1.7.1. Die Addons PNP4Nagios (ab Version 0.6.12) und NagVis (ab Version 1.6.6) sind ebenfalls standardmäßig vorhanden. Die Einbindung eines weiteren Repositories ist nicht notwendig.

debmon-repository: Die aktuellsten Paket-Versionen erhalten Sie über das Repository des Debian Monitoring-Projektes (*http://www.debmon.org/*). Sie müssen hier allerdings mit einer im Vergleich zur Installation von den Standardpackages erhöhten Anzahl von notwendigen Anpassungen rechnen.

apt-get & aptitude: Als Benutzerschnittstellen zum eigentlich Paketmanager `dpkg` stehen Ihnen sowohl `apt-get` als auch `aptitude` zur Verfügung, die funktional äquivalent sein sollten. Wir verwenden in diesem Buch aus Gründen der Einheitlichkeit stets `aptitude`.

Basierend auf einer 64bit-SSH-Server Installation von Debian können Sie mit folgenden Anweisungen zunächst eine laufende Icinga-Umgebung installieren.

Deinstallation: Sofern Sie eine Deinstallation erwarten und dabei sichergehen möchten, dass alle mitinstallierten Pakete wieder entfernt werden, sollten Sie sich an dieser Stelle die Liste der auf Ihrem System mitinstallierten Pakete durch Kopieren notieren. Der Hintergrund dabei ist der, dass bei der Deinstallation über Paketmanager regelmäßig ganze Pakete auf dem System verbleiben (genau dann nämlich, wenn diese in anderen installierten Paketen als optionale Abhängigkeit gelistet sind).

```
root@moni-wheezy-icinga:~# aptitude install icinga
```

Bestätigen Sie die bei der Installation angebotenen Pakete. Während der Installation fragt ein Dialog verschiedene Parameter ab. Bestätigen Sie hier die Vorgaben und wählen Sie Apache2 als zu nutzenden Webserver. Anschließend werden Sie nach dem Passwort des Administratorzugangs `icingaadmin` für das klassische Web-Interface von Icinga abgefragt. Die Benutzung des hierzu für die Authentifizierung verwendeten `http-auth` werden wir in Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche* genauer erläutern.

Nach der Konfiguration läuft Icinga bereits. Sie sollen noch sicherstellen, dass beide Services bei einem Neustart der Maschine automatisch gestartet werden. Hierzu sind entsprechende Verweise auf das jeweilige Startscript in den dafür vorgesehenen Unterverzeichnissen von */etc/rc?.d* erforderlich. Mit folgenden Anweisungen können Sie sich die vorhandenen Verweise für die beiden fraglichen Dienste Apache2 und Icinga anzeigen lassen:

```
root@moni-wheezy-icinga:~# find /etc/rc?.d/ -iname "*apache2"
/etc/rc0.d/K01apache2
/etc/rc1.d/K01apache2
/etc/rc2.d/S16apache2
/etc/rc3.d/S16apache2
/etc/rc4.d/S16apache2
/etc/rc5.d/S16apache2
/etc/rc6.d/K01apache2
root@moni-wheezy-icinga:~# find /etc/rc?.d/ -iname "*icinga"
/etc/rc0.d/K01icinga
```

```
/etc/rc1.d/K01icinga
/etc/rc2.d/S17icinga
/etc/rc3.d/S17icinga
/etc/rc4.d/S17icinga
/etc/rc5.d/S17icinga
/etc/rc6.d/K01icinga
```

Die Nummer im Namen des Ordners gibt dabei den Runlevel an, und der erste Buchstabe im Namen des Verweises spezifiziert, ob der Dienst in diesem Runlevel gestartet (S für start) oder gestoppt (K für kill) werden soll.

> **insserv**: Wenn ein Service automatisch gestartet werden soll, können Sie dies mit dem Kommando insserv <service> konfigurieren.

Damit haben Sie die Icinga-Basisinstallation abgeschlossen und können über *http://127.0.0.1/icinga* beziehungsweise auch über die systemspezifische IP-Adresse auf die Oberfläche zugreifen. Sie benötigen für den Zugang das vorher für den Nutzer icinga-admin spezifizierte Passwort. Die Konfiguration des Webservers finden Sie unter */etc/apache2/conf.d/icinga.conf*. Werfen Sie an dieser Stelle mit Ihrem Webbrowser einen ersten Blick auf Ihre Icinga-Installation. Sie werden feststellen, dass Icinga bereits eine Überwachung des lokalen Systems durchführt.

> **Oberflächentest von der Kommandozeile**: Sollte der entfernte Webzugriff auf Ihr System (noch) nicht möglich sein, beispielsweise aufgrund einer Firewall, können Sie auch über *http://127.0.0.1/icinga/* auf Ihr frisch installiertes Icinga zugreifen. Wenn Sie über keine graphische Oberfläche und bzw. oder keinen lokalen Webbrowser verfügen, können Sie die Erreichbarkeit der Oberfläche durch folgende Anweisung mit dem Kommandozeilen-Werkzeug telnet testen:

```
root@moni-wheezy-icinga:~# telnet localhost 80
[...]
GET /icinga
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en"> <head>
<title>Authentication required!</title>
[...]
```

Installation der Datenbankanbindung über IDOUtils

Als nächsten Schritt beschreiben wir die Anbindung von Icinga an eine MySQL-Datenbank mit Hilfe des Addons IDOUtils (Icinga Data Out Utilities). Debian Wheezy enthält das Paket icinga-idoutils ab Version 1.7.1. Da das zu installierende Paket zwar einen Konfigurationsdialog für einen vorhandenen Datenbankserver besitzt, diesen aber nicht automatisch installiert, führen Sie gegebenenfalls zunächst eine Installation des MySQL-Servers mit folgender Anweisung durch:

```
root@moni-wheezy-icinga:~# aptitude install mysql-server
```

Während der Installation und Konfiguration werden Sie nach einem zu setzenden Administrator-Passwort für MySQL-gefragt.

MySQL-Passwörter: Wenn Sie das hier gesetzte Passwort für den `root`-Nutzer von MySQL ändern möchten, funktioniert das folgendermaßen:

```
root@moni-wheezy-icinga:~# mysqladmin -u root -p PASSWORD
```

Um das Passwort eines nicht-administrativen MySQL-Benutzers zu ändern, können Sie sich von der Kommandozeile aus als Benutzer `root` auf dem Server anmelden und dann das hier nachfolgend angeführte Kommando nutzen:

```
root@moni-wheezy-icinga:~# mysql -u root -p
mysql> use mysql;
mysql> update user set Password=DBUSERPW('NEWDBUSERPW') WHERE User='DBUSER';
mysql> exit
```

Nach der Installation wird der MySQL automatisch gestartet. Stellen Sie zunächst sicher, dass der MySQL-Dienst tatsächlich bereits läuft und für den automatischen Start konfiguriert wurde (Erläuterung hierzu siehe weiter vorne im Buch):

```
root@moni-wheezy-icinga:~# service mysql status
Server version          5.5.28-1
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/run/mysqld/mysqld.sock
Uptime:                 2 min 17 sec
root@moni-wheezy-icinga:~# find /etc/rc?.d/ -iname "*mysql"
/etc/rc0.d/K02mysql
/etc/rc1.d/K02mysql
/etc/rc2.d/S17mysql
/etc/rc3.d/S17mysql
/etc/rc4.d/S17mysql
/etc/rc5.d/S17mysql
/etc/rc6.d/K02mysql
```

Nachdem Sie sichergestellt haben, dass der Datenbankserver läuft, installieren Sie nun die IDOUtils. Bei der Installation wird die automatische Einrichtung der Icinga-Datenbank über einen Assistenten angeboten. Nutzen Sie dies. Als Datenbanktyp wählen Sie dabei passend zur vorherigen Installation `mysql`. Sie werden dann nach dem weiter vorne eingerichteten Root-Passwort für Ihren Datenbankserver und anschließend nach einem einzurichtenden Passwort für den neu angelegten MySQL-Benutzer `icinga-idoutils` gefragt.

```
root@moni-wheezy-icinga:~# aptitude install icinga-idoutils
```

Nach der Installation wird IDO2DB noch nicht gestartet. Zunächst müssen Sie Icinga anweisen, dass es die Daten an das dafür vorgesehene und mitinstallierte Modul `idomod` übergeben soll. Dazu müssen Sie im Verzeichnis */etc/icinga/modules/* die Vorlage für die Modulkonfiguration von Icinga kopieren und umbenennen. Das sogenannte Broker-Modul wird auf diese Weise automatisch eingebunden:

```
root@moni-wheezy-icinga:~# cp /usr/share/doc/icinga-idoutils/examples/idoutils.cfg-sample
/etc/icinga/modules/idoutils.cfg
```

Dann müssen Sie Icinga in der Datei */etc/default/icinga* mitteilen, dass es den *IDO2DB Agent* starten soll:

```
# start ido2db daemon (no/yes)
IDO2DB=yes
```

Jetzt aktivieren Sie die Konfiguration, indem Sie IDO2DB starten und auch einen Neustart von Icinga durchführen:

```
root@moni-wheezy-icinga:~# service ido2db start
root@moni-wheezy-icinga:~# service icinga restart
```

Damit laufen Icinga und IDOUtils und die Daten werden in die neu angelegte Datenbank geschrieben. Prüfen Sie auch hier, ob der neue Dienst IDO2DB bei einem Neustart automatisch gestartet wird (Erläuterung siehe weiter vorne in diesem Buch):

```
root@moni-wheezy-icinga:~# find /etc/rc?.d/ -iname "*ido2db"
/etc/rc0.d/K02ido2db
/etc/rc1.d/K02ido2db
/etc/rc2.d/S18ido2db
/etc/rc3.d/S18ido2db
/etc/rc4.d/S18ido2db
/etc/rc5.d/S18ido2db
/etc/rc6.d/K02ido2db
```

Nun sollten Sie abschließend auch noch prüfen, ob tatsächlich Daten in die angelegte Datenbank geschrieben werden. Hierzu können Sie testen, ob die Icinga-Datenbank kürzlich aktualisiert wurde. Lesen Sie zu diesem Zweck den Eintrag `status_update_time` aus der Tabelle `icinga_programstatus` aus:

```
root@moni-wheezy-icinga:~# mysql -u root -p
mysql> USE icinga;
mysql> SELECT status_update_time FROM icinga_programstatus;
+---------------------+
| status_update_time |
+---------------------+
| 2012-11-30 15:17:05 |
+---------------------+
1 row in set (0.00 sec)
mysql> exit
```

Damit läuft Icinga und die Daten werden über das Modul IDO2DB in die neu angelegte Datenbank geschrieben.

Installation der Weboberfläche Icinga-Web

Icinga bietet neben der klassischen, an Nagios orientierten Nutzerschnittstelle eine neu entwickelte als Icinga-Web bezeichnete Weboberfläche. Sie können Icinga-Web problemlos parallel zu Icinga-Classic betreiben. Icinga-Web verwendet als Backend die IDOUtils-Datenbank, während Icinga-Classic die aus Nagios bekannten `status.dat`/`objects.cache`-Dateien verwendet.

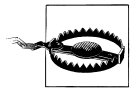
Installieren Sie zunächst einige erforderliche PHP-Pakete und anschließend Icinga-Web wie folgt :

```
root@moni-wheezy-icinga:~# aptitude install php5 php5-cli php-pear php5-xmlrpc php5-xsl
php5-gd php5-ldap php5-mysql
root@moni-wheezy-icinga:~# aptitude install icinga-web
```

In unserem Falle war hier zuvor der Apache-Webserver über das Paket `apache2-mpm-worker` installiert. Da PHP5 an dieser Stelle jedoch stattdessen die Installation des Pakets `apache2-mpm-prefork` bedingt, wird es hier gegebenenfalls entsprechend ersetzt.

Icinga-Web benötigt zusätzlich zu der IDOUtils-Datenbank eine eigene Datenbank, um Informationen aus der Weboberfläche abspeichern zu können. Bei der Installation bietet der Einrichtungs-Assistent an, Ihnen diese automatisiert anzulegen und die von Icinga-Web vorgegebene Struktur zu importieren. Sie können dies bestätigen.

Zunächst wählen Sie `mysql` als den zu nutzenden Datenbanktyp, dann geben Sie das von Ihnen gewählte Passwort für den MySQL-Benutzer `root` ein. Für die Einrichtung der `icinga_web` Datenbank fragt der Assistent nun ein Passwort für den neuen MySQL-Benutzer `icinga_web` ab. Dann erstellt und konfiguriert der Assistent die Datenbank und versieht den neuen Benutzer mit den notwendigen Zugriffsrechten. Abschließend fragt der Assistent den zu nutzenden Webserver ab. Wählen Sie hier `apache2` und geben Sie ein Passwort für den `root-Benutzer von` Icinga-Web an.



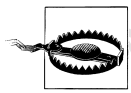
DBPASSWD: Achten Sie darauf, dass Sie die Passwörter für den `root-Benutzer` von MySQL und den Zugriff auf die IDO2DB korrekt angeben. Andernfalls wird der Installations-Assistent die Icinga-Web-Datenbank nicht korrekt anlegen und bzw. oder Icinga-Web kann auf die IDO2DB-Datenbank nicht zugreifen. Sie erhalten dann beim Aufruf von Icinga-Web eine Fehlermeldung wie beispielsweise `PDO Connection Error: SQLSTATE[28000] [1045] Access denied for user 'icinga_web'@'localhost' (using password: YES)`. Lassen Sie in diesem Fall den Konfigurations-Assistenten mit dem Kommando `dpkg-reconfigure icinga-web` noch einmal durchlaufen oder passen Sie die Daten manuell an.

Die Zugriffsdaten für beide Datenbanken werden von dem Installations-Assistenten von Debian in zwei eigenen XML-Dateien hinterlegt. Dabei enthält die Datei */etc/icinga-web/conf.d/database-web.xml* die Zugriffsdaten für die Datenbank `icinga_web` und die Datei */etc/icinga-web/conf.d/database-ido.xml* die Zugriffsdaten der für IDOUtils erstellten Datenbank `icinga`.

Wenn Sie diese Zugriffsdaten manuell ändern möchten, dann müssen Sie diese Anpassung in Datei */etc/icinga-web/conf.d/databases.xml* vornehmen und die entsprechenden `include`-Anweisungen wie folgt ausklammern:

```
[...]
<databases xmlns:db="http://agavi.org/agavi/config/parts/databases/1.0" xmlns:ae="http://
agavi.org/agavi/config/global/envelope/1.0">
```

```
 <db:database name="icinga_web" class="AppKitDoctrineDatabase">
 <!-- will be overridden by a include! -->
 <ae:parameter name="dsn">mysql://icinga_web:DBUSERPW@localhost:3306/icinga_web
 </ae:parameter>
[...]
 </db:database>
 <!-- comment this if you do not want to use debconf with dbconfig-common -->
<!--
 <xi:include xmlns:xi="http://www.w3.org/2001/XInclude" href="/etc/icinga-web/conf.d/
database-web.xml#xpointer(databases/node())">
 <xi:fallback></xi:fallback>
 </xi:include>
-->
[...]
 <db:database xmlns="http://agavi.org/agavi/config/parts/databases/1.0" name="icinga"
class="IcingaDoctrineDatabase">
 <!-- will be overridden by a include! -->
 <ae:parameter name="dsn">mysql://icinga-
idoutils:DBUSERPW@localhost:3306/icinga</ae:parameter>
[...]
 </db:database>
 <!-- comment this if you do not want to use debconf with
dbconfig-common for the IDO database -->
<!--
 <xi:include xmlns:xi="http://www.w3.org/2001/XInclude" href="/etc/icinga-web/conf.d/
database-ido.xml#xpointer(databases/node())">
 <xi:fallback></xi:fallback>
 </xi:include>
-->
[...]
</databases>
```

access denied: Konfigurationsabschnitte in der Datei *databases.xml* werden durch die Zeichenketten `<!--` und `-->` auskommentiert. Achten Sie darauf, die entsprechenden Kommentarzeichen korrekt zu setzen, da unter Umständen Icinga-Web anderenfalls nicht auf die Zugangsdaten zugreifen kann. Ein Editor mit Syntax-Highlightning, wie beispielsweise vim mit `syntax on`, hilft Ihnen bei der Identifizierung von auskommentierten Blöcken.

Nach Änderungen an den Konfigurationsdateien von Icinga-Web müssen Sie immer das Script zum Löschen der Cache-Dateien ausführen:

```
root@moni-wheezy-icinga:~# /usr/lib/icinga-web/bin/clearcache.sh
```

Damit ist die Installation der zusätzlichen Oberfläche abgeschlossen. Starten Sie jetzt den Webserver neu, damit er die hinzugekommene Konfigurationsdatei einliest:

```
root@moni-wheezy-icinga:~# service apache2 restart
```

no data: Falls Sie vor dem Problem stehen, dass Icinga-Web keine Daten anzeigt, stellen Sie sicher, dass IDOUtils und Icinga korrekt funktionieren. Weitere Tests finden Sie im Icinga Wiki: *https://wiki.icinga.org/display/testing/Icinga+Web+Testing*.

Zulassen externer Kommandos

Bei der Standard-Installation von Icinga aus den Debian-Paketen sind keine externen Befehle zugelassen. Diese werden jedoch benötigt, um beispielsweise aus der Oberfläche heraus einen Kommentar zu setzen, einen Check manuell anzustoßen oder ein Problem als bekannt zu markieren. Um externe Befehle zu aktivieren, müssen Sie den folgenden Parameter in der Icinga-Konfigurationsdatei */etc/icinga/icinga.cfg* aktivieren:

```
[...]
# EXTERNAL COMMAND OPTION
[...]
check_external_commands=1
[...]
```

Abschließend müssen Sie die Zugriffsrechte für die Kommando-Datei noch folgendermaßen anpassen, damit sie nicht beim nächsten Paketupgrade überschrieben werden:

```
root@moni-wheezy-icinga:~# service icinga stop
root@moni-wheezy-icinga:~# dpkg-statoverride --update --add nagios
www-data 2710 /var/lib/icinga/rw
root@moni-wheezy-icinga:~# dpkg-statoverride --update
--add nagios nagios 751 /var/lib/icinga
root@moni-wheezy-icinga:~# service icinga start
```

Sie sollten jetzt externe Kommandos in Icinga absetzen können.

Installation einer graphischen Auswertung mit PNP4Nagios

Sie haben dann weiter die Möglichkeit, PNP4Nagios als Addon zur Analyse und Darstellung von Performance-Daten zu installieren. Bei Wheezy ist das Paket ab Version 0.6.16 enthalten und sowohl für Nagios als auch für Icinga verfügbar. Sie können das entsprechende Paket `pnp4nagios` mittels folgender Anweisung installieren:

```
root@moni-wheezy-icinga:~# aptitude install pnp4nagios
```

Paketabhängigkeiten: Lassen Sie eventuell entstehe Konflikte zwischen Abhängigkeiten einzelner Pakete durch den Paketmanager `aptitude` auflösen. Wählen Sie im Zweifel die zuerst angebotene Lösung.

In PNP4Nagios werden die gesammelten Daten in sogenannten Round-Robin-Datenbanken (RRD) gespeichert. Die dafür nötigen Programme werden über das Paket `rrdtool` automatisch mitinstalliert.

PNP4Nagios unterstützt eine ganze Reihe von Betriebsmodi (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/modes*). Der Modus mit NPCD (Nagios Perfdata C Daemon) wird vom PNP4Nagios-Projekt mit »Dies ist aus Nagios-Sicht die sauberste Art der Verarbeitung« beschrieben. Im Folgenden beschreiben wir die Installation des sogenannten »Bulk mode mit NPCD«. Alternativ können Sie Synchronous Mode installieren, der etwas einfacher zu konfigurieren ist. Wenn Sie die Installation von Synchronous Mode bevorzugen, können Sie sich an dem entsprechenden Abschnitt im Rezept 1.10, *Nagios unter Debian installieren* orientieren.

Nehmen Sie nach der Installation als Erstes in der Konfigurationsdatei */etc/pnp4nagios/config.php* eine kleine Anpassung für Icinga vor:

```
$conf['nagios_base'] = "/icinga/cgi-bin";
```

Als nächstes müssen Sie Icinga so konfigurieren, dass zusätzlich zu den reinen Statusinformationen überhaupt eine Verarbeitung der sogenannten Performancedaten stattfindet. Dazu modifizieren Sie die folgenden Zeilen der Datei */etc/icinga/icinga.cfg*:

```
[...]
# PROCESS PERFORMANCE DATA OPTION
[...]
process_performance_data=1
[...]
# HOST AND SERVICE PERFORMANCE DATA FILES
[...]
host_perfdata_file=/var/spool/pnp4nagios/nagios/host-perfdata
service_perfdata_file=/var/spool/pnp4nagios/nagios/service-perfdata
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE TEMPLATES
[...]
host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET::$TIMET$\tHOSTNAME::
$HOSTNAME$\tHOSTPERFDATA::$HOSTPERFDATA$\tHOSTCHECKCOMMAND::
$HOSTCHECKCOMMAND$\tHOSTSTATE::$HOSTSTATE$\tHOSTSTATETYPE::
$HOSTSTATETYPE$
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::
$TIMET$\tHOSTNAME::$HOSTNAME$\tSERVICEDESC::$SERVICEDESC$\tSERVICEPERFDATA::
$SERVICEPERFDATA$\tSERVICECHECKCOMMAND::$SERVICECHECKCOMMAND$\tHOSTSTATE::
$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$\tSERVICESTATE::
$SERVICESTATE$\tSERVICESTATETYPE::$SERVICESTATETYPE$
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE MODES
[...]
host_perfdata_file_mode=a
service_perfdata_file_mode=a
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING INTERVAL
[...]
host_perfdata_file_processing_interval=30
service_perfdata_file_processing_interval=30
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING COMMANDS
[...]
host_perfdata_file_processing_command=pnp-bulknpcd-host
service_perfdata_file_processing_command=pnp-bulknpcd-service
```

Jetzt müssen Sie noch die soeben referenzierten Befehle `pnp-bulknpcd-host` und `pnp-bulknpcd-service` folgendermaßen in der Datei */etc/icinga/commands.cfg* einfügen (zu Kommandodefinitionen siehe auch Rezept 4.3, *Befehle hinzufügen (command)*):

```
define command {
       command_name                            pnp-bulknpcd-host
       command_line                            /bin/mv /var/spool/pnp4nagios/nagios/ \
host-perfdata /var/spool/pnp4nagios/npcd/host-perfdata.$TIMET$
}
```

```
define command {
       command_name                            pnp-bulknpcd-service
       command_line                            /bin/mv /var/spool/pnp4nagios/nagios/ \
service-perfdata /var/spool/pnp4nagios/npcd/service-perfdata.$TIMET$
}
```

Auf diese Weise wurde eingerichtet, dass die von Icinga erstellten Dateien in regelmäßigen Abständen in das Verzeichnis */var/spool/pnp4nagios/npcd* verschoben werden. Von dort wird sie dann der NPCD abrufen und verarbeiten. Dies ist über die folgende Konfigurationsoption in der Datei */etc/pnp4nagios/npcd.cfg* voreingestellt:

```
perfdata_spool_dir = /var/spool/pnp4nagios/npcd/
```

Legen Sie nun in der Datei */etc/default/npcd* fest, dass der NPCD Agent automatisch gestartet wird:

```
# Should NPCD be started? ("yes" to enable)
RUN="yes"
```

Nehmen Sie nun bitte die folgende Anpassung in der Datei */etc/apache2/conf.d/pnp4nagios.conf* vor, damit Sie sich unter PNP4Nagios mit den bereits vorhandenen Zugangsdaten für den Benutzer `icingaadmin` anmelden können.

```
<Directory "/usr/share/pnp4nagios/html">
[...]
        AuthUserFile /etc/icinga/htpasswd.users
```

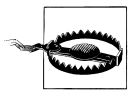
Nun starten Sie erst den NPCD und führen dann auch einen Neustart des Apache-Webservers und von Icinga durch:

```
root@moni-wheezy-icinga:~# service npcd start
root@moni-wheezy-icinga:~# service apache2 restart
root@moni-wheezy-icinga:~# service icinga restart
```

Nach der jeweils ersten Messung wird im Verzeichnis */var/lib/pnp4nagios/perfdata* ein Verzeichnis für jedes Gerät angelegt. In diesem werden für jeden Service dieses Gerätes eine RRD-Datenbank und eine entsprechende XML-Datei erstellt. Überprüfen Sie dies gegebenenfalls, indem Sie sich die Dateien im entsprechenden Ordner *localhost* anzeigen lassen:

```
root@moni-wheezy-icinga:~# ls /var/lib/pnp4nagios/perfdata/localhost/
Current_Load.rrd Current_Users.rrd Disk_Space.rrd _HOST_.rrd HTTP.rrd SSH.rrd
Current_Load.xml Current_Users.xml Disk_Space.xml _HOST_.xml HTTP.xml SSH.xml
```

Sie können PNP4Nagios unter *http://127.0.0.1/pnp4nagios*, alternativ auch unter der systemeigenen IP-Adresse, erreichen. Nutzen Sie dazu die unter Icinga gesetzten Zugangsdaten. Sobald die ersten Messungen durchgeführt sind, werden entsprechende Graphen angezeigt. Per Standardeinstellung werden nun für jedes Gerät und jeden Dienst Performancedaten verarbeitet, sofern diese vom Plugin geliefert werden.



Keine Graphen: Wenn PNP4Nagios keine Performancedaten findet, können auch keine Graphen generiert werden. Prüfen Sie bei fehlenden Graphen zunächst, ob von den entsprechenden Check-Plugins tatsächlich Performancedaten erzeugt werden.

Integration von PNP4Nagios mit Icinga-Classic

Damit Sie von der Weboberfläche Icinga-Classic aus direkt auf die Graphen zugreifen können, sollten Sie die Graphen direkt mit der Anzeige von Geräten und Diensten verzahnen. Legen Sie sich zunächst wie folgt eine Vorlage in dem dafür vorgesehenen Verzeichnis (siehe hierzu Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*) */etc/icinga/objects* an, etwa *pnptemplate.cfg* (siehe Rezept 4.1, *Maschinen einbinden (host)* für die Definition von Maschinen, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* für die Definition von Services und Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System* für die Verwendung des Vorlagensystems):

```
define host {
       name                                pnp-hst
       register                            0
       action_url                          /pnp4nagios/graph?host=$HOSTNAME$' \
class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=_HOST_
}

define service {
       name                                pnp-svc
       register                            0
       action_url                          /pnp4nagios/graph?host=$HOSTNAME$&srv= \
$SERVICEDESC$' class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=$SERVICEDESC$
}
```

Anschließend referenzieren Sie diese Definitionen dann in den vorhandenen Definitionen `generic-host` in der Datei */etc/icinga/objects/generic-host_icinga.cfg*

```
define host {
       name                                generic-host
       use                                 pnp-hst
[...]
```

und die für `generic-service` in der Datei */etc/icinga/objects/generic-service_icinga.cfg*:

```
define service {
       name                                generic-service
       use                                 pnp-svc
[...]
```

Graph-Integration: Hierbei werden entsprechende Icons für alle Geräte und Services definiert, auch für solche ohne Graphen. Alternativ besteht auch die Möglichkeit, die Vorlage `pnp-svc` nur bei den Maschinen bzw. Diensten einzubinden, für die Sie Graphen haben oder anzeigen möchten.

Die oben als Vorlage definierte Konfigurationsoption `action_url` wird jeder Maschine und jedem Dienst ein Graphen-Symbol hinzufügen. Allerdings erst nach dem nächsten Neustart von Icinga, vor dem Sie aber noch eine weitere Anpassung vornehmen sollten. Sie können sich die Graphen nämlich bereits bei einem Darüberfahren mit der Maus (`mouse-over`) anzeigen lassen. Kopieren Sie hierzu die als Vorlage gelieferte Konfigurationsdatei *status-header.ssi* in das Verzeichnis */usr/share/icinga/htdocs/ssi/* und passen Sie ihre Rechte wie folgt an:

```
root@moni-wheezy-icinga:~# cp /usr/share/doc/pnp4nagios/examples/ssi/status-header.ssi
/usr/share/icinga/htdocs/ssi/
root@moni-wheezy-icinga:~# chmod 644 /usr/share/icinga/htdocs/ssi/status-header.ssi
```

Starten Sie nun Icinga neu, damit die geänderten Dateien eingelesen werden:

```
root@moni-wheezy-icinga:~# service icinga restart
```

Damit weist nun die klassischen Oberfläche Symbole für die Graphen auf. Beim Darüberfahren mit der Maus sehen Sie eine Vorschau, durch einen Klick gelangen Sie auf die Seite von PNP4Nagios mit dem entsprechenden Graphen.

Integration von PNP4Nagios in Icinga-Web

Deutlich einfacher ist die Integration mit Icinga-Web. Hierfür benötigten Sie die Vorlagen `pnp-host-extension.xml` und `pnp-service-extension.xml`. Diese können Sie mit dem Paket `icinga-web-pnp` einspielen.

```
root@moni-wheezy-icinga:~# aptitude install icinga-web-pnp
```

Damit enthält auch die Oberfläche von Icinga-Web Symbole für die Graphen und Sie können diese abrufen.

Expander Icon: Icinga-Web zeigt die PNP-Icons seit Version 1.8 im sogenannten. `Expander Icon` an, wo sich diese dann in die Hauptansicht verschieben lassen (`Move to Grid`). Weiter ist zu beachten, dass die Icons nur dann angezeigt werden, wenn in der Konfiguration des Host/Service `process_perf_data 1` gesetzt ist.

Installation des Visalisierungsaddons NagVis

Mit Hilfe des Addons NagVis können Sie Symbole für Maschinen und Services auf einer Oberfläche beliebig anordnen. Durch die Wahl eines entsprechenden Hintergrundes erhalten die Symbole dadurch einen Mehrwert, etwa durch die Platzierung auf Fotos oder Karten. NagVis ist in Wheezy ab Version 1.6.6 enthalten, und lässt sich sowohl für Nagios als auch für Icinga als Addon installieren. Sie können das Paket `nagvis` und das dafür erforderliche Paket `php5-mysql` wie folgt installieren:

```
root@moni-wheezy-icinga:~# aptitude install php5-mysql nagvis
```

In einem Dialog wird dabei das von NagVis zu verwendende Monitoring-System abfragt. Wählen Sie hier in diesem Falle entsprechend Icinga. Bei der Installation wurde für Sie bereits ein administrativer Zugang `admin` zur Oberfläche eingerichtet, dessen Passwort `admin` Sie bei der ersten Anmeldung unbedingt sofort ändern sollten.

NagVis bietet Ihnen unterschiedliche Schnittstellen, mit denen es auf die Monitoringdaten zugreifen kann. `ndomy` war ursprünglich die Standardschnittstelle, seit Version 1.6.4-1 wurde dieses durch `mklivestatus` ersetzt. Mit `ndomy` greift NagVis über die IDOUtils-Datenbank auf die Monitoringdaten zu. Sofern Sie diese nutzen, empfehlen wir Ihnen, die `ndomy`-Schnittstelle zu verwenden. Setzen Sie dazu in der Datei */etc/nagvis/nagvis.*

ini.php den Backendtyp auf `ndomy_1` und hinterlegen Sie die Zugangsdaten zu der über IDO2DB eingebundenen Datenbank:

```
backend="ndomy_1"
[...]
dbhost="localhost"
dbport=3306
dbname="icinga"
dbuser="icinga-idoutils"
dbpass="DBUSERPW"
dbprefix="icinga_"
dbinstancename="default"
```

Sie können sich jetzt mit den oben genannten Zugangsdaten unter *http://127.0.0.1/nagvis* anmelden. Sie sollten als Erstes das Administrator-Passwort für `admin` ändern. Ansonsten ist die Basiseinrichtung von NagVis damit komplett. Für den weiteren Umgang mit PNP4Nagios und NagVis lesen Sie bitte das Kapitel zur Datenvisualisierung (Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*).

NagVis-Beispiele: Sie können sich optional mit dem Debian-Paket `nagvis-demos` diverse Anwendungsbeispiele installieren.

Diskussion

Die Ablageorte für diverse Komponenten und Konfigurationsdateien sind für Sie von Bedeutung, damit Sie schnell die richtige Stelle für gewünschte Änderungen finden können. Der Tabelle 1-8 können Sie zunächst entnehmen, welches die Speicherorte der einzelnen Komponenten der Icinga-Installation sind.

Tabelle 1-8: Ablageorte der Komponenten von Icinga

Komponente	Pfad
Programm	/usr/sbin/icinga
Haupt-Konfigurationsdatei	/etc/icinga/icinga.cfg
Plugins	/usr/lib/nagios/plugins/
Logdatei	/var/log/icinga/
Webserver-Konfiguration	/etc/apache2/conf.d/icinga.conf
Webseiten	/usr/share/icinga/htdocs/

Das Addon IDOUtils verwendete hier die in Tabelle 1-9 aufgeführten Dateien und Ordner.

Tabelle 1-9: Pfade der wichtigsten Dateien von IDOUtils

Komponente	Pfad
Konfigurationsdatei ido2db	/etc/icinga/ido2db.cfg
Konfigurationsdatei idomod	/etc/icinga/idomod.cfg
Konfigurationsdatei Event-Broker-Modul	/etc/icinga/modules/idoutils.cfg
Debug-Logs	/var/log/icinga/*.debug

Das Addon Icinga-Web verwendet die in Tabelle 1-10 aufgeführten Speicherorte.

Tabelle 1-10: Pfade zu den Komponenten von Icinga-Web

Komponente	Pfad
Haupt-Konfigurationsdateien	/etc/icinga-web/
Webserver-Konfiguration	/etc/apache2/conf.d/icinga-web.conf
Webseiten	/usr/share/icinga-web/pub/

Das Addon PNP4Nagios hat hier die in Tabelle 1-11 aufgezeigten Installationsorte verwendet.

Tabelle 1-11: Pfade zu den Komponenten von PNP4Nagios

Komponente	Pfad
Konfigurationsdateien	/etc/pnp4nagios/
Haupt-Konfigurationsdatei	/etc/pnp4nagios/config.php
NPCD-Konfigurationsdatei	/etc/pnp4nagios/npcd.cfg
Webserver-Konfiguration	/etc/apache2/conf.d/pnp4nagios.conf
Webseiten	/usr/share/pnp4nagios/html/
Perfdata-Script zur Verarbeitung	/usr/lib/pnp4nagios/libexec/process_perfdata.pl
Perfdata-Konfigurationsdatei	/etc/pnp4nagios/process_perfdata.cfg
Logdateien	/var/log/pnp4nagios/

Das Addon Nagvis hat hier die in Tabelle 1-12 aufgezeigten Installationsorte verwendet.

Tabelle 1-12: Pfade zu den Komponenten von NagVis

Komponente	Pfad
Haupt-Konfigurationsdatei	/etc/nagvis/nagvis.ini.php
Webserver-Konfiguration	/etc/apache2/conf.d/nagvis.conf
Webseiten	/usr/share/nagvis/share/

Bei der Installation von Icinga wird automatisch das Paket `nagios-plugins` installiert. Dieses enthält die Pakete `nagios-plugins-basic` und `nagios-plugins-standard`. Auf die zugehörigen Plugins werden wir in den Rezepten in den Kapiteln 5 und 6 genauer eingehen. Für die Überwachung von System-Updates können Sie das in den Nagios-Plugins enthaltene Plugin `check_apt` verwenden (siehe Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*). Außerdem stehen das Paket `nagios-snmp-plugins` sowie einige weitere spezielle Plugins zur Verfügung.

Für die Plugins wird unter */etc/nagios-plugins* ein eigenes Konfigurationsverzeichnis erstellt. Dieses enthält vorgefertigte Kommando-Definitionen (siehe hierzu Rezept 4.3, *Befehle hinzufügen (command)*), die sich für die entsprechende Einbindung unter Icinga nutzen lassen.

Als weiteres Addon können Sie NRPE (Nagios Remote Plugin Executor) installieren (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*, und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*).

Siehe auch

- Rezept mit Entscheidungshilfe zur Installation: Rezept 1.1, *Entscheidungshilfe für die Installation*
- Webseite des Icinga-Projekts: *https://www.icinga.org/*
- Webseite des Monitoring-Projekt-Repositories von Debian: *http://debmon.org/*
- Icinga-HOWTOs für die paketbasierte Installation: *https://wiki.icinga.org/display/howtos/Setting+up+Icinga+with+IDOUtils*
- Icinga Fehlersuche: *https://wiki.icinga.org/display/testing/Icinga+Web+Testing*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Webseite von NagVis: *http://www.nagvis.org/*
- Webseite des Debian-Projekts: *http://www.debian.org/*
- Rezept zur quellenbasierten Installation von Icinga: Rezept 1.4, *Icinga aus den Quellen installieren*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zum Aufbau und zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Kapitel zur Konfiguration von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere Rezept 4.3, *Befehle hinzufügen (command)*, Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* sowie Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Kapitel zur Datenvisualisierung mit PNP4Nagios und NagVis: Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*
- Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezept zur Überwachung von Systemupdates: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*

- Rezepte zur Installation von NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

1.6 Icinga unter Ubuntu installieren

Problem

Sie möchten Icinga unter Ubuntu paketbasiert installieren.

Lösung

Wir zeigen zunächst die Installation von Icinga und darauf aufbauend die des Datenbank-Addons IDOUtils (Icinga Data Out Utilities). Des Weiteren beschreiben wir die Installation des Addons PNP4Nagios zur automatischen Weiterverarbeitung und graphischen Auswertung von Performancedaten.

Die Installation von NagVis, einem Addon, das die Datenbankanbindung über die NDOUtils verwendet, ist paketbasiert unter Ubuntu zur Zeit nur mit Nagios möglich. Das in Ubuntu 12.04 (Precise Pangolin) mitgelieferte Paket war in unserer Referenzinstallation nicht funktionsfähig. Wir vermuten, dass dieser Fehler mit Erscheinen dieses Buches behoben sein wird. Daher empfehlen wir Ihnen, dass Sie sich bei der paketbasierten Installation von NagVis an dem entsprechenden Abschnitt des Installationsrezeptes für Debian orientieren. Ansonsten müssen Sie mit der Installation aus Quellen vorliebnehmen.

Installation von Icinga

Die aktuelle Ubuntu LTS-Version 12.04 Precise Pangolin enthält Icinga ab Version 1.6.1.

Icinga-PPAs: Eine noch aktuellere Version von Icinga lässt sich über das Personal Packet Archive (PPA) des Debian Packagers beziehen (*https://launchpad.net/%7Eformorer/+archive/icinga*). Aufgrund der Aktualität der Pakete sollten Sie aber auf zusätzlich notwendige Anpassungen vorbereitet sein.

`apt-get & aptitude`: Als Benutzerschnittstellen zum eigentlich Paketmanager `dpkg` stehen Ihnen sowohl `apt-get` als auch `aptitude` zur Verfügung, die funktional äquivalent sein sollten. Wir verwenden in diesem Buch aus Gründen der Einheitlichkeit stets `aptitude`.

Basierend auf einer minimalen 64Bit-Server-Installation von Precise Pangolin können Sie sich mit folgendem Befehl eine laufende Icinga-Umgebung einrichten. Dabei handelt es

sich um eine Installation von Icinga mit der klassischen, an Nagios orientierten, Oberfläche (Icinga-Classic). Die neue Oberfläche von Icinga-Web ist in Ubuntu 12.04 nicht vorgesehen, kann aber über externe PPA eingebunden werden (siehe *https://wiki.icinga.org/display/howtos/Setting+up+Icinga+Web+on+Ubuntu*):

Deinstallation: Sofern Sie eine Deinstallation erwarten und dabei sichergehen möchten, dass alle mitinstallierten Pakete wieder entfernt werden, sollten Sie sich an dieser Stelle die Liste der auf Ihrem System mitinstallierten Pakete durch Kopieren notieren. Der Hintergrund dabei ist der, dass bei der Deinstallation über Paketmanager regelmäßig ganze Pakete auf dem System verbleiben (genau dann nämlich, wenn diese in anderen installierten Paketen als optionale Abhängigkeit gelistet sind).

```
root@moni-pangolin-icinga:~# aptitude install icinga
```

Während der Installation erfolgt dabei zunächst eine Abfrage, wie das Mail-System `postfix` eingerichtet werden soll. Für Testzwecke wählen Sie im Zweifel `nur lokal` und anschließend `localdomain`.

Icinga fragt dann weiter ab, mit welchem Webserver es laufen soll. Wählen Sie hier den automatisch mit-installierten Apache2. In einem weiteren Dialog wird noch ein Passwort für den Webbenutzer `icingaadmin` abgefragt. Dieses wird in der Datei */etc/icinga/htpasswd.users* hinterlegt. Wie Sie mit dieser `http-auth`-Authentifizierung arbeiten, ist in Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche* noch genauer beschrieben. Nach Abschluss der Installation und Konfiguration laufen Icinga und Apache. Dies prüfen Sie mit folgender Anweisung:

```
root@moni-pangolin-icinga:~# service apache2 status
root@moni-pangolin-icinga:~# service icinga status
```

Es wird nun bereits eine Überwachung der lokalen Maschine durchgeführt. Nun prüfen Sie noch, ob beide Dienste nach dem Booten automatisch gestartet werden. Hierzu suchen wir in den entsprechenden Ordnern nach Links auf die betreffenden Startscripte:

```
root@moni-pangolin-icinga:~# find /etc/rc?.d/ -iname "*apache2"
/etc/rc0.d/K09apache2
/etc/rc1.d/K09apache2
/etc/rc2.d/S91apache2
/etc/rc3.d/S91apache2
/etc/rc4.d/S91apache2
/etc/rc5.d/S91apache2
/etc/rc6.d/K09apache2
root@moni-pangolin-icinga:~# find /etc/rc?.d/ -iname "*icinga"
/etc/rc0.d/K18icinga
/etc/rc1.d/K18icinga
/etc/rc2.d/S30icinga
/etc/rc3.d/S30icinga
/etc/rc4.d/S30icinga
/etc/rc5.d/S30icinga
/etc/rc6.d/K18icinga
```

Die Nummer im Namen des Ordners gibt dabei den Runlevel an, und der erste Buchstabe im Namen des Verweises spezifiziert, ob der Dienst in diesem Runlevel gestartet (`S` für `start`) oder gestoppt (`K` für `kill`) werden soll.

`insserv`: Wenn ein Service automatisch gestartet werden soll, können Sie dies mit dem Kommando `insserv <service>` konfigurieren.

Konfigurationsschritte: Die Schritte für eine manuelle, angepasste Konfiguration können Sie der quellcodebasierenden Icinga-Installation entnehmen (siehe Rezept 1.4, *Icinga aus den Quellen installieren*).

Sowohl Icinga als auch Apache werden also bei zukünftigen Boot-Vorgängen automatisch gestartet werden. Der für die Bereitstellung der Weboberfläche mitinstallierte Webserver Apache wird in */etc/apache2/conf.d/icinga.conf* so eingerichtet, dass ein Zugriff über *http://127.0.0.1/icinga* und über die eigene IP Adresse möglich ist. Der Zugang erfolgt über denn Webbenutzer `icingaadmin` unter Eingabe des soeben vergebenen Passwortes (siehe hierzu auch Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*).

Oberflächentest von der Kommandozeile: Sollte der entfernte Webzugriff auf Ihr System (noch) nicht möglich sein, beispielsweise aufgrund einer Firewall, können Sie auch über *http://127.0.0.1/icinga/* auf Ihr frisch installiertes Icinga zugreifen. Wenn Sie über keine graphische Oberfläche und bzw. oder keinen lokalen Webbrowser verfügen, können Sie die Erreichbarkeit der Oberfläche durch folgende Anweisung mit dem Kommandozeilen-Werkzeug `telnet` testen:

```
root@moni-pangolin-icinga:~# telnet localhost 80
[...]
GET /icinga
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en"> <head>
<title>Authentication required!</title>
[...]
```

Werfen Sie an dieser Stelle, wie beschrieben, einen ersten Blick auf die Weboberfläche. Sie werden feststellen, dass Icinga bereits eine Überwachung des lokalen Systems durchführt.

Installation der Datenbankanbindung über IDOUtils

Sie können nun Icinga mit den IDOUtils (Icinga Data Out Utilities) an einen Datenbankserver anbinden. Precise Pangolin enthält das Paket `icinga-idoutils` ab Version 1.6.1. Das Paket umfasst einen Konfigurationsdialog zur Einrichtung der Datenbank, installiert

allerdings nicht automatisch einen Datenbankserver. Um diesen Konfigurationsdialog für einen lokalen MySQL-Server nutzen zu können, installieren Sie deshalb vor den IDOUtils zunächst den MySQL-Server:

```
root@moni-pangolin-icinga:~# aptitude install mysql-server
```

Dabei wird das für den root-Benutzer von MySQL zu verwendende Passwort abgefragt.

MySQL-Passwörter: Wenn Sie das hier gesetzte Passwort für den root-Nutzer von MySQL ändern möchten, funktioniert das folgendermaßen:

```
root@moni-pangolin-icinga:~# mysqladmin -u root -p PASSWORD
```

Um das Passwort eines nicht-administrativen MySQL-Benutzers zu ändern, können Sie sich von der Kommandozeile aus als Benutzer root auf dem Server anmelden und dann das hier nachfolgend angeführte Kommando nutzen:

```
root@moni-pangolin-icinga:~# mysql -u root -p
mysql> use mysql;
mysql> update user set Password=DBUSERPW('NEWDBUSERPW') WHERE User='DBUSER';
mysql> exit
```

Vergewissern Sie sich, dass der MySQL-Server auch gestartet wurde, und überprüfen Sie, ob er für den automatischen Start konfiguriert wurde:

```
root@moni-pangolin-icinga:~# service mysql status
root@moni-pangolin-icinga:~# find /etc/init -iname "mysql*"
```

Anschließend installieren Sie die IDOUtils für die Anbindung an den MySQL-Server:

```
root@moni-pangolin-icinga:~# aptitude install icinga-idoutils
```

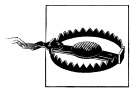
Nutzen Sie den vom Paket icinga-idoutils angebotenen und eingangs erwähnten Einrichtungs-Assistenten. Bestätigen Sie zunächst mysql als die zu nutzende Datenbank. Dann geben Sie das vorher angelegte Passwort für den MySQL-Benutzer root ein und generieren direkt im Anschluss ein neues Passwort für den MySQL-Benutzer icinga-idoutils, der für Icinga angelegt wird. Die Datenbank wird dann erstellt und konfiguriert und der Benutzer mit den entsprechenden Rechten für diese versehen.

Die IDOUtils werden dabei aber an dieser Stelle noch nicht mit Icinga verknüpft und deshalb auch nicht gestartet. Hier ist noch ein wenig Nacharbeit von Ihnen erforderlich. Weisen Sie zunächst Icinga über die Datei */etc/default/icinga* ab, dass es IDO2DB starten soll:

```
# start ido2db daemon (no/yes)
IDO2DB=yes
```

Außerdem müssen Sie im Verzeichnis */etc/icinga/modules/* die Vorlage für die Modulkonfiguration von Icinga kopieren (oder wahlweise umbenennen). Das sogenannte Broker-Modul wird damit automatisch eingebunden:

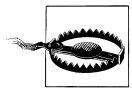
```
root@moni-pangolin-icinga:~# cp /etc/icinga/modules/idoutils.cfg-sample
/etc/icinga/modules/idoutils.cfg
```



Eventbroker-Modul: Seit IDOUtils Version 1.4 sollte das Eventbroker-Modul `idomod` über eine Modules-Definition im modules-Unterverzeichnis */etc/icinga/modules/* eingebunden werden. Sie sollten daher nicht zusätzlich den entsprechenden Eintrag `broker_modul` in der Hauptkonfigurationsdatei `/etc/icinga/icinga.cfg` aktivieren, da es in einem solchen Fall, verursacht durch die Doppeldefinition, zu schwer zu analysierenden Fehlern kommen könnte.

In der neuen Datei */etc/icinga/modules/idoutils.cfg* passen Sie dann den Pfad wie folgt an:

```
path /usr/lib/icinga/idomod.o
```



idomod: Seit den IDOUtils Version 1.7 ist `idomod.o -> idomod.so` .

Jetzt können Sie IDO2DB starten und auch einen Neustart von Icinga durchführen, damit die vorgenommenen Änderungen wirksam werden:

```
root@moni-pangolin-icinga:~# service ido2db start
root@moni-pangolin-icinga:~# service icinga restart
```

An dieser Stelle sollten Sie auch prüfen, ob IDO2DB beim Bootvorgang automatisch gestartet wird:

```
root@moni-pangolin-icinga:~# find /etc/rc?.d/ -iname "*ido2db"
/etc/rc0.d/K20ido2db
/etc/rc1.d/K20ido2db
/etc/rc2.d/S35ido2db
/etc/rc3.d/S35ido2db
/etc/rc4.d/S35ido2db
/etc/rc5.d/S35ido2db
/etc/rc6.d/K20ido2db
```

Damit ist die Icinga-Installation um die Datenbankanbindung erweitert. Vergewissern Sie sich noch abschließend, dass tatsächlich Daten in die Datenbank geschrieben werden. Dazu melden Sie sich als MySQL-Benutzer `root` am Datenbankserver an und überprüfen, ob die Icinga-Datenbank kürzlich aktualisiert wurde. Lassen Sie sich dazu den Eintrag `status_update_time` der Tabelle `icinga_programstatus` anzeigen:

```
root@moni-pangolin-icinga:~# mysql -u root -p
[...]
mysql> USE icinga;
mysql> SELECT status_update_time FROM icinga_programstatus;
+---------------------+
| status_update_time  |
+---------------------+
| 2012-11-30 20:56:21 |
+---------------------+
1 row in set (0.00 sec)
mysql> exit
```

Damit laufen sowohl Icinga als auch IDOUtils, und die Daten werden in die angelegte Datenbank geschrieben.

Installation der Weboberfläche Icinga-Web

Icinga bietet neben Icinga-Classic, der an Nagios orientierten Nutzerschnittstelle, noch eine zweite Web-Oberfläsche names Icinga-Web. Icinga-Web wurde speziell für Icinga entwickelt und nutzt als Backend die IDOUtils-Datenbank. Icinga-Classic verwendet im Gegensatz dazu die aus Nagios bekannten `status.dat/objects.cache`-Dateien. Sie können beide Oberflächen problemlos parallel betreiben.

Die neuere Oberfläche Icinga-Web ist in der aktuellen Version von Ubuntu nicht als offizielles Paket verfügbar. Sie können Icinga-Web trotzdem paketbasiert installieren, indem Sie das entsprechende PPA des Debian Packagers verwenden (*https://launchpad.net/%7Eformorer/+archive/icinga-web*). Stellen Sie sich aber dabei auf eventuelle Nacharbeiten ein. Eine aktuelle Installationshilfe finden Sie im Wiki des Icinga-Projektes (*https://wiki.icinga.org/display/howtos/Setting+up+Icinga+Web+on+Ubuntu*).

Sie können sich auch bei der Installation an den beiden Rezepten mit der Icinga-Webinstallation unter Debian orientieren (Rezept 1.5, *Icinga unter Debian installieren*).

Installation einer graphischen Auswertung mit PNP4Nagios

Sie haben jetzt die Möglichkeit, ein Addon zur Analyse und Darstellung von Performancedaten zu installieren. Precise Pangolin (Version 12.04) enthält PNP4Nagios ab Version 0.6.13 . Sie können das Paket `pnp4nagios` mit folgender Anweisung installieren:

```
root@moni-pangolin-icinga:~# aptitude install pnp4nagios
```

Paketabhängigkeiten: Lassen Sie einen eventuell entstehenden Konflikt zwischen Abhängigkeiten einzelner Pakete durch den Paketmanager `aptitude` auflösen. Wählen Sie im Zweifel die zuerst angebotene Lösung.

Gesammelte Daten werden von PNP4Nagios in sogenannten Round-Robin-Datenbanken (RRD) gespeichert. Die dafür erforderlichen Programme werden über das Paket `rrdtool` mitinstalliert.

Von PNP4Nagios wird eine ganze Reihe von Betriebsmodi unterstützt (siehe dazu *http://docs.pnp4nagios.org/de/pnp-0.6/modes*). Der Modus mit NPCD (Nagios Perfdata C Daemon) wird vom PNP4Nagios-Projekt mit »Dies ist aus Nagios-Sicht die sauberste Art der Verarbeitung« beschrieben. Deshalb beschreiben wir die Installation des sogenannten »Bulk mode mit NPCD« im Folgenden. Alternativ können Sie Synchronous Mode installieren, der etwas einfacher zu konfigurieren ist. Orientieren Sie sich für die Installation von Synchronous Mode an dem entsprechenden Abschnitt in dem Rezept 1.11, *Nagios unter Ubuntu installieren*.

Nehmen Sie jetzt zuerst einmal folgende kleine Anpassung in der Konfigurationsdatei */etc/pnp4nagios/config.php* vor:

```
$conf['nagios_base'] = "/icinga/cgi-bin";
```

Dann müssen Sie die Icinga-Konfiguration so anpassen, dass neben den reinen Statusinformationen auch die sogenannten Performancedaten verarbeitet werden. Nehmen Sie dazu folgende Modifikationen in der Datei */etc/icinga/icinga.cfg* vor:

```
[...]
# PROCESS PERFORMANCE DATA OPTION
[...]
process_performance_data=1
[...]
# HOST AND SERVICE PERFORMANCE DATA FILES
[...]
host_perfdata_file=/var/spool/pnp4nagios/nagios/host-perfdata
service_perfdata_file=/var/spool/pnp4nagios/nagios/service-perfdata
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE TEMPLATES
[...]
host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET::$TIMET$\tHOSTNAME::
$HOSTNAME$\tHOSTPERFDATA::$HOSTPERFDATA$\tHOSTCHECKCOMMAND::
$HOSTCHECKCOMMAND$\tHOSTSTATE::$HOSTSTATE$\tHOSTSTATETYPE::
$HOSTSTATETYPE$
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::
$TIMET$\tHOSTNAME::$HOSTNAME$\tSERVICEDESC::$SERVICEDESC$\tSERVICEPERFDATA::
$SERVICEPERFDATA$\tSERVICECHECKCOMMAND::$SERVICECHECKCOMMAND$\tHOSTSTATE::
$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$\tSERVICESTATE::
$SERVICESTATE$\tSERVICESTATETYPE::$SERVICESTATETYPE$
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE MODES
[...]
host_perfdata_file_mode=a
service_perfdata_file_mode=a
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING INTERVAL
[...]
host_perfdata_file_processing_interval=30
service_perfdata_file_processing_interval=30
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING COMMANDS
[...]
host_perfdata_file_processing_command=pnp-bulknpcd-host
service_perfdata_file_processing_command=pnp-bulknpcd-service
```

Anschließend fügen Sie dann die beiden neu referenzierten Befehle `pnp-bulknpcd-host` und `pnp-bulknpcd-service` in die Datei */etc/icinga/commands.cfg* ein (zu Kommandodefinitionen siehe Rezept 4.3, *Befehle hinzufügen (command)*):

```
define command {
       command_name                            pnp-bulknpcd-host
       command_line                            /bin/mv /var/spool/pnp4nagios/nagios/ \
host-perfdata /var/spool/pnp4nagios/npcd/host-perfdata.$TIMET$
}

define command {
       command_name                            pnp-bulknpcd-service
       command_line                            /bin/mv /var/spool/pnp4nagios/nagios/ \
service-perfdata /var/spool/pnp4nagios/npcd/service-perfdata.$TIMET$
}
```

Mit dieser Anweisung werden die von Icinga erstellten Dateien in regelmäßigen Abständen in das Verzeichnis */var/spool/pnp4nagios/npcd* verschoben. Von dort wird sie dann der NPCD abrufen und verarbeiten. Dies ist durch folgende Konfigurationsoption in der Datei */etc/pnp4nagios/npcd.cfg* bereits voreingestellt:

```
perfdata_spool_dir = /var/spool/pnp4nagios/npcd/
```

Sie müssen jetzt nur noch in der Datei */etc/default/npcd* festlegen, dass der NPCD Agent auch automatisch gestartet wird:

```
# Should NPCD be started? ("yes" to enable)
RUN="yes"
```

Damit Sie sich unter PNP4Nagios mit Ihren Zugangsdaten für den Benutzer `icingaadmin` authentifizieren können, nehmen Sie bitte folgende Anpassung in der Datei */etc/apache2/conf.d/pnp4nagios.conf* vor:

```
<Directory "/usr/share/pnp4nagios/html">
[...]
        AuthUserFile /etc/icinga/htpasswd.users
```

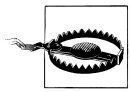
Abschließend starten Sie erst den NPCD und führen dann einen Neustart des Apache-Webservers und von Icinga durch:

```
root@moni-pangolin-icinga:~# service npcd start
root@moni-pangolin-icinga:~# service apache2 restart
root@moni-pangolin-icinga:~# service icinga restart
```

Im Verzeichnis */var/lib/pnp4nagios/perfdata* wird nun nach einigen Minuten für jedes Gerät ein Verzeichnis angelegt. In diesem Verzeichnis wird für jeden Service eine RRD-Datenbank und eine entsprechende XML-Datei erstellt. Prüfen Sie dies mit folgendem Kommando für `localhost`:

```
root@moni-pangolin-icinga:~# ls /var/lib/pnp4nagios/perfdata/localhost/
Current_Load.rrd  Current_Users.rrd  Disk_Space.rrd  _HOST_.rrd  HTTP.rrd
Current_Load.xml  Current_Users.xml  Disk_Space.xml  _HOST_.xml  HTTP.xml
```

An dieser Stelle können Sie PNP4Nagios bereits unter *http://127.0.0.1/pnp4nagios* erreichen und sich nach einiger Zeit die ersten Graphen ansehen. Nutzen Sie dabei die unter Icinga gesetzten Zugangsdaten. Per Standardeinstellung werden nun für jedes Gerät und jeden Dienst Performancedaten verarbeitet, sofern diese vom Plugin geliefert werden.



Keine Graphen: Wenn PNP4Nagios keine Performancedaten findet, können auch keine Graphen generiert werden. Prüfen Sie bei fehlenden Graphen zunächst, ob von den entsprechenden Check-Plugins tatsächlich Performancedaten erzeugt werden.

Integration von PNP4Nagios mit Icinga-Classic

Damit Sie von der Weboberfläche Icinga-Classic aus direkt auf die Graphen zugreifen können, sollten Sie die Graphen direkt mit der Anzeige von Geräten und Diensten verzahnen. Legen Sie sich zunächst wie folgt eine Vorlage in dem dafür vorgesehenen Verzeichnis

(siehe hierzu Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*) */etc/icinga/objects* an, etwa *pnptemplate.cfg* (siehe Rezept 4.1, *Maschinen einbinden (host)* für die Definition von Maschinen, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* für die Definition von Services und Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System* für die Verwendung des Vorlagensystems):

```
define host {
       name                                pnp-hst
       register                            0
       action_url                          /pnp4nagios/graph?host=$HOSTNAME$' \
class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=_HOST_
}

define service {
       name                                pnp-svc
       register                            0
       action_url                          /pnp4nagios/graph?host=$HOSTNAME$&srv= \
$SERVICEDESC$' class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=$SERVICEDESC$
}
```

Anschließend referenzieren Sie diese Definitionen dann in den vorhandenen Definitionen `generic-host` in der Datei */etc/icinga/objects/generic-host_icinga.cfg*

```
define host {
       name                                generic-host
       use                                 pnp-hst
[...]
```

und die für `generic-service` in der Datei */etc/icinga/objects/generic-service_icinga.cfg*:

```
define service {
       name                                generic-service
       use                                 pnp-svc
[...]
```

Graph-Integration: Hierbei werden entsprechende Icons für alle Geräte und Services definiert, auch für solche ohne Graphen. Alternativ haben Sie auch die Möglichkeit, die Vorlage `pnp-svc` nur bei den Maschinen bzw. Diensten einzubinden, für die Sie Graphen haben oder anzeigen möchten.

Die oben als Vorlage definierte Konfigurationsoption `action_url` wird jeder Maschine und jedem Dienst ein Graphen-Symbol hinzufügen, allerdings erst nach dem nächsten Neustart von Icinga, vor dem Sie aber noch eine weitere Anpassung vornehmen sollten. Sie können sich die Graphen nämlich bereits bei einem Darüberfahren mit der Maus (`mouse-over`) anzeigen lassen. Kopieren Sie hierzu die als Vorlage gelieferte Konfigurationsdatei *status-header.ssi* in das Verzeichnis */usr/share/icinga/htdocs/ssi/* und passen Sie ihre Rechte wie folgt an:

```
root@moni-pangolin-icinga:~# cp /usr/share/doc/pnp4nagios/examples/ssi/status-header.ssi
/usr/share/icinga/htdocs/ssi/
root@moni-pangolin-icinga:~# chmod 644 /usr/share/icinga/htdocs/ssi/status-header.ssi
```

Starten Sie nun Icinga neu, damit die geänderten Dateien eingelesen werden:

```
root@moni-pangolin-icinga:~# service icinga restart
```

Nun weist die klassische Oberfläche Symbole für die Graphen auf. Beim Darüberfahren mit der Maus sehen Sie eine Vorschau, durch Klick gelangen Sie auf die Seite von PNP4Nagios mit dem entsprechenden Graphen. Für den weiteren Umgang mit PNP4Nagios lesen Sie bitte die Rezepte zur Datenvisualisierung (Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*).

Installation des Visalisierungsaddons NagVis

Sie können mit Hilfe des Addons NagVis Symbole für Maschinen und Services beliebig auf einem Hintergrund anordnen. Durch die Wahl eines entsprechenden Hintergrundes erhalten die Symbole dadurch einen Mehrwert, etwa durch die Platzierung auf Fotos oder Karten. Precise Pangolin (Version 12.04) enthält NagVis zwar in der bereits veralteten Version ab 1.4.6, weist aber feste Paketabhängigkeiten mit den Nagios-Paketen auf und lässt sich somit erst einmal nur mit Nagios installieren.

Sollte der Fehler bis zum Erscheinen dieses Buches behoben sein, empfehlen wir Ihnen, dass Sie sich bei der paketbasierten Installation von NagVis an den entsprechenden Abschnitten des Installationsrezepte für Icinga unter Debian (siehe Rezept 1.5, *Icinga unter Debian installieren*) orientieren. Ansonsten müssen Sie mit der Installation aus Quellen vorliebnehmen, können dann aber dafür auf die aktuelle NagVis-Version zurückgreifen.

Diskussion

Die Ablageorte für diverse Komponenten und Konfigurationsdateien sind für Sie von Bedeutung, damit Sie schnell die richtige Stelle für gewünschte Änderungen finden können. Der Tabelle 1-13 können Sie zunächst entnehmen, an welchen Speicherorten die einzelnen Komponenten der Icinga-Installation hier abgelegt wurden.

Tabelle 1-13: Installationsorte wichtiger Komponenten von Icinga

Komponente	Pfad
Programm	/usr/sbin/icinga
Haupt-Konfigurationsdatei	/etc/icinga/icinga.cfg
Plugins	/usr/lib/nagios/plugins
Logdatei	/var/log/icinga/
Webserver-Konfiguration	/etc/apache2/conf.d/icinga.conf
Webseiten	/usr/share/icinga/htdocs/

Das Addon IDOUtils verwendet die in Tabelle 1-14 aufgeführten Dateien und Ordner:

Tabelle 1-14: Ablage wichtiger Dateien von IDOUtils

Komponente	Pfad
Konfigurationsdatei ido2db	/etc/icinga/ido2db.cfg
Konfigurationsdatei idomod	/etc/icinga/idomod.cfg
Konfigurationsdatei Event-Broker-Modul	/etc/icinga/modules/idoutils.cfg
Debug-Logs	/var/log/icinga/*.debug

Das Addon PNP4Nagios hat hier die in Tabelle 1-15 aufgezeigten Installationsorte verwendet.

Tabelle 1-15: Pfade zu den Komponenten von PNP4Nagios

Komponente	Pfad
Konfigurationsdateien	/etc/pnp4nagios/
Haupt-Konfigurationsdatei	/etc/pnp4nagios/config.php
NPCD-Konfigurationsdatei	/etc/pnp4nagios/npcd.cfg
Webserver-Konfiguration	/etc/apache2/conf.d/pnp4nagios.conf
Webseiten	/usr/share/pnp4nagios/html/
Perfdata-Script zur Verarbeitung	/usr/lib/pnp4nagios/libexec/process_perfdata.pl
Perfdata-Konfigurationsdatei	/etc/pnp4nagios/process_perfdata.cfg
Logdateien	/var/log/pnp4nagios/

Über die Paket-Abhängigkeiten werden hier die Plugins aus dem Paket `nagios-plugins` installiert, das die Pakete `nagios-plugins-basic` und `nagios-plugins-standard` umfasst. Auf die enthaltenen Plugins werden wir in den Rezepten der Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen* und Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen* genauer eingehen. Für die Überwachung von System-Updates können Sie das in den Nagios-Plugins enthaltene Plugin `check_apt` verwenden (siehe Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*). Als weitere Pakete mit Plugins stehen Ihnen `nagios-snmp-plugins`, `nagios-plugins-extra`, `gosa-plugin-nagios`, `uwsgi-plugin-nagios` sowie diverse `check-mk-*` zur Verfügung.

Unter Ubuntu wird für die Plugins außerdem ein eigenes Konfigurationsverzeichnis unter */etc/nagios-plugins* erstellt, das vorgefertigte Kommando-Definitionen für die Einbindung in Nagios zur Verfügung stellt (zu Kommando-Definitionen siehe Rezept 4.3, *Befehle hinzufügen (command)*).

Als Erweiterungen für Icinga sind in der Ubuntu-Version neben `pnp4nagios` und `nagvis` auch noch die Pakete `nagiosgrapher`, `nagstamon`, `nagircbot` verfügbar. Damit sind hier über das Paketmanagement mehrere Lösungen für eine graphische Aufbereitung sowie für den Anschluss an einen IRC-Server verfügbar, auf die wir hier aber nicht weiter eingehen werden. Als weiteres Addon können Sie NRPE (Nagios Remote Plugin Executor)

installieren (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*).

Siehe auch

- Rezept mit Entscheidungshilfe zur Installation: Rezept 1.1, *Entscheidungshilfe für die Installation*
- Webseite des Icinga-Projekts: *https://www.icinga.org/*
- PPA des Debian Packagers von Icinga: *https://launchpad.net/%7Eformorer/+archive/icinga*
- Icinga-HOWTOs für die paketbasierte Installation: *https://wiki.icinga.org/display/howtos/Setting+up+Icinga+with+IDOUtils*
- Installationshilfe für Icinga-Web unter Ubuntu: *https://wiki.icinga.org/display/howtos/Setting+up+Icinga+Web+on+Ubuntu*
- Icinga-Fehlersuche: *https://wiki.icinga.org/display/testing/Icinga+Web+Testing*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Webseite von Ubuntu: *http://www.ubuntu.com/*
- Rezept zur quellenbasierten Installation von Icinga: Rezept 1.4, *Icinga aus den Quellen installieren*
- Rezept zur paketbasierten Installation von Icinga unter Debian: Rezept 1.5, *Icinga unter Debian installieren*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zum Aufbau und zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Kapitel zur Konfiguration von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere das Rezept 4.3, *Befehle hinzufügen (command)*, Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* und Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Kapitel zur Datenvisualisierung mit PNP4Nagios und NagVis: Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*
- Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*

- Rezept zur Überwachung von System-Updates: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*
- Rezepte zur Installation von NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

1.7 Icinga unter SLES|openSUSE installieren

Problem

Sie möchten Icinga paketbasiert unter SuSE Linux Enterprise Server (SLES) beziehungsweise openSUSE installieren.

Lösung

Wir beschreiben im Folgenden zunächst die Installation von Icinga selbst unter der derzeit noch aktuellen openSUSE Version 12.2, dieInstallation unter SLES verläuft weitgehend analog. Darauf aufbauend können Sie in weiteren Schritten das Addon IDOUtils (Icinga Data Out Utilities) installieren, um die Daten für weitere Addons und Auswertungen in einer Datenbank vorzuhalten. Anschließend zeigen wir Ihnen anhand eines Beispiels die Installation von PNP4Nagios, wozu Sie allerdings weitere Paketquellen hinzufügen müssen. Alternativ müssten Sie mit der Installation aus Quellen vorlieb nehmen (siehe Rezept 1.4, *Icinga aus den Quellen installieren*) oder zu einer anderen Distribution wechseln (siehe Rezept 1.1, *Entscheidungshilfe für die Installation*).

Installation von Icinga

In den Paketquellen der derzeit aktuellen openSUSE-Version 12.2 sind Icinga und die IDOUtils in Version ab 1.7.4 enthalten. Für PNP4Nagios und NagVis sind unter der Standardinstallation von openSUSE 12.2 keine Pakete vorhanden. Sie müssen dazu die Paketquelle `server:monitoring` hinzufügen. Anschließend stehen Ihnen aktuelle Versionen von Icinga und PNP4Nagios zur Verfügung. Wenn Sie wissen, dass Sie diese Addons ebenfalls installieren werden, sollten Sie die Paketquelle auch schon vor der Installation von Icinga hinzufügen. Dies gilt auch, wenn Sie Icinga unter SLES installieren möchten.

YAST versus Zypper: Unter openSUSE steht neben dem systemeigenen graphischen Konfigurationstool YAST auch Zypper als Paketmanager zur Verfügung. Beide Tools greifen auf dieselbe Datenbasis zu und können somit alternativ genutzt werden. Im Gegensatz zu Yast deinstalliert Zypper automatisch auch abhängige Pakete, weshalb wir hier mit Zypper arbeiten.

Repositories: Sie können Repositories unter openSUSE sowohl mit YAST als auch mit Zypper hinzufügen. Die openSUSE-Repositories `server:monitoring` und `server:php:applications` können Sie mit Hilfe von Zypper wie nachfolgend gezeigt hinzufügen. Dabei müssen Sie den Namen der openSUSE-Distribution gegebenenfalls anpassen:

```
moni-suse122-icinga:~ # zypper ar
http://download.opensuse.org/repositories/server:/monitoring/
openSUSE_12.2/ server-mon
moni-suse122-icinga:~ # zypper ar
http://download.opensuse.org/repositories/server:/php:/applications/
openSUSE_12.2/ server-php-app
```

Basierend auf einer minimalen openSUSE-64bit-Server-Installation können Sie unter Nutzung des Paketmanagers Zypper mit der im Nachfolgenden angeführten Anweisung eine Icinga-Umgebung installieren. Dabei handelt es sich um eine Installation von Icinga mit der klassischen, an Nagios orientierten, Oberfläche. Die neuere Oberfläche von Icinga-Web ist in openSUSE 12.2 nicht vorgesehen, dazu müssten Sie ebenfalls das openSUSE-Repository `server:monitoring` hinzufügen:

```
moni-suse122-icinga:~ # zypper install icinga
```

Deinstallation: Sofern Sie eine Deinstallation erwarten und dabei sichergehen möchten, dass alle mitinstallierten Pakete wieder entfernt werden, sollten Sie sich an dieser Stelle die Liste der auf Ihrem System mit-installierten Pakete kopieren. Der Hintergrund dabei ist, dass bei der Deinstallation über Paketmanager regelmäßig ganze Pakete auf dem System verbleiben (dann nämlich, wenn diese in anderen installierten Paketen als optionale Abhängigkeit gelistet sind).

Die Abhängigkeiten werden aufgelöst und erforderliche Pakete automatisch mit-installiert. Sie müssen dann zunächst ein Passwort für den in der Konfiguration des Webservers vorgesehenen Schutz mittels `http-auth` (siehe hierzu Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*) einrichten:

```
moni-suse122-icinga:~ # htpasswd2 -c /etc/icinga/htpasswd.users icingaadmin
```

Nach der Installation starten weder der Apache2-Webserver noch Icinga automatisch. Mit folgenden Kommandos veranlassen und überprüfen Sie, dass die beide Dienste bei jedem Bootvorgang mit gestartet werden.

```
moni-suse122-icinga:~ # systemctl enable apache2.service
moni-suse122-icinga:~ # chkconfig -a icinga
[...]
icinga 0:off 1:off 2:off 3:on 4:off 5:on 6:off
```

Wenn Sie die Maschine aktuell nicht für diesen Zweck herunter und neu hochfahren möchten, starten Sie den Apache-Webserver und Icinga manuell:

```
moni-suse122-icinga:~ # systemctl start apache2.service
moni-suse122-icinga:~ # systemctl start icinga.service
```

chkconfig: Wenn ein Service automatisch gestartet werden soll, können Sie dies mit dem Kommando `chkconfig <service> on|off` konfigurieren.

Icinga ist nun mit dem Benutzernamen `icingaadmin` und dem oben gesetzten Passwort über *http://127.0.0.1/icinga/* oder die systemspezifische IP-Adresse erreichbar und führt bereits eine Überwachung des lokalen Systems durch. Die Konfiguration des Webservers finden Sie unter */etc/apache2/conf.d/icinga.conf*.

Konfigurationsschritte: Die adäquaten Schritte für eine manuelle, angepasste Konfiguration können Sie der quellcodebasierenden Icinga-Installation entnehmen (siehe Rezept 1.4, *Icinga aus den Quellen installieren*).

Nun besteht eine gute Gelegenheit, dass Sie mit einem Webbrowser einen ersten Blick auf die Oberfläche werfen. Stellen Sie bei einem Netzwerkzugriff vorher sicher, dass die lokale SUSE-Firewall die für den Apache-Webserver notwendigen Ports (80, 443) geöffnet hat. Die Firewall-Konfiguration können Sie durch das SUSE-Tool `yast` anpassen lassen. Sie müssen dort den Zugriff auf die Dienste `HTTP Server` und `HTTPS Server` freischalten.

Oberflächentest von der Kommandozeile: Falls der entfernte Webzugriff auf Ihr System (noch) nicht möglich ist, beispielsweise aufgrund einer Firewall, können Sie auch über *http://127.0.0.1/icinga/* auf Ihr frisch installiertes Icinga zugreifen. Wenn Sie über keine graphische Oberfläche und bzw. oder keinen lokalen Webbrowser verfügen, können Sie die Erreichbarkeit der Oberfläche durch folgende Anweisung mit dem Kommandozeilen-Werkzeug `telnet` testen:

```
moni-suse-icinga:~ # telnet localhost 80
[...]
GET /icinga
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en"> <head>
<title>Authentication required!</title>
[...]
```

Installation der Datenbankanbindung über IDOUtils

Zunächst benötigen Sie einen Datenbankserver, an den die IDOUtils (Icinga Data Out Utilities) dann die Daten senden sollen. OpenSUSE 12.2 führt als dazu empfohlenes Paket die Datenbank MariaDB. Aus Gründen der Einheitlichkeit arbeiten wir in diesem Buch stattdessen mit dem ebenso verfügbaren MySQL. Den MySQL-Server installieren Sie durch folgende Anweisung:

```
moni-suse122-icinga:~ # zypper install mysql-community-server
```

Der MySQL-Server läuft nach der Installation noch nicht und ist noch nicht per Passwort geschützt. Mittels folgender Kommandos teilen Sie dem System den Datenbankserver zunächst als Dienst mit und starten ihn dann manuell:

```
moni-suse122-icinga:~ # chkconfig mysql on
moni-suse122-icinga:~ # systemctl start mysql.service
```

Setzen Sie anschließend das Passwort für den MySQL-Benutzer `root` wie folgt:

```
moni-suse122-icinga:~ # mysqladmin -u root PASSWORD
```

MySQL-Passwörter: Wenn Sie das hier gesetzte Passwort für den MySQL-Nutzer `root` ändern wollen, funktioniert das folgendermaßen:

```
moni-suse-icinga:~ # mysqladmin -u root -p PASSWORD
```

Um das Passwort eines nicht-administrativen MySQL-Benutzers zu ändern, können Sie sich von der Kommandozeile aus als Benutzer `root` auf dem Server anmelden und dann das hier nachfolgend angeführte Kommando nutzen:

```
moni-suse-icinga:~ # mysql -u root -p
mysql> use mysql;
mysql> update user set Password=DBUSERPW('NEWDBUSERPW') WHERE User='DBUSER';
mysql> exit
```

Anschließend installieren Sie die IDOUtils:

```
moni-suse122-icinga:~ # zypper install icinga-idoutils-mysql
```

Nun müssen Sie noch eine zu nutzende Datenbank einrichten. Melden Sie sich dafür als MySQL-Benutzer `root` an dem Datenbankserver an und richten die Datenbank `icinga` und den Benutzer `sql_icinga` mit entsprechenden Rechten sowie einem Passwort (hier: `DBPASSWD`) ein:

```
moni-suse122-icinga:~ # mysql -u root -p
mysql> CREATE DATABASE icinga;
mysql> GRANT USAGE ON icinga.* TO 'sql_icinga'@'localhost' IDENTIFIED BY 'DBUSERPW' WITH
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0;
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, DROP, CREATE VIEW, INDEX, EXECUTE ON
icinga.* TO 'sql_icinga'@'localhost';
mysql> FLUSH PRIVILEGES;
mysql> exit
```

Außerdem müssen Sie die von den IDOUtils vorgegebene Datenbank-Struktur importieren (hier unter Nutzung des MySQL-Benutzers `root`):

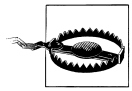
```
moni-suse122-icinga:~ # mysql -u root -p icinga < /usr/share/doc/packages/icinga-
idoutils-mysql/mysql/mysql.sql
```

Die Zugangsdaten für Datenbankzugriffe müssen in der Datei */etc/icinga/ido2db.cfg* hinterlegt sein. Sie werden zumindest die Variable `db_pass` entsprechend dem weiter oben gesetzten Passwort anpassen müssen (hier darüber hinaus den unter `db_user` angegebenen Benutzer):

```
db_user=sql_icinga
db_pass=DBUSERPW
```

Das sogenannte Broker-Modul ist über den Ordner */etc/icinga/modules/* automatisch eingebunden. Sie müssen die IDOUtils nun noch dem System als Dienst mitteilen:

```
moni-suse122-icinga:~ # chkconfig ido2db on
```



Eventbroker-Modul: Seit IDOUtils Version 1.4 sollte das Eventbroker-Modul `idomod` über eine Modules-Definition im modules-Unterverzeichnis */etc/icinga/modules/* eingebunden werden. Sie sollten daher nicht zusätzlich den entsprechenden Eintrag `broker_modul` in der Hauptkonfigurationsdatei `/etc/icinga/icinga.cfg` aktivieren, da es in einem solchen Fall, verursacht durch die Doppeldefinition, zu schwer zu analysierenden Fehlern kommen könnte.

An dieser Stelle können Sie mit folgenden Befehlen überprüfen, ob alle notwendigen Dienste für den automatischen Start konfiguriert sind:

```
moni-suse122-icinga:~ # systemctl is-enabled apache2.service
enabled
moni-suse122-icinga:~ # chkconfig icinga
icinga  on
moni-suse122-icinga:~ # chkconfig mysql
mysql  on
moni-suse122-icinga:~ # chkconfig ido2db
ido2db  on
```

Nun können Sie das konfigurierte System neu starten, damit die Änderungen effektiv werden:

```
moni-suse122-icinga:~ # systemctl reboot
```

Nach dem Neustart laufen die IDOUtils und sind eingerichtet, so dass Daten in die angelegte Datenbank geschrieben werden. Testen können Sie dies beispielsweise, indem Sie den Wert `status_update_time` der Tabelle `icinga_programstatus` in Ihrer Datenbank abfragen:

```
moni-suse122-icinga:~ # mysql -u root -p
mysql> USE icinga;
mysql> SELECT status_update_time FROM icinga_programstatus;
+---------------------+
| status_update_time  |
+---------------------+
| 2013-03-16 11:05:10 |
+---------------------+
1 row in set (0.00 sec)

mysql> exit
```

Nun laufen sowohl Icinga als auch IDOUtils und die Daten werden in die angelegte Datenbank geschrieben.

Installation der Weboberfläche Icinga-Web

Icinga bietet in neben der an Nagios orientierte Nutzerschnittstelle Icinga-Classic noch ein zweite, speziell für Icinga entwickelte Oberfläche. Sie können beide Weboberflächen problemlos parallel betreiben. Icinga-Web greift dabei auf die IDOUtils-Datenbank zu, Icinga-Classic verwendet im Gegensatz dazu die aus Nagios bekannten `status.dat`/`objects.cache`-Dateien.

Installieren Sie nun zunächst einige erforderliche PHP-Pakete und anschließend Icinga-Web wie folgt:

```
moni-suse122-icinga:~ # zypper install php5 php5-cli php-pear php5-xmlrpc php5-xsl
php5-gd php5-ldap php5-mysql
moni-suse122-icinga:~ # aptitude install icinga-web
```

Icinga-Web ist darauf ausgelegt, Daten in einer eigenen Datenbank zu speichern. Sie benötigen diese Datenbank zusätzlich zu der bereits für die IDOUtils installierten. Legen Sie die neue Datenbank zunächst unter Nutzung des MySQL-Benutzers `root` an und richten Sie dabei außerdem einen entsprechenden MySQL-Benutzer `sql_icinga-web` für den Zugriff ein (verwenden Sie anstelle des hier genutzten Passworts `DBUSERPW` bitte wieder ein eigenes).

```
moni-suse122-icinga:~ # mysql -u root -p
[...]
mysql> CREATE DATABASE icinga_web;
mysql> GRANT USAGE ON icinga_web.* TO 'sql_icinga-
web'@'localhost' IDENTIFIED BY 'DBUSERPW' WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER
_HOUR 0 MAX_UPDATES_PER_HOUR 0;
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, INDEX ON icinga_web.*
TO 'sql_icinga-web'@'localhost';
mysql> FLUSH PRIVILEGES;
mysql> EXIT;
```

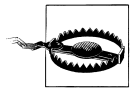
Sie können jetzt die vorgegebene Struktur in die noch leere Datenbank importieren (hier mit dem vorher angelegten MySQL-Benutzer `sql_icinga-web`):

```
moni-suse122-icinga:~ # mysql -u sql_icinga-web -p icinga_web <
/usr/share/icinga-web/etc/schema/mysql.sql
```

Anschließend müssen Sie die Zugriffsdaten für diese Datenbank in der Datei */etc/icinga-web/databases.xml* hinterlegen. Sie benötigen zunächst die Zugangsdaten (Benutzername und Passwort) für die Datenbank `icinga_web` und außerdem die für die Datenbank `icinga`, die für IDOUtils erstellt wurde.

```
[...]
    <db:database name="icinga_web" class="AppKitDoctrineDatabase">
 <ae:parameter name="dsn">mysql://sql_icinga-web:
DBUSERPW@localhost:3306/icinga_web</ae:parameter>
[...]
  </ae:parameter>
    </db:database>
[...]
```

```
    <db:database xmlns="http://agavi.org/agavi/config/parts/databases/1.0" name="icinga" c
lass="IcingaDoctrineDatabase">
 <ae:parameter name="dsn">mysql://sql_icinga:DBUSERWD@localhost:3306/icinga</ae:parameter>
[...]
  </ae:parameter>
    </db:database>
[...]
```



Zugriff verweigert: In openSUSE sind die betroffenen Konfigurationsabschnitte in der Datei */etc/icinga-web/databases.xml* durch die Zeichenketten `<!--` und `-->` auskommentiert. Achten Sie darauf, die entsprechenden Kommentarzeichen zu entfernen, da anderenfalls Icinga-Web nicht auf die Zugangsdaten zugreifen kann. Ein Editor mit Syntax-Highlightning, wie beispielsweise `vim` mit `syntax on`, hilft Ihnen bei der Identifizierung von auskommentierten Blöcken.

Nach Änderungen an den Konfigurationsdateien von Icinga-Web müssen Sie immer das Script zum Löschen der Cache-Dateien ausführen:

```
moni-suse122-icinga:~ # /usr/sbin/icinga-web-clearcache
```

Damit ist die Installation der zusätzlichen Oberfläche abgeschlossen. Starten Sie jetzt den Webserver jetzt neu, damit er die hinzugekommene Konfigurationsdatei einliest:

```
moni-suse122-icinga:~ # service apache2 restart
```

Sie können sich die neue Oberfläche nun unter *http://127.0.0.1/icinga-web* anschauen (siehe dazu Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*). Loggen Sie sich mit dem Nutzernamen `root` und dem Passwort `password` erstmalig ein. Als erstes sollten Sie dann dieses Passwort ändern. Das entsprechende Menü finden Sie rechts oben.

`no data`: Falls Sie vor dem Problem stehen, dass Icinga-Web keine Daten anzeigt, stellen Sie sicher, dass IDOUtils und Icinga korrekt funktionieren. Weitere Tests finden Sie im Icinga Wiki: *https://wiki.icinga.org/display/testing/Icinga+Web+Testing*.

Installation einer graphischen Auswertung mit PNP4Nagios

Bei PNP4Nagios handelt es sich um ein Addon zur Analyse und Darstellung von Performancedaten. Um PNP4Nagios zu installieren, benötigen Sie das openSUSE-`Repository server:monitoring`, in dem eine aktuelle Version von PNP4Nagios enthalten ist. Da dieses Paket für Nagios gedacht ist, müssen Sie zuerst einen Nutzer und eine Gruppe `nagios` erstellen und entsprechende Rechte zuweisen:

```
moni-suse122-icinga:~ # useradd -m nagios
moni-suse122-icinga:~ # passwd nagios
moni-suse122-icinga:~ # groupadd nagios
moni-suse122-icinga:~ # usermod -G nagios nagios
moni-suse122-icinga:~ # usermod -G nagios icinga
moni-suse122-icinga:~ # usermod -g icinga nagios
moni-suse122-icinga:~ # usermod -g icingacmd nagios
```

Sie können das entsprechende Paket `pnp4nagios` mit folgender Anweisung installieren:

```
moni-suse122-icinga:~ # zypper install pnp4nagios
```

In PNP4Nagios werden die gesammelten Daten in Round-Robin-Datenbanken (RRD) gespeichert. Die erforderlichen Programme werden über das Paket `rrdtool` automatisch mitinstalliert.

Bei PNP4Nagios werden eine ganze Reihe von Betriebsmodi unterstützt (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/modes*). Der Modus mit NPCD (Nagios Perfdata C Daemon) wird vom PNP4Nagios-Projekt mit »Dies ist aus Nagios-Sicht die sauberste Art der Verarbeitung« beschrieben. Im Folgenden werden wir die Installation des sogenannten »Bulk mode mit NPCD« erläutern. Alternativ können Sie Synchronous Mode installieren, der etwas einfacher zu konfigurieren ist. Orientieren Sie sich für die Installation von Synchronous Mode an dem entsprechenden Abschnitt im Rezept 1.12, *Nagios unter SLES|openSUSE installieren*.

Zunächst nehmen Sie nach der Installation in der Konfigurationsdatei */etc/pnp4nagios/config.php* folgende kleine Anpassung vor:

```
$conf['nagios_base'] = "/icinga/cgi-bin";
```

Anschließend müssen Sie Icinga so konfigurieren, dass zusätzlich zu den reinen Statusinformationen eine Verarbeitung der sogenannten Performancedaten stattfindet. Dazu modifizieren Sie die folgenden Zeilen der Datei */etc/icinga/icinga.cfg*:

```
[...]
# PROCESS PERFORMANCE DATA OPTION
[...]
process_performance_data=1
[...]
# HOST AND SERVICE PERFORMANCE DATA FILES
[...]
host_perfdata_file=/var/spool/pnp4nagios/nagios/host-perfdata
service_perfdata_file=/var/spool/pnp4nagios/nagios/service-perfdata
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE TEMPLATES
[...]
host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET::$TIMET$\tHOSTNAME::
$HOSTNAME$\tHOSTPERFDATA::$HOSTPERFDATA$\tHOSTCHECKCOMMAND::
$HOSTCHECKCOMMAND$\tHOSTSTATE::$HOSTSTATE$\tHOSTSTATETYPE::
$HOSTSTATETYPE$
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::
$TIMET$\tHOSTNAME::$HOSTNAME$\tSERVICEDESC::$SERVICEDESC$\tSERVICEPERFDATA::
$SERVICEPERFDATA$\tSERVICECHECKCOMMAND::$SERVICECHECKCOMMAND$\tHOSTSTATE::
$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$\tSERVICESTATE::
$SERVICESTATE$\tSERVICESTATETYPE::$SERVICESTATETYPE$
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE MODES
[...]
host_perfdata_file_mode=a
service_perfdata_file_mode=a
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING INTERVAL
[...]
```

```
host_perfdata_file_processing_interval=30
service_perfdata_file_processing_interval=30
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING COMMANDS
[...]
host_perfdata_file_processing_command=pnp-bulknpcd-host
service_perfdata_file_processing_command=pnp-bulknpcd-service
```

Die soeben referenzierten Befehle `pnp-bulknpcd-host` und `pnp-bulknpcd-service` müssen Sie jetzt wie folgt in die Datei */etc/icinga/objects/commands.cfg* einfügen (zu Kommandodefinitionen siehe auch Rezept 4.3, *Befehle hinzufügen (command)*):

```
define command {
       command_name                            pnp-bulknpcd-host
       command_line                            /bin/mv /var/spool/pnp4nagios/nagios/ \
host-perfdata/var/spool/pnp4nagios/npcd/host-perfdata.$TIMET$
}

define command {
       command_name                            pnp-bulknpcd-service
       command_line                            /bin/mv /var/spool/pnp4nagios/nagios/ \
service-perfdata/var/spool/pnp4nagios/npcd/service-perfdata.$TIMET$
}
```

Die von Icinga erstellten Performancedaten werden nun in regelmäßigen Abständen in das Verzeichnis */var/spool/pnp4nagios/npcd* verschoben. Von dort wird sie dann der NPCD Agent abrufen und verarbeiten. Das Quellverzeichnis muss dann ebenfalls in der Datei */etc/pnp4nagios/npcd.cfg* angepasst werden:

```
perfdata_spool_dir = /var/spool/pnp4nagios/npcd
```

Außerdem müssen Sie noch beide Verzeichnisse erstellen und die Rechte für den Nutzer `nagios` entsprechend anpassen.

```
moni-suse122-icinga:~ # mkdir -p /var/spool/pnp4nagios/npcd
moni-suse122-icinga:~ # mkdir -p /var/spool/pnp4nagios/nagios
moni-suse122-icinga:~ # chown nagios:nagios /var/spool/pnp4nagios/npcd
moni-suse122-icinga:~ # chown nagios:nagios /var/spool/pnp4nagios/nagios
moni-suse122-icinga:~ # chmod g+w /var/spool/pnp4nagios/nagios
moni-suse122-icinga:~ # chmod g+w /var/spool/pnp4nagios/npcd
```

Anschließend passen Sie noch das Startscript `/etc/init.d/npcd` wie folgt an:

```
# Required-Start: $local_fs $remote_fs icinga
# Required-Stop: $local_fs $remote_fs icinga
```

Dann legen Sie fest, dass der NPCD Agent automatisch gestartet wird.

```
moni-suse122-icinga:~ # systemctl enable npcd.service
moni-suse122-icinga:~ # chkconfig -a npcd
[...]
nscd 0:off 1:off 2:off 3:on 4:off 5:on 6:off
```

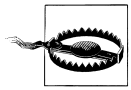
Im Anschluss daran nehmen Sie bitte die folgende Anpassung in der Datei */etc/apache2/conf.d/nagios-pnp.conf* vor, damit Sie sich unter PNP4Nagios mit den bereits vorhandenen Zugangsdaten für den Benutzer `icingaadmin` anmelden können.

```
<Directory "/usr/share/pnp4nagios/html">
[...]
        AuthUserFile /etc/icinga/htpasswd.users
```

Nun starten Sie den Apache-Webserver und Icinga neu:

```
moni-suse122-icinga:~ # service npcd start
moni-suse122-icinga:~ # service apache2 restart
moni-suse122-icinga:~ # service icinga restart
```

Rufen Sie dann zunächst die PNP4Nagios-Oberfläche unter *http://127.0.0.1/pnp4nagios* beziehungsweise der jeweiligen IP-Adresse auf, die nun die ersten Graphen enthält. Nutzen Sie dabei die unter Icinga gesetzten Zugangsdaten. Sie haben damit die Installation von PNP4Nagios abgeschlossen. Es werden jetzt für jedes Gerät und für jeden Dienst Performancedaten verarbeitet, sofern diese von dem Plugin geliefert werden.



Keine Graphen: Wenn PNP4Nagios keine Performancedaten findet, können auch keine Graphen generiert werden. Prüfen Sie bei fehlenden Graphen zunächst, ob von den entsprechenden Check-Plugins tatsächlich Performancedaten erzeugt werden.

Integration von PNP4Nagios in Icinga-Classic

Für den Zugriff auf die von PNP4Nagios erstellten Graphen aus der Weboberfläche Icinga-Classic müssen Sie noch einige Anpassungen vornehmen. Legen sich Sie sich dazu in der Konfigurationsdatei */etc/icinga/objects/templates.cfg* die Vorlagen `pnp-host` und `pnp-srv` an (siehe hierzu Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*, Rezept 4.1, *Maschinen einbinden (host)* für die Definition von Maschinen, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* für die Definition von Services und Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System* für die Verwendung des Vorlagensystems):

```
define host {
       name                                pnp-hst
       register                            0
       action_url                          /pnp/index.php/graph?host=$HOSTNAME$' \
class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=_HOST_
}

define service {
       name                                pnp-svc
       register                            0
       action_url                          /pnp/index.php/graph?host=$HOSTNAME$ \
&srv=$SERVICEDESC$' class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=$SERVICEDESC$
}
```

Dann müssen Sie noch in derselben Datei die vorhandenen Definitionen für `generic-host` und `generic-service` wie folgt ergänzen:

```
[...]
define host {
       name                                generic-host
       use                                 pnp-hst
[...]
```

```
define service {
       name                                    generic-service
       use                                     pnp-svc
[...]
```

Graph-Integration: Hierbei werden entsprechende Icons für alle Geräte und Services definiert, auch für solche ohne Graphen. Alternativ haben Sie auch die Möglichkeit, die Vorlage `pnp-svc` nur bei den Maschinen bzw. Diensten einzubinden, für die Sie Graphen haben oder anzeigen möchten.

Nach einem Neustart von Icinga, wird in der soeben als Vorlage definierten Konfigurationsoption `action_url` jeder Maschine und jedem Dienst ein Graphen-Symbol hinzugefügt. Durch eine weitere Ergänzung können Sie sich die Graphen bereits bei einem Darüberfahren mit der Maus (`mouse-over`) anzeigen lassen. Kopieren Sie hierzu die als Vorlage gelieferte Konfigurationsdatei *status-header.ssi* in das Verzeichnis */usr/share/icinga/ssi/* und passen Sie ihre Rechte wie folgt an und starten Icinga anschließend neu:

```
moni-suse122-icinga:~ # cp /usr/share/doc/packages/pnp4nagios/example/ssi/
status-header.ssi /usr/share/icinga/ssi/
moni-suse122-icinga:~ # chmod 644 /usr/share/icinga/ssi/status-header.ssi
moni-suse122-icinga:~ # service icinga restart
```

Damit weist die klassische Oberfläche nun Symbole für die Graphen auf. Beim Darüberfahren mit der Maus sehen Sie eine Vorschau, durch Klick gelangen Sie auf die Seite von PNP4Nagios mit dem entsprechenden Graphen.

Integration von PNP4Nagios in Icinga-Web

Die Integration von PNP4Nagios mit Icinga-Web ist deutlich einfacher. Hierfür benötigten Sie die Vorlagen `pnp-host-extension.xml` und `pnp-service-extension.xml`, die Sie mit dem Paket `icinga-web-pnp` wie folgt einspielen:

```
moni-suse122-icinga:~# zypper install icinga-web-pnp_integration
```

Damit enthält nun auch die Oberfläche von Icinga-Web Symbole für die Graphen, und Sie können diese abrufen.

Expander-Icon: Icinga-Web zeigt die PNP-Icons seit Version 1.8 im sogenannten `Expander Icon` an, wo sie dann in die Hauptansicht verschoben werden können (`Move to Grid`). Des Weiteren ist zu beachten, dass die Icons nur dann angezeigt werden, wenn in der Konfiguration des Host/Service `process_perf_data 1` gesetzt ist.

Installation des Visualisierungsaddons NagVis

Sie können mit Hilfe der Erweiterung NagVis Symbole für Maschinen und Services auf einer Oberfläche beliebig anordnen. Durch die Wahl eines entsprechenden Hintergrundes erhalten die Symbole dadurch einen Mehrwert, etwa durch die Platzierung auf Fotos oder Karten. Um unter openSUSE Zugriff auf das NagVis-Paket zu erhalten, müssen Sie zusätzlich zum Repository `server:monitoring` das Repository `server:php:applications` hinzufü-

gen. Dieses enthält NagVis ab der Version 1.7.6, weist aber feste Paketabhängigkeiten mit den Nagios-Paketen auf und lässt sich somit erst einmal nur mit Nagios installieren.

Sollte der Fehler bis zum Erscheinen dieses Buchen behoben sein, empfehlen wir Ihnen, dass Sie sich bei der paketbasierten Installation von NagVis an dem entsprechenden Abschnitt des Installationsrezeptes für Nagios unter Suse (siehe Rezept 1.12, *Nagios unter SLES|openSUSE installieren*) orientieren. Anderenfalls müssen Sie mit der sourcebasierten Installation vorliebnehmen, können dann aber auch auf die aktuelle NagVis-Version zurückgreifen. Orientierung bei bei der Installation bietet das entsprechende Rezept unserer Debian-Referenzinstallation (siehe Rezept 1.4, *Icinga aus den Quellen installieren*).

Diskussion

Die Ablageorte für diverse Komponenten und Konfigurationsdateien sind für Sie von Bedeutung, damit Sie schnell die richtige Stelle für gewünschte Änderungen finden können. Der Tabelle 1-16 können Sie zunächst entnehmen, an welchen Speicherorten die einzelnen Komponenten der Icinga-Installation hier abgelegt wurden.

Tabelle 1-16: Installationsorte der Komponenten von Icinga

Komponente	Pfad
Programm	/usr/sbin/icinga
Haupt-Konfigurationsdatei	/etc/icinga/icinga.cfg
Plugins	/usr/lib/nagios/plugins/
Logdatei	/var/log/icinga/icinga.log
Webserver-Konfiguration	/etc/apache2/conf.d/icinga.conf
Webseiten	/usr/share/icinga/

Das Addon IDOUtils verwendet die in Tabelle 1-17 aufgeführten Speicherorte:

Tabelle 1-17: Installationsorte der IDOUtils

Komponente	Pfad
Konfigurationsdatei idomod	/etc/icinga/idomod.cfg
Konfigurationsdatei ido2db	/etc/icinga/ido2db.cfg
Konfigurationsdatei Event-Broker-Modul	/etc/icinga/modules/idoutils.cfg
Debug-Logs	/var/lib/icinga/*.debug

Das Addon Icinga-Web verwendet die in Tabelle 1-18 aufgeführten Speicherorte.

Tabelle 1-18: Pfade zu den Komponenten von Icinga-Web

Komponente	Pfad
Haupt-Konfigurationsdateien	/etc/icinga-web/
Webserver-Konfiguration	/etc/apache2/conf.d/icinga-web.conf
Webseiten	/usr/share/icinga-web/pub/

Das Addon PNP4Nagios hat hier die in Tabelle 1-19 aufgezeigten Installationsorte verwendet.

Tabelle 1-19: Pfade zu den Komponenten von PNP4Nagios

Komponente	Pfad
Konfigurationsdateien	/etc/pnp4nagios/
Haupt-Konfigurationsdatei	/etc/pnp4nagios/config.php
NPCD-Konfigurationsdatei	/etc/pnp4nagios/npcd.cfg
Webserver-Konfiguration	/etc/apache2/conf.d/nagios-pnp.conf
Webseiten	/usr/share/pnp4nagios/
Perfdata-Script zur Verarbeitung	/usr/lib/nagios/plugins/process_perfdata.pl
Perfdata-Konfigurationsdatei	/etc/pnp4nagios/process_perfdata.cfg
Logdateien	/var/log/pnp4nagios/

Bei der Installation wurden die Plugins aus dem Paket `nagios-plugins` installiert. Auf die enthaltenen Plugins werden wir in den Rezepten der Kapitel Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen* und Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen* genauer eingehen. Für die Überwachung von System-Updates können Sie das Paket `nagios-plugins-zypper` installieren, das das Plugin `check_zypper` enthält (siehe Rezept zu `check_apt` Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*). Zusätzliche Plugins stehen Ihnen bei Bedarf über das Paket `nagios-plugins-extras` und weitere Pakete zur Verfügung.

Ein Grapher ist unter openSUSE leider nicht direkt paketiert. Zur Installation von PNP4Nagios und bzw. oder Nagvis müssen Sie also auf das Repository `server:monitoring` zurückgreifen. Im Zweifel steht Ihnen auch der klassische Weg über die sourcebasierte Installation offen. Diese haben wir (allerdings unter Debian) ausführlich in Rezept 1.4, *Icinga aus den Quellen installieren* erläutert.

Als weiteres Addon können Sie NRPE (Nagios Remote Plugin Executor) installieren (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*).

Siehe auch

- Rezept 1.1, *Entscheidungshilfe für die Installation*
- Webseite des Icinga-Projekts: *https://www.icinga.org/*
- Webseite des server:monitoring Repositories: *http://download.opensuse.org/repositories/server:/monitoring/openSUSE_12.2/*
- Webseite des Repositories server:php:applications: *http://download.opensuse.org/repositories/server:/php:/applications/openSUSE_12.2/*
- Icinga-HOWTOs für die paketbasierte Installation: *https://wiki.icinga.org/display/howtos/Setting+up+Icinga+with+IDOUtils*

- Icinga-Fehlersuche: *https://wiki.icinga.org/display/testing/Icinga+Web+Testing*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Webseite von openSUSE: *http://de.opensuse.org/*
- Rezept 1.4, *Icinga aus den Quellen installieren*
- Rezept zur paketbasierten Installation von Nagios unter openSUSE: Rezept 1.12, *Nagios unter SLES|openSUSE installieren*
- Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zum Aufbau und zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Kapitel zur Konfiguration von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere das Rezept 4.3, *Befehle hinzufügen (command)*, Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* sowie Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Kapitel zur Datenvisualisierung mit PNP4Nagios und NagVis: Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*
- Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezept zur Überwachung von System-Updates: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*
- Rezepte zur Installation von NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

1.8 Icinga unter RHEL|CentOS|Fedora installieren

Problem

Sie möchten Icinga unter unter Red Hat Enterprise Linux (RHEL)|CentOS|Fedora paketbasiert installieren.

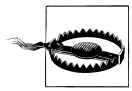
Lösung

Zunächst zeigen wie Ihnen, wie Sie Icinga unter CentOS installieren, dabei verläuft die Installation in RHEL und Fedore weitgehend analog. Darauf aufbauend erläutern wir die Installation von optionalen Addons, die ebenfalls als Pakete angeboten werden. Anschlie-

ßend zeigen wir Ihnen, wie Sie das Datenbank-Addon IDOUtils installieren, mit dem Sie die Daten in einer Datenbank für weitere Addons vorhalten können. Für die Addons PNP4Nagios und NagVis ist leider in dem von uns gewählten Repository kein Paket vorhanden. Für deren Installation müssten Sie mit der sourcebasierten Installation vorliebnehmen (siehe Rezept 1.4, *Icinga aus den Quellen installieren*).

Installation Icinga

Icinga ist nicht Teil der offiziellen RHEL-Distribution und damit also ebenfalls nicht in CentOS enthalten. Um Icinga paketorientiert auf einem auf RHEL-basierenden System zu installieren, müssen Sie die Paketquelle RepoForge (ehemals RPMforge) einbinden.



Paketquellen RepoForge und EPEL: Wir verwenden in diesem Rezept die Paketquelle RPMForge. Diese ist zu Extra Packages for Enterprise Linux (EPEL) nicht kompatibel, und Sie sollten diese daher nicht miteinander mischen (siehe dazu *http://wiki.centos.org/AdditionalResources/Repositories*). Bei Fedora ist diese Quelle als Vorgabe aktiviert, weshalb Sie hier leicht auf Probleme stoßen können.

Mit dem Kommando `yum repolist` können Sie überprüfen, welche Paketquellen bereits in Ihr System eingebunden sind. Wenn RepoForge nicht in Ihr System eingebunden ist, können Sie dies nachholen, indem Sie zuerst den zugehörigen GPG-Schlüssel in Ihre RPM-Datenbank installieren und die Quelle dann wie folgt hinzufügen:

```
[root@moni-centos63-icinga ~]# rpm --import http://apt.sw.be/RPM-GPG-KEY.dag.txt
[root@moni-centos63-icinga ~]# rpm -ivh http://pkgs.repoforge.org/rpmforge-release/
rpmforge-release-0.5.2-2.el6.rf.x86_64.rpm
[root@moni-centos63-icinga ~]# yum repolist
[...]
rpmforge                                          RHEL 6 - RPMforge.net -
dag                                               4,458
[...]
```

Die RepoForge-Quelle enthielt in unserem Fall Icinga ab der Version 1.7.2. Installieren Sie zunächst das Paket `icinga` inklusive der dem System bekannten Abhängigkeiten:

```
[root@moni-centos63-icinga ~]# yum install icinga icinga-gui
```

Deinstallation: Sofern Sie eine Deinstallation erwarten und dabei sichergehen möchten, dass alle mitinstallierten Pakete wieder entfernt werden, sollten Sie sich an dieser Stelle die Liste der auf Ihrem System mitinstallierten Pakete kopieren. Der Hintergrund dabei ist der, dass bei der Deinstallation über Paketmanager regelmäßig ganze Pakete auf dem System verbleiben (dann nämlich, wenn diese in anderen installierten Paketen als optionale Abhängigkeit gelistet sind).

Dabei wird mit dem Paket `httpd` der Apache2-Webserver gleich mit-installiert. Icinga enthält ein Web-Interface, dass standardmäßig eine Authentifizierung über den Webserver

mittels `http-auth` erfordert (siehe auch Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*). Richten Sie deshalb zunächst einen entsprechenden Benutzer `icinga-admin` mit einem Passwort ein:

```
[root@moni-centos63-icinga ~]# htpasswd -c /etc/icinga/password icingaadmin
```

Falls Sie über das Netzwerk von außen auf den Webserver zugreifen möchten, lassen Sie Port 80 in der lokalen Firewall zu, indem Sie in der Datei */etc/sysconfig/iptables* die folgende Zeile oben in den Abschnitt ":OUTPUT ACCEPT [0:0]" eintragen:

```
[...]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
[...]
```

Icinga führt Tests über sogenannte Plugins durch. Über die Abhängigkeiten im Paketmanagement wurden noch keine Plugins installiert. Die automatisch eingerichteten Test auf `localhost` würden deshalb Fehlermeldungen erzeugen. Installieren Sie daher die über das Paketmanagement verfügbaren Plugins:

```
[root@moni-centos63-icinga ~]# yum install nagios-plugins
```

Damit wären die grundlegenden Komponenten prinzipiell einsatzfähig, jedoch wird das standardmäßig aktivierte SELinux nicht automatisch für die Belange von Icinga (und später betrachteten Addons) angepasst. Betrachtungen zur Anpassung des SELinux für Icinga und die einzelnen Module sind nicht Bestandteil dieses Buches. Ändern Sie bitte im Zweifel den folgenden Wert in der Datei */etc/selinux/config*, um das SELinux in einen Modus zu schalten, in dem es alle Operationen zulässt und lediglich eine entsprechende Protokollierung vornimmt:

```
[...]
#SELINUX=enforcing
SELINUX=permissive
[...]
```

Damit diese Änderung wirksam wird, müssen Sie das System neu starten. In unserem Fall waren der Webserver und Icinga noch deaktiviert. Sie können den automatischen Start der beiden Anwendungen wie folgt sicherstellen:

```
[root@moni-centos63-icinga ~]# chkconfig httpd on
[root@moni-centos63-icinga ~]# chkconfig icinga on
```

`chkconfig`: Wenn ein Service automatisch gestartet werden soll, können Sie dies meist mit dem Kommando `chkconfig <service> on|off` (RHEL|CentOS|Fedora, SLES|openSUSE) konfigurieren.

Starten Sie jetzt das System neu:

```
[root@moni-centos63-icinga ~]# reboot
```

Rufen Sie dann den URL *http://127.0.0.1/icinga* auf. Wenn Sie von einem anderen Rechner auf den Monitoring-Server zugreifen, müssen Sie anstelle von `localhost` die entspre-

chende IP-Adresse aufrufen. Icinga läuft und führt bereits Tests am lokalen System durch. Zur Anmeldung verwenden Sie den Benutzernamen `icingaadmin` mit dem vorher von Ihnen gesetzten Passwort.

Oberflächentest von der Kommandozeile: Sollte der entfernte Webzugriff auf Ihr System (noch) nicht möglich sein, beispielsweise aufgrund einer Firewall, können Sie auch über *http://localhost/nagios/* auf Ihr frisch installiertes Icinga zugreifen. Wenn Sie über keine graphische Oberfläche und bzw. oder keinen lokalen Webbrowser verfügen, können Sie die Erreichbarkeit der Oberfläche durch folgende Anweisung mit dem Kommandozeilen-Werkzeug `telnet` testen:

```
[root@moni-centos63-icinga ~]# telnet localhost 80
[...]
GET /icinga
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Authorization Required</title>
</head>
[...]
```

Installation der Datenbankanbindung über IDOUtils

Zum Zeitpunkt dieser Installation sind die IDOUtils (Icinga Data Out Utilities) im RepoForge-Repository ab der Version 1.6.1 enthalten. Die IDOUtils ermöglichen die Anbindung von Icinga an einen Datenbankserver, in den später die Daten geschrieben werden. Beginnen Sie damit, einen lokalen MySQL-Datenbankserver zu installieren:

```
[root@moni-centos63-icinga ~]# yum install mysql-server
```

Auf diese Weise haben Sie den Datenbankserver installiert, allerdings ist er weder gestartet noch mit einem Passwort versehen. Starten Sie ihn deshalb zunächst und vergeben Sie dann ein Passwort für den MySQL-Benutzer `root`:

```
[root@moni-centos63-icinga ~]# service mysqld start
[root@moni-centos63-icinga ~]# mysqladmin -u root password 'DBROOTPW'
```

MySQL-Passwörter: Wenn Sie das hier gesetzte Passwort für den MySQL-Nutzer `root` ändern möchten, funktioniert dies folgendermaßen:

```
root@moni-centos63-icinga:~# mysqladmin -u root -p PASSWORD
```

Um das Passwort eines nicht-administrativen MySQL-Benutzers zu ändern, können Sie sich von der Kommandozeile aus als Benutzer `root` auf dem Server anmelden und das nachfolgend angeführte Kommando nutzen:

```
root@moni-centos63-icinga:~# mysql -u root -p
mysql> use mysql;
mysql> update user set Password=DBUSERPW('NEWDBUSERPW') WHERE User='DBUSER';
mysql> exit
```

Nun können Sie sich mit diesem Passwort am Server anmelden und eine Datenbank `icinga` sowie einen Benutzer `sql_icinga` mit Passwort (hier DBUSERPW) erstellen und den Benutzer mit entsprechenden Rechten versehen:

```
[root@moni-centos63-icinga ~]# mysql -u root -p
mysql> CREATE DATABASE icinga;
mysql> GRANT ALL ON icinga.* TO 'sql_icinga'@'localhost' IDENTIFIED BY 'DBUSERPW';
mysql> FLUSH PRIVILEGES;
mysql> exit
```

Installieren Sie nun das Paket `icinga-idoutils-libdbi-mysql`:

```
[root@moni-centos63-icinga ~]# yum install icinga-idoutils-libdbi-mysql
```

Die Datenbank initialisieren Sie nun mittels der beim Paket `idoutils` mitgelieferten Vorlage:

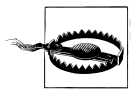
```
[root@moni-centos63-icinga /]# mysql -u sql_icinga -p icinga <
/usr/share/doc/icinga-idoutils-libdbi-mysql-1.7.2/db/mysql/mysql.sql
```

Damit die IDOUtils auf die Datenbank zugreifen können, müssen die Zugangsdaten in der Datei */etc/icinga/ido2db.cfg* hinterlegt sein. Passen Sie hier die nötigen Zeilen an (zumindest `db_pass`, hier auch `db_user`):

```
db_servertype=mysql
db_host=localhost
db_port=3306
db_name=icinga
db_prefix=icinga_
db_user=sql_icinga
db_pass=DBUSERPW
```

Das sogenannte Broker-Modul ist über den Ordner */etc/icinga/modules/* automatisch eingebunden. Der MySQL-Server und die IDOUtils sind in unserem Fall noch nicht für den automatischen Start eingerichtet. Die können Sie wir folgt nachholen:

```
[root@moni-centos63-icinga ~]# chkconfig mysqld on
[root@moni-centos63-icinga ~]# chkconfig ido2db on
```



Eventbroker-Modul: Seit IDOUtils Version 1.4 sollte das Eventbroker-Modul `idomod` über eine Modules-Definition im modules-Unterverzeichnis */etc/icinga/modules/* eingebunden werden. Sie sollten daher nicht zusätzlich den entsprechenden Eintrag `broker_modul` in der Hauptkonfigurationsdatei `/etc/icinga/icinga.cfg` aktivieren, da es in einem solchen Fall, verursacht durch die Doppeldefinition, zu schwer zu analysierenden Fehlern kommen kann.

Nun müssen Sie nur noch den ido2db-Agent und Icinga neu starten, damit alle vorgenommenen Änderungen wirksam werden.

```
[root@moni-centos63-icinga ~]# service ido2db start
[root@moni-centos63-icinga ~]# service icinga restart
```

Jetzt können Sie überprüfen, ob die Ergebnisse auch in die Datenbank geschrieben werden. Schauen Sie dazu nach, ob der Eintrag `status_update_time` aus der Tabelle `icinga_programstatus` Einträge enthält:

```
[[root@moni-centos63-icinga ~]# mysql -u root -p
mysql> USE icinga;
mysql> SELECT status_update_time FROM icinga_programstatus;
```

```
+---------------------+
| status_update_time  |
+---------------------+
| 2013-01-30 12:35:12 |
+---------------------+
1 row in set (0.00 sec)

mysql> exit
```

Damit haben Sie die Datenbankanbindung für Icinga über IDOUtils erfolgreich installiert, konfiguriert und getestet. Die Daten werden jetzt in die angelegte Datenbank geschrieben.

Installation der Weboberfläche Icinga-Web

Icinga bietet Ihnen zwei Nutzerschnittstellen. Zum einen die an Nagios orientierte Oberfläche Icinga-Classic und zum anderen eine für Icinga neu entwickelte Weboberfläche names Icinga-Web. Im Gegensatz zu Icinga-Classic, das die aus Nagios bekannten `status.dat`/`objects.cache`-Dateien verwendet, nutzt Icinga-Web die von den IDOUtils bereitgestellte Datenbank. Icinga-Classic und Icinga-Web lassen sich beide problemlos parallel betreiben.

Installieren Sie zunächst einige erforderliche PHP-Pakete und anschließend Icinga-Web wie folgt:

```
[root@moni-centos63-icinga ~]# yum install php php-cli php-common php-gd php-ldap
php-pdo php-pear php-xml php-mysql libxslt
[root@moni-centos63-icinga ~]# yum install icinga-web
```

Icinga-Web ist darauf ausgelegt Daten in einer eigenen Datenbank zu speichern. Sie benötigen diese Datenbank zusätzlich zu der bereits für die IDOUtils installierten. Legen Sie die neue Datenbank zunächst unter Nutzung des MySQL-Benutzers `root` an und richten Sie dabei außerdem einen entsprechenden MySQL-Benutzer `sql_icinga-web` für den Zugriff ein (verwenden Sie anstelle des hier genutzten Passworts `DBUSERPW` bitte wieder ein eigenes).

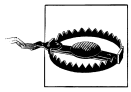
```
[root@moni-centos63-icinga ~]# mysql -u root -p
[...]
mysql> CREATE DATABASE icinga_web;
mysql> GRANT USAGE ON icinga_web.* TO 'sql_icinga-web'@'localhost' IDENTIFIED BY
'DBUSERPW' WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0;
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, INDEX ON icinga_web.*
TO 'sql_icinga-web'@'localhost';
mysql> FLUSH PRIVILEGES;
mysql> EXIT;
```

Importieren Sie jetzt die vorgegebene Struktur in die noch leere Datenbank (hier mit dem oben angelegten MySQL-Benutzer `sql_icinga-web`):

```
[root@moni-centos63-icinga ~]# mysql -u sql_icinga-web -p icinga_web <
/usr/share/icinga-web/etc/schema/mysql.sql
```

Sie müssen nun noch die Zugriffsdaten für diese Datenbank in der Datei */etc/icinga-web/conf.d/databases.xml* hinterlegen. Sie benötigen zunächst die Zugangsdaten (Benutzername und Passwort) für die Datenbank `icinga_web` und dann außerdem die für die Datenbank `icinga`, die für IDOUtils erstellt wurde.

```
[...]
    <db:database name="icinga_web" class="AppKitDoctrineDatabase">
 <ae:parameter name="dsn">mysql://sql_icinga-
web:DBUSERPW@localhost:3306/icinga_web</ae:parameter>
[...]
  </ae:parameter>
    </db:database>
[...]
    <db:database xmlns="http://agavi.org/agavi/config/parts/databases/1.0" name="icinga" c
lass="IcingaDoctrineDatabase">
 <ae:parameter name="dsn">mysql://sql_icinga:DBUSERWD@localhost:3306/icinga</ae:parameter>
[...]
  </ae:parameter>
    </db:database>
[...]
```



Zugriff verweigert: In CentOS sind die betroffenen Konfigurationsabschnitte in der Datei */etc/icinga-web/conf.d/databases.xml* durch die Zeichenketten `<!--` und `-->` auskommentiert. Achten Sie darauf, die entsprechenden Kommentarzeichen zu entfernen, anderenfalls kann Icinga-Web nicht auf die Zugangsdaten zugreifen. Ein Editor mit Syntax-Highlightning, wie beispielsweise vim mit `syntax on`, hilft Ihnen bei der Identifizierung von auskommentierten Blöcken.

Nachdem Sie die Konfigurationsdateien von Icinga-Web angepasst haben, müssen Sie anschließend das Script zum Löschen der Cache-Dateien ausführen. Das gilt auch für spätere Anpassungen:

```
[root@moni-centos63-icinga ~]# /usr/bin/icinga-web-clearcache
```

Starten Sie jetzt den Webserver neu, damit er die hinzugekommene Konfigurationsdatei einliest:

```
[root@moni-centos63-icinga ~]# service httpd restart
```

Die Installation der zusätzlichen Oberfläche ist damit abgeschlossen. Die neue Oberfläche können Sie sich jetzt unter *http://127.0.0.1/icinga-web* anschauen (siehe dazu Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*). Loggen Sie sich mit dem Nutzernamen `root` und dem Passwort `password` erstmalig ein. Abschließend sollten Sie dann unbedingt das Passwort neu setzen. Das entsprechende Menü finden Sie rechts oben.

`no data`: Falls Sie vor dem Problem stehen, dass Icinga-Web keine Daten anzeigt, stellen Sie sicher, dass IDOUtils und Icinga korrekt funktionieren. Weitere Tests finden Sie im Icinga Wiki: *https://wiki.icinga.org/display/testing/Icinga+Web+Testing*.

Installation einer graphischen Auswertung mit PNP4Nagios

Bei PNP4Nagios handelt es sich um ein Addon zur Analyse und Darstellung von Performancedaten. PNP4Nagios ist zur Zeit noch nicht Teil der RepoForge-Paketquelle. Sie müssen hier gegebenenfalls auf die quellbasierte Installation ausweichen. Sie können sich an unserem Rezept für die quellbasierte Installation von Icinga orientieren (siehe Rezept 1.4, *Icinga aus den Quellen installieren*).

Installation des Visalisierungsaddons NagVis

Mit Hilfe des NagVis-Addons können Sie Symbole für Maschinen und Services auf einer Oberfläche beliebig anordnen. Diese erhalten durch die Wahl eines entsprechenden Hintergrundes einen Mehrwert, etwa durch die Platzierung auf Fotos oder Karten. Das Addon ist nicht Bestandteil der von uns genutzten Paketquellen, so dass Sie hier gegebenenfalls wieder auf die Installation aus den Quellen ausweichen müssen, wie sie in unserem Rezept für die quellbasierte Installation von Icinga (siehe Rezept 1.4, *Icinga aus den Quellen installieren*) beschrieben ist.

Diskussion

Die mit dem Paket `icinga` installierten Komponenten nutzen die in Tabelle 1-20 aufgezeigten Pfade.

Tabelle 1-20: Ablageorte der Komponenten von Icinga

Komponente	Pfad
Programm	/usr/bin/icinga
Haupt-Konfigurationsdatei	/etc/icinga/icinga.cfg
Plugins	/usr/lib64/nagios/plugins
Logdatei	/var/log/icinga/icinga.log
Webserver-Konfiguration	/etc/httpd/conf.d/icinga.conf
Webseiten	/usr/share/icinga

Das Addon IDOUtils verwendet die in Tabelle 1-21 aufgeführten Dateien und Ordner.

Tabelle 1-21: Pfade der wichtigsten Dateien von IDOUtils

Komponente	Pfad
Konfigurationsdatei ido2db	/etc/icinga/ido2db.cfg
Konfigurationsdatei idomod	/etc/icinga/idomod.cfg
Konfigurationsdatei Event-Broker-Modul	/etc/icinga/modules/idoutils.cfg
Debug-Logs	/var/log/icinga/*.debug

Das Addon Icinga-Web verwendet die in Tabelle 1-22 aufgeführten Speicherorte.

Tabelle 1-22: Pfade zu den Komponenten von Icinga-Web

Komponente	Pfad
Haupt-Konfigurationsdateien	/etc/icinga-web/
Webserver-Konfiguration	/etc/httpd/conf.d/icinga-web.conf
Webseiten	/usr/share/icinga-web/pub/

Die in den Nagios-Plugins enthaltenen Plugins erläutern wir in den Rezepten der Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen* und Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*. Für die Überwachung von System-Updates können Sie das Plugin `check_updates` einsetzen (siehe Rezept zu `check_apt` Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*). Als weiteres Addon können Sie NRPE (Nagios Remote Plugin Executor) installieren (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*).

Siehe auch

- Rezept mit Entscheidungshilfe zur Installation: Rezept 1.1, *Entscheidungshilfe für die Installation*
- Webseite des Icinga-Projekts: *https://www.icinga.org/*
- Webseite des Repoforge-Repositories: *http://repoforge.org/*
- Icinga-HOWTOs für die paketbasierte Installation: *https://wiki.icinga.org/display/howtos/Setting+up+Icinga+with+IDOUtils*
- Icinga-Fehlersuche: *https://wiki.icinga.org/display/testing/Icinga+Web+Testing*
- Webseite von Nagiosplugins: *http://nagiosplugins.org//*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Webseite von CentOS: *http://www.centos.org/*
- Rezept zur quellenbasierten Installation von Icinga: Rezept 1.4, *Icinga aus den Quellen installieren*
- Rezept zur paketbasierten Installation von Nagios unter CentOS: Rezept 1.13, *Nagios unter RHEL|CentOS|Fedora installieren*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zum Aufbau und zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Kapitel zur Konfiguration von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere das Rezept 4.3, *Befehle hinzufügen (command)*,

Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* sowie Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*

- Kapitel zur Datenvisualisierung mit PNP4Nagios und NagVis: Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*
- Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezept zur Überwachung von System-Updates: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*
- Rezepte zur Installation von NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

1.9 Nagios aus den Quellen installieren

Problem

Sie möchten ein aktuelles Nagios aus den Quellen installieren.

Lösung

Wir zeigen Ihnen im Folgenden zunächst schrittweise die Installation von Nagios v3.4 aus Quellcode. Darauf aufbauend werden wir außerdem erläutern, wie Sie die Addons NDOUtils (Datenbankanbindung), PNP4Nagios (Graphen) und NagVis (Karten) installieren und integrieren. Unsere folgenden Beispiele basieren dabei auf einer Minimalinstallation von Debian 7 (Codename Wheezy) mit zusätzlich installiertem SSH.

Vorbereitung der Installation

Tarball oder GIT/SVN: In diesem Buch beschreiben wir zwei unterschiedliche Wege, um an den aktuellen Sourcecode zu gelangen. Für die Nagios-Source-Installation zeigen wir Ihnen die Installation aus einem sogenannten Tarball. Alternativ können Sie auch aus GIT/SVN installieren. Die aktuellen Quellen für Nagios via SVN finden Sie unter *http://sourceforge.net/projects/nagios/develop*. Die Installation aus GIT haben wir Ihnen im Rezept für die Icinga-Source beschrieben Rezept 1.4, *Icinga aus den Quellen installieren*.

Laden Sie sich zunächst die Quellen von den Seiten des Nagios-Projektes herunter und packen diese aus:

```
root@moni-wheezy-nagsrc:/usr/src# wget
http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-3.4.4.tar.gz
root@moni-wheezy-nagsrc:/usr/src# tar xvfz nagios-3.4.4.tar.gz
```

Sie benötigen dann noch zusätzliche Pakete, um überhaupt Quellcode übersetzen zu können. Da mit Nagios eine Weboberfläche installiert werden soll, installieren Sie auch gleich noch den Webserver Apache2 sowie die PHP5-Erweiterung für diesen. Darüber hinaus benötigen Sie noch das Entwicklerpaket von GD. Mit der folgenden Anweisung installieren Sie diese Komponenten:

```
root@moni-wheezy-nagsrc:~# aptitude install build-essential libapache2-mod-php5
libgd2-xpm-dev
```

Damit haben Sie die für die Installation von Nagios erforderliche Software installiert. Richten Sie an dieser Stelle die vorgesehene Authentifizierung über `http-auth` (siehe auch Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*) für den Benutzer `nagiosadmin` ein:

```
root@moni-wheezy-nagsrc:~# mkdir -p /usr/local/nagios/etc/
root@moni-wheezy-nagsrc:~# htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
```

Nagios soll nach dem Start unter einem Benutzerkonto laufen. Legen Sie dieses an und versehen Sie es mit einem Passwort:

```
root@moni-wheezy-nagsrc:~# useradd -m nagios
root@moni-wheezy-nagsrc:~# passwd nagios
```

Debian wird dabei automatisch eine entsprechende Gruppe mit dem gleichen Namen anlegen und den betreffenden Benutzer dieser Gruppe zuweisen, so dass Sie diesen Schritt hier nicht manuell durchführen müssen. Sie benötigen dann eine weitere Gruppe, die es Ihren Mitgliedern erlaubt, Befehle an Nagios zu übermitteln. Legen Sie diese an und weisen Sie ihr die Benutzerkonten `nagios` und `www-data` zu, damit Sie später über die Weboberfläche Anweisungen an Nagios absetzen können.

```
root@moni-wheezy-nagsrc:~# groupadd nagcmd
root@moni-wheezy-nagsrc:~# usermod -a -G nagcmd nagios
root@moni-wheezy-nagsrc:~# usermod -a -G nagcmd www-data
```

Sie können mit folgenden Anweisungen überprüfen, ob der Apache2-Webserver automatisch nach jedem Bootvorgang gestartet wird:

```
root@moni-wheezy-nagsrc:~# find /etc/rc?.d/ -iname "*apache2"
/etc/rc0.d/K02apache2
/etc/rc1.d/K02apache2
/etc/rc2.d/S17apache2
/etc/rc3.d/S17apache2
/etc/rc4.d/S17apache2
/etc/rc5.d/S17apache2
/etc/rc6.d/K02apache2
```

insserv: Wenn ein Service automatisch gestartet werden soll, können Sie dies mit dem Kommando `insserv <service>` konfigurieren.

Installation von Nagios

Damit haben Sie alle grundlegenden Voraussetzungen für eine Installation von Nagios erfüllt. Starten Sie die Anpassung an Ihr System unter Angabe der soeben angelegten Gruppe. Sofern dabei keine Fehler gemeldet werden, können Sie dann mit Übersetzung der Quellen und der Installation der Haupt-Programmbestandteile fortfahren.

```
root@moni-wheezy-nagsrc:~# cd /usr/src/nagios
root@moni-wheezy-nagsrc:/usr/src/nagios# ./configure --with-command-group=nagcmd
root@moni-wheezy-nagsrc:/usr/src/nagios# make all
root@moni-wheezy-nagsrc:/usr/src/nagios# make install
root@moni-wheezy-nagsrc:/usr/src/nagios# make install-init
root@moni-wheezy-nagsrc:/usr/src/nagios# make install-config
root@moni-wheezy-nagsrc:/usr/src/nagios# make install-commandmode
```

Abschließend installieren Sie zusätzlich eine Konfigurationsdatei für den Webserver:

```
root@moni-wheezy-nagsrc:/usr/src/nagios# make install-webconf
```

Hinweise auf neue Versionen: Wenn Sie Nagios über den Quellcode installieren, entfällt die automatische Versorgung mit Aktualisierungen. Tragen Sie sich daher in die entsprechende Mailing-Liste ein (*https://lists.sourceforge.net/lists/listinfo/nagios-announce*), damit Sie über Neuerungen auf dem Laufenden gehalten werden.

Installation der Standard-Plugins

Tests werden in Nagios mit Hilfe sogenannter Plugins durchgeführt. Nagios selbst beinhaltet zunächst keine Plugins, kann also diesbezüglich erst einmal nicht tätig werden. Installieren Sie deshalb als Nächtes das Standardset von Plugins, `nagios-plugins`. Dazu besorgen Sie sich im ersten Schritt auch hier die entsprechenden Quellen und entpacken sie:

```
root@moni-wheezy-nagsrc:~# cd /usr/src
root@moni-wheezy-nagsrc:/usr/src# wget
http://downloads.sourceforge.net/project/nagiosplug/nagiosplug/1.4.16/nagios-plugins-
1.4.16.tar.gz
root@moni-wheezy-nagsrc:/usr/src# tar xvfz nagios-plugins-1.4.16.tar.gz
```

Auch diese Plugins benötigen nun wieder andere Programme als Voraussetzung. Installieren Sie diese über den Paketmanager:

```
root@moni-wheezy-nagsrc:/usr/src# aptitude install libradiusclient-ng-dev libgnutls-dev
libssl-dev libldap2-dev libmysqlclient-dev smbclient libnet-snmp-perl snmp libsnmp-dev
fping
```

Sie werden dabei nach der Standard-Arbeitsgruppe für den Samba-Server gefragt, wobei Sie im Zweifel einfach den vorgegebenen Wert (meist `workgroup`) bestätigen können. Nachfolgend starten Sie die Anpassung an Ihr System. Anschließend übersetzten und installieren Sie die Plugins:

```
root@moni-wheezy-nagsrc:/usr/src# cd nagios-plugins-1.4.16/
root@moni-wheezy-nagsrc:/usr/src/nagios-plugins-1.4.16# ./configure
--with-nagios-user=nagios --with-nagios-group=nagios
root@moni-wheezy-nagsrc:/usr/src/nagios-plugins-1.4.16# make all
root@moni-wheezy-nagsrc:/usr/src/nagios-plugins-1.4.16# make install
```

Abhängigkeiten: Bei welchen Plugins dabei Warnungen bezüglich nicht erfüllter Abhängigkeiten bestehen, können Sie sich anschließend mit dem folgenden Befehl anzeigen lassen:

```
root@moni-wheezy-icisrc:/usr/src/nagiosplug# cat config.log | grep WARNING
```

Dabei werden die übersetzten Plugins in das Verzeichnis */usr/local/nagios/libexec/* kopiert. Damit haben Sie zunächst alle erforderlichen Komponenten installiert. Um Nagios zu testen, müssen Sie jetzt noch den bereits laufenden Webserver neu starten, damit die Nagios-Konfiguration eingelesen wird. Führen Sie dabei auch gleich noch einen Neustart von Nagios durch:

```
root@moni-wheezy-nagsrc:~# service apache2 restart
root@moni-wheezy-nagsrc:~# service nagios restart
```

Nagios läuft nun und führt bereits erste Checks an der lokalen Maschine durch. Öffnen Sie über *http://127.0.0.1/nagios* beziehungsweise die jeweilige IP-Adresse die Oberfläche von Nagios und verschaffen Sie sich einen ersten Eindruck.

Oberflächentest von der Kommandozeile: Falls der entfernte Webzugriff auf Ihr System (noch) nicht möglich ist, beispielsweise aufgrund einer Firewall, können Sie auch über *http://127.0.0.1/nagios/* auf Ihr frisch installiertes Nagios zugreifen. Wenn Sie über keine graphische Oberfläche und bzw. oder keinen lokalen Webbrowser verfügen, können Sie die Erreichbarkeit der Oberfläche durch folgende Anweisung mit dem Kommandozeilen-Werkzeug `telnet` testen:

```
root@moni-wheezy-nagsrc:~# telnet localhost 80
[...]
GET /nagios
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Authorization Required</title>
</head><body>
<h1>Authorization Required</h1>
[...]
```

Stellen Sie abschließend sicher, dass Nagios automatisch gestartet wird:

```
root@moni-wheezy-nagsrc:~# insserv nagios
```

Fehlende LSB-Tags: Debian benötigt ab Version 6 LSB (Linux Standards Base)-konforme Startscripte. Sollten Sie also die Fehlermeldung

```
root@moni-wheezy-nagsrc:~# insserv nagios
insserv: warning: script 'K01nagios' missing LSB tags and overrides
insserv: warning: script 'nagios' missing LSB tags and overrides
```

erhalten, dann fügen Sie einfach den folgenden Abschnitt am Anfang der Datei */etc/init.d/nagios* ein:

```
### BEGIN INIT INFO
# Provides: scriptname
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Start daemon at boot time
# Description: Enable service provided by daemon.
### END INIT INFO
```

Installation der Datenbankanbindung über NDOUtils

Installieren Sie das Addon NDOUtils, um Daten aus Nagios in eine Datenbank zu schreiben. Besorgen Sie sich auch hier zunächst die Quellen und entpacken Sie diese:

MariaDB: Sie können mit den NDOUtils auch MariaDB als Datenbank einsetzen. Die Konfiguration funktioniert dann weitestgehend analog.


```
root@moni-wheezy-nagsrc:~# cd /usr/src/
root@moni-wheezy-nagsrc:/usr/src# wget
http://downloads.sourceforge.net/project/nagios/ndoutils-1.x/ndoutils-1.5.2/ndoutils-
1.5.2.tar.gz
root@moni-wheezy-nagsrc:/usr/src# tar xvfz ndoutils-1.5.2.tar.gz
```

Um eine Datenbank zur Verfügung stellen zu können, installieren Sie dann noch vorbereitend den Datenbankserver MySQL:

```
root@moni-wheezy-nagsrc:~# aptitude install mysql-server
```

Während der Installation werden Sie dabei nach einem Passwort für den eingerichteten MySQL-Benutzer `root` gefragt.

MySQL-Passwörter: Wenn Sie das hier gesetzte Passwort für den MySQL-Nutzer `root` ändern wollen, funktioniert dies folgendermaßen:

```
root@moni-wheezy-nagsrc:~# mysqladmin -u root -p PASSWORD
```

Um das Passwort eines nicht-administrativen MySQL-Benutzers zu ändern, können Sie sich von der Kommandozeile aus als Benutzer `root` auf dem Server anmelden und das nachfolgend angeführte Kommando nutzen:

```
root@moni-wheezy-nagsrc:~# mysql -u root -p
mysql> use mysql;
mysql> update user set Password=DBUSERPW('NEWDBUSERPW') WHERE User='DBUSER';
mysql> exit
```

Lassen Sie nun die Anpassung an Ihr System durchlaufen und übersetzen Sie anschließend die Quellen:

```
root@moni-wheezy-nagsrc:/usr/src# cd ndoutils-1.5.2/
root@moni-wheezy-nagsrc:/usr/src/ndoutils-1.5.2# ./configure
root@moni-wheezy-nagsrc:/usr/src/ndoutils-1.5.2# make
root@moni-wheezy-nagsrc:/usr/src/ndoutils-1.5.2# make fullinstall
```

Um die NDOUtils zu nutzen, benötigen Sie zunächst eine Datenbank. Legen Sie wie nachfolgend angeführt eine Datenbank an und statten Sie einen Benutzer mit den nötigen Zugriffsrechten auf diese aus (Sie benötigen hier zunächst das oben vergebene Passwort für den MySQL-Benutzer `root`):

```
root@moni-wheezy-nagsrc:~# mysql -u root -p
[...]
mysql> CREATE DATABASE nagios;
mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON nagios.* TO 'sql_nagios'@'localhost'
IDENTIFIED BY 'DBUSERPW';
mysql> FLUSH PRIVILEGES;
mysql> exit
```

In der so angelegten Datenbank können Sie dann die nötige Struktur anlegen lassen. Verwenden Sie dazu das Script `installdb` im Verzeichnis `db` (erneut unter Nutzung des MySQL-Benutzers `root`):

```
root@moni-wheezy-nagsrc:/usr/src/ndoutils-1.5.2# cd db/
root@moni-wheezy-nagsrc:/usr/src/ndoutils-1.5.2/db# chmod +x installdb
root@moni-wheezy-nagsrc:/usr/src/ndoutils-1.5.2/db# ./installdb -u root -p 'DBROOTPW'
-h localhost -d nagios
```

Vorlagen für die notwendigen Konfigurationsdateien wurden bei der Installation im Konfigurationsverzeichnis von Nagios abgelegt. Kopieren Sie diese zunächst, um die Endung »-sample« zu entfernen:

```
root@moni-wheezy-nagsrc:~# cd /usr/local/nagios/etc/
root@moni-wheezy-nagsrc:/usr/local/nagios/etc# cp ndo2db.cfg-sample ndo2db.cfg
root@moni-wheezy-nagsrc:/usr/local/nagios/etc# cp ndomod.cfg-sample ndomod.cfg
```

In der so entstandenen Datei */usr/local/nagios/etc/ndo2db.cfg* müssen Sie jetzt die Zugangsdaten für die Datenbank hinterlegen (im Beispiel hier müssen wir nur den Benutzernamen in `db_user` und das dazugehörige Nutzerpasswort in `db_pass` ändern):

```
db_servertype=mysql
db_host=localhost
db_port=3306
db_name=nagios
db_prefix=nagios_
db_user=sql_nagios
db_pass=DBUSERPW
```

Damit Nagios Daten an das zu diesem Zweck installierte Modul `ndomod` übergibt, tragen Sie es als sogenanntes Broker-Modul in die Hauptkonfigurationsdatei von Nagios (hier */usr/local/nagios/etc/nagios.cfg*) ein:

```
[...]
# EVENT BROKER MODULE(S)
[...]
broker_module=/usr/local/nagios/bin/ndomod.o config_file=/usr/local/nagios/etc/ndomod.cfg
[...]
```

Nun sind Installation und Konfiguration abgeschlossen. Starten Sie jetzt NDO2DB und führen Sie einen Neustart von Nagios durch, damit dieses anfängt, Daten an die NDOUtils zu übergeben:

```
root@moni-wheezy-nagsrc:~# service ndo2db start
root@moni-wheezy-nagsrc:~# service nagios restart
```

Jetzt können Sie nach ein paar Minuten verifizieren, ob tatsächlich Daten in die Datenbank geschrieben werden. Lesen Sie hierzu den Eintrag `status_update_time` aus der Tabelle `nagios_programstatus` aus:

```
root@moni-wheezy-nagsrc:~# mysql -u sql_nagios -p
[...]
mysql> USE nagios;
mysql> SELECT status_update_time FROM nagios_programstatus;
+---------------------+
| status_update_time  |
+---------------------+
| 2013-01-22 11:00:31 |
+---------------------+
1 row in set (0.00 sec)
mysql> exit
```

Stellen Sie abschließend sicher, dass auch NDO2DB automatisch gestartet wird:

```
root@moni-wheezy-nagsrc:~# insserv ndo2db
```

Damit haben Sie die Installation des Addons NDOUtils abgeschlossen. Nagios schreibt nun die Daten in die neu angelegte Datenbank.

Fehlende Daten: In den NDOUtils ab Version 1.5.2 werden Nachrichten nur dann in die Warteschlage weitergeleitet, wenn der Kernel sie nicht annimmt. Falls Sie alles einwandfrei konfiguriert haben und trotzdem keine Daten in Ihre Datenbank geschrieben werden, prüfen Sie, ob die Daten auch korrekt weitergeleitet werden:

```
moni-suse122-nagios:~ # tail -f /var/log/messages | grep ndo2db
[...]
Jan 21 16:49:37 moni-suse122-nagios ndo2db[1893]: Starting ndo2db ..done
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Successfully connected to
MySQL database
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Successfully connected to
MySQL database
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Error: queue send error.
Jan 21 16:49:37 moni-suse122-nagios ndo2db: last message repeated 39 times
[...]
```

Sie können das Problem gegebenenfalls lösen, indem Sie die Zahl der maximalen Nachrichten erhöhen. Fügen Sie dazu folgende Zeilen in der Datei `/etc/sysctl.conf` hinzu:

```
kernel.msgmax = 131072000
kernel.msgmnb = 131072000
kernel.msgmni = 65536000
```

Installation einer graphischen Auswertung mit PNP4Nagios

Das Addon PNP4Nagios speichert Messwerte von Nagios und bereitet sie graphisch auf. Im Folgenden zeigen wir Ihnen die Installation aus Quellcode. PNP4Nagios ist über Git verfügbar. Installieren Sie zunächst Git, um darauf zugreifen zu können:

```
root@moni-wheezy-nagsrc:~# aptitude install git
```

Jetzt können Sie sich eine Kopie der Quelle anlegen:

```
root@moni-wheezy-nagsrc:~# cd /usr/src/
root@moni-wheezy-nagsrc:/usr/src# git clone
git://pnp4nagios.git.sourceforge.net/gitroot/pnp4nagios/pnp4nagios
```

Anschließend müssen Sie nur noch die Version wählen, die Sie nutzen möchten. Wir verwenden hier die derzeit aktuelle Version 0.6.16:

```
root@moni-wheezy-nagsrc:/usr/src# cd pnp4nagios
root@moni-wheezy-nagsrc:/usr/src/pnp4nagios# git checkout 0.6.19
```

`git`: In der Versionierungsanwendung Git können Versionen mit Hilfe von sogenannten Tags markiert werden. Mit Hilfe des Kommandos `git tag` können Sie sich sich alle vorhandene Tags eines Repositories anzeigen lassen. Hiermit können Sie leicht prüfen, welche Versionen verfügbar sind.

PNP4Nagios speichert die Daten in sogenannten Round-Robin-Datenbanken (RRDs) und benötigt Perl und die Perl-Anbindung an diese RRDs sowie die PHP-GD-Erweiterung für den Webserver. Installieren Sie diese Notwendigkeiten zunächst auf folgende Weise:

```
root@moni-wheezy-nagsrc:/usr/src/pnp4nagios# aptitude install perl rrdtool librrds-perl
php5-gd
```

Damit haben Sie alle Voraussetzungen erfüllt und können die Anpassung an Ihr System durchführen lassen und anschließend die Übersetzung und Installation vornehmen:

```
root@moni-wheezy-nagsrc:/usr/src/pnp4nagios# ./configure
root@moni-wheezy-nagsrc:/usr/src/pnp4nagios# make all
root@moni-wheezy-nagsrc:/usr/src/pnp4nagios# make fullinstall
```

Bei der Option `fullinstall` installieren Sie dabei den NPCD (Nagios Perfdata C Daemon) mit. PNP4Nagios unterstützt eine ganze Reihe von Betriebsmodi (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/modes*). Der Modus mit NPCD wird vom PNP4Nagios-Projekt mit »Dies ist aus Nagios-Sicht die sauberste Art der Verarbeitung« beschrieben, weshalb wir ihn im Folgenden verwenden.

Für die Speicherung der Daten sind Parameter zur Auflösung und Aufbewahrungsdauer anzugeben. Kopieren Sie hier zunächst einfach die für Sie angelegte Beispieldatei um und nutzen Sie die hier vorgegebenen Werte:

```
root@moni-wheezy-nagsrc:/usr/src/pnp4nagios# cd /usr/local/pnp4nagios/etc/
root@moni-wheezy-nagsrc:/usr/local/pnp4nagios/etc#
root@moni-wheezy-nagsrc:/usr/local/pnp4nagios/etc# cp rra.cfg-sample rra.cfg
```

Um das auf diese Weise installierte PNP4Nagios an das Nagios anzubinden und die Verarbeitung von Performance-Daten überhaupt zu aktivieren, müssen Sie jetzt noch entsprechende Direktiven in der Hauptkonfigurationsdatei von Nagios (*/usr/local/nagios/etc/nagios.cfg*) setzen:

```
[...]
# PROCESS PERFORMANCE DATA OPTION
[...]
process_performance_data=1
[...]
# HOST AND SERVICE PERFORMANCE DATA FILES
[...]
host_perfdata_file=/usr/local/pnp4nagios/var/host-perfdata
service_perfdata_file=/usr/local/pnp4nagios/var/service-perfdata
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE TEMPLATES
[...]
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::$TIMET$\tHOSTNAME::
$HOSTNAME$\tSERVICEDESC::$SERVICEDESC$\tSERVICEPERFDATA::
$SERVICEPERFDATA$\tSERVICECHECKCOMMAND::$SERVICECHECKCOMMAND$\tHOSTSTATE::
$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$\tSERVICESTATE::
$SERVICESTATE$\tSERVICESTATETYPE::$SERVICESTATETYPE$
host_perfdata_file_template=DATATYPE::
HOSTPERFDATA\tTIMET::$TIMET$\tHOSTNAME::$HOSTNAME$\tHOSTPERFDATA::
$HOSTPERFDATA$\tHOSTCHECKCOMMAND::$HOSTCHECKCOMMAND$\tHOSTSTATE::
$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE MODES
[...]
host_perfdata_file_mode=a
service_perfdata_file_mode=a
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING INTERVAL
[...]
host_perfdata_file_processing_interval=30
service_perfdata_file_processing_interval=30
[...]
# HOST AND SERVICE PERFORMANCE DATA FILE PROCESSING COMMANDS
[...]
host_perfdata_file_processing_command=process-host-perfdata-file
service_perfdata_file_processing_command=process-service-perfdata-file
[...]
```

Hiermit richten Sie zunächst ein, dass Nagios die Performancedaten, wie in der Vorlage mit Makros spezifiziert (zu Makros siehe Rezept 4.3, *Befehle hinzufügen (command)*), in die angegebene Datei schreibt. Dabei geben die letzten beiden Direktiven die auszufüh-

renden Befehle `process-host-perfdata-file` und `process-service-perfdata-file` an (siehe Rezept 4.3, *Befehle hinzufügen (command)*). Richten Sie diese ein, indem sie die folgenden Definitionen in der Datei */usr/local/nagios/etc/objects/commands.cfg* entsprechend anpassen:

```
# 'process-host-perfdata' command definition
define command{
       command_name                            process-host-perfdata-file
       command_line                            /bin/mv /usr/local/pnp4nagios/var/ \
host-perfdata/usr/local/pnp4nagios/var/spool/host-perfdata.$TIMET$
}

# 'process-service-perfdata' command definition
define command{
       command_name                            process-service-perfdata-file
       command_line                            /bin/mv /usr/local/pnp4nagios/var/ \
service-perfdata/usr/local/pnp4nagios/var/spool/service-perfdata.$TIMET$
}
```

Auf diese Weise wurde eingerichtet, dass die von Icinga erstellten Dateien in regelmäßigen Abständen in das Verzeichnis */usr/local/pnp4nagios/var/spool* verschoben werden. Von dort wird sie dann der NPCD abrufen und verarbeiten. Dies ist über die folgende Konfigurationsoption in der Datei */usr/local/pnp4nagios/etc/npcd.cfg* voreingestellt:

```
perfdata_spool_dir = /usr/local/pnp4nagios/var/spool
```

Vor dem ersten Test verlangt die Weboberfläche von PNP4Nagios noch die Aktivierung des Moduls rewrite. Verwenden Sie dazu die folgende Anweisung:

```
root@moni-wheezy-nagsrc:~# a2enmod rewrite
```

Um die Konfiguration zu testen, starten Sie nun zunächst den NPCD und führen dann einen Neustart von Nagios und des Webservers durch, damit die geänderten Konfigurationen wirksam werden:

```
root@moni-wheezy-nagsrc:~# service npcd start
root@moni-wheezy-nagsrc:~# service apache2 restart
root@moni-wheezy-nagsrc:~# service nagios restart
```

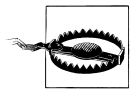
Rufen Sie jetzt *http://127.0.0.1/pnp4nagios* auf beziehungsweise verwenden Sie gegebenenfalls die jeweilige IP-Adresse. PNP4Nagios führt für Sie einen Test durch und präsentiert Ihnen hier die Ergebnisse. Wenn alles in Ordnung ist, fordert Sie die Seite auf, die für diesen Test genutzte Datei zu entfernen, damit Sie Zugriff auf die Oberfläche erhalten. Nehmen Sie hierzu eine Umbenennung vor, so dass Ihnen die Datei erhalten bleibt:

```
root@moni-wheezy-nagsrc:~# mv /usr/local/pnp4nagios/share/install.php
/usr/local/pnp4nagios/share/install.php.orig
```

Wenn Sie den URL jetzt erneut besuchen, wird Ihnen die Oberfläche von PNP4Nagios angezeigt. Bis die Graphen tatsächlich etwas darstellen, kann allerdings ein wenig Zeit vergehen, da hierzu ja zunächst entsprechende Messwerte von Nagios gesammelt und weitergeleitet werden müssen.

Stellen Sie abschließend sicher, dass der NPCD automatisch gestartet wird:

```
root@moni-wheezy-nagsrc:~# insserv npcd
```



Keine Graphen: Wenn PNP4Nagios keine Performancedaten findet, können auch keine Graphen generiert werden. Prüfen Sie bei fehlenden Graphen zuerst, ob von den entsprechenden Check-Plugins tatsächlich Performancedaten erzeugt werden beziehungsweise diese erst hinzugefügt werden müssen.

Damit haben Sie die Installation des Addons PNP4Nagios abgeschlossen. Per Standardeinstellung werden nun für jedes Gerät und jeden Dienst Performancedaten verarbeitet, sofern diese vom Plugin geliefert werden.

Integration von PNP4Nagios

Sie können und sollten sich jetzt noch entsprechend verlinkte Icons in der Nagios-Oberfläche anzeigen lassen. Fügen Sie zu diesem Zweck der Datei */usr/local/nagios/etc/objects/templates.cfg* (siehe Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*) die folgenden Definitionen (siehe Rezept 4.1, *Maschinen einbinden (host)* für die Definition von Maschinen, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* für die Definition von Services, sowie Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System* für die Verwendung des Vorlagensystems) hinzu und referenzieren Sie diese folgendermaßen in den Definitionen `generic-host` und `generic-service`:

```
[...]
# HOST TEMPLATES
[...]
define host{
       name                                    pnp-hst
       register                                0
       action_url                              /pnp4nagios/graph?host=$HOSTNAME$' \
class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=_HOST_
}

define host{
       name                                    generic-host
       use                                     pnp-hst
[...]
# SERVICE TEMPLATES
[...]
define service {
       name                                    pnp-svc
       register                                0
       action_url                              /pnp4nagios/graph?host=$HOSTNAME$&srv=$ \
SERVICEDESC$'class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=$SERVICEDESC$
}
[...]
define service{
       name                                    generic-service
       use                                     pnp-svc
[...]
```

Hierdurch werden entsprechende Graphiken als Icons für alle Geräte und Services angezeigt, mit der entsprechende PNP4Nagios-Seite verlinkt sind. Dies gilt dann eben aber auch für Dienste, die möglicherweise gar keine Performance-Daten liefern und zu denen deshalb auch keine Graphen vorliegen. Sie können sich jedoch über die aufgerufene Seite zumindest schnell zu den vorhandenen Graphen der jeweiligen Maschine weiterklicken.

Differenzierung: Hierbei werden entsprechende Icons für alle Geräte und Services definiert, auch für solche ohne Graphen. Sie haben aber alternativ auch die Möglichkeit, die Vorlage `pnp-svc` nur bei den Diensten einzubinden, für die Sie Graphen haben möchten.

Bei PNP4Nagios ist hier vorgesehen, dass Ihnen die jeweiligen Graphen beim Darüberfahren mit der Maus angezeigt werden. Diese Funktionalität nutzt JavaScript, dass Sie mit der folgenden Kopier-Operation einbinden können:

```
root@moni-wheezy-nagsrc:~# cp /usr/src/pnp4nagios/contrib/ssi/status-header.ssi
/usr/local/nagios/share/ssi/
```

Damit Nagios die neuen Konfigurationsoptionen berücksichtigt, müssen Sie es neu starten:

```
root@moni-wheezy-nagsrc:~# service nagios restart
```

Installation des Visualisierungsaddons NagVis

Eine weitere Möglichkeit, Ergebnisse der Messungen aufzubereiten, bietet das Addon NagVis. Mit diesem können Sie Symbole für Status von Geräten und Services auf Hintergrundbildern wie Karten oder Fotos platzieren.

Um NagVis zu installieren, besorgen Sie sich zunächst über Git den Quellcode:

```
root@moni-wheezy-nagsrc:~# cd /usr/src/
root@moni-wheezy-nagsrc:/usr/src# git clone
git://nagvis.git.sourceforge.net/gitroot/nagvis/nagvis
```

Wählen Sie dann die Version, hier die derzeit aktuelle Version 1.7.4:

`git`: In der Versionierungsanwendung Git können Versionen mittels sogenannter Tags markiert werden. Mit Hilfe des Kommandos `git tag` können Sie sich sich alle vorhandenen Tags eines Repositories anzeigen lassen. Auf diese Weise können Sie leicht prüfen, welche Versionen verfügbar sind.

```
root@moni-wheezy-nagsrc:/usr/src/nagvis# git checkout nagvis-1.7.4
```

NagVis baut auf der Graphbibliothek GraphViz auf und benötigt die PHP5-Anbindung an MySQL. Installieren Sie diese deshalb zunächst über die Paketverwaltung:

```
root@moni-wheezy-nagsrc:/usr/src/nagvis# aptitude install graphviz php5-mysql php5-sqlite
```

Für die Installation wird bei NagVis dann ein Script verwendet. Starten Sie dieses wie folgt und verwenden Sie bei allen Fragen die Vorgabewerte, mit Ausnahme der im Folgenden genannten:

```
root@moni-wheezy-nagsrc:/usr/src/nagvis# ./install.sh
[...]
| Checking Backends. (Available: mklivestatus,ndo2db,ido2db,merlinmy) |
| Do you want to use backend mklivestatus? [y]: n
| Do you want to use backend ndo2db? [n]: y
| Do you want to use backend ido2db? [n]:
| Do you want to use backend merlinmy? [n]:
[...]
```

Beachten Sie die dabei die ausgegebenen Zugangsdaten, die Sie für den Zugriff auf die Weboberfläche benötigen (`admin`:`admin`). Alternativ können Sie die Werte über die Kommandozeile übergeben und so einen Teil der Fragen umgehen:

```
root@moni-wheezy-nagsrc:/usr/src/nagvis# ./install.sh -n /usr/local/nagios
-p /usr/local/nagvis -b ndo2db -u www-data -g www-data -w /etc/apache2/conf.d -a y
```

NagVis bezieht seine Daten aus der von den NDOUtils befüllten Datenbank. Hinterlegen Sie deshalb die Zugangsdaten zu dieser in der Datei */usr/local/nagvis/etc/nagvis.ini.php*:

```
dbhost="localhost"
dbport=3306
dbname="nagios"
dbuser="sql_nagios"
dbpass="DBUSERPW"
dbprefix="nagios_"
dbinstancename="default"
maxtimewithoutupdate=180
htmlcgi="/nagios/cgi-bin"
```


Starten Sie jetzt den Webserver neu, damit dieser die hinzugefügte Konfiguration einliest:

```
root@moni-wheezy-nagsrc:~# service apache2 restart
```

Sie können sich nun unter *http://127.0.0.1/nagvis* oder der jeweiligen IP-Adresse mit den oben ausgegebenen Anmeldeinformationen (hier Benutzername `admin` mit Passwort `admin`) anmelden. Ändern Sie dieses Passwort am besten gleich im Nutzermenu rechts oben. Für den weiteren Umgang mit PNP4Nagios und NagVis lesen Sie bitte die Rezepte zur Datenvisualisierung (Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*).

Diskussion

Die Installation aus den Quellen ist im Vergleich zur Installation über das Paketmanagementsystem relativ aufwendig. Dafür bieten sich Möglichkeiten, die Installation anzupassen und etwa nicht benötigten Plugins wegzulassen. Sie können diese unterschiedlichen Installationsmethoden allerdings nicht ohne Weiteres mischen, also etwa zusätzliche im Paketmanagement-System enthaltene Plugins nachinstallieren.

Alternativ zu dieser Installation aus den Quellen haben wir für Sie die paketbasierte Installation von Icinga unter verschiedenen Distributionen beschrieben:

- Rezept 1.10, *Nagios unter Debian installieren*
- Rezept 1.11, *Nagios unter Ubuntu installieren*
- Rezept 1.12, *Nagios unter SLES|openSUSE installieren*
- Rezept 1.13, *Nagios unter RHEL|CentOS|Fedora installieren*

Nagios nutzt bei dieser Installation die in Tabelle 1-23 aufgeführten Speicherorte. Die Ablageorte für diverse Komponenten und Konfigurationsdateien sind für Sie von Bedeutung, damit Sie schnell die richtige Stelle für gewünschte Änderungen finden können.

Tabelle 1-23: Installationsorte wichtiger Komponenten von Nagios

Komponente	Ort
Programm	/usr/local/nagios/bin/nagios
Haupt-Konfigurationsdatei	/usr/local/nagios/etc/nagios.cfg
Plugins	/usr/local/nagios/libexec/
Logdatei	/usr/local/nagios/var/nagios.log
Webserver-Konfiguration	/etc/apache2/conf.d/nagios.conf
Webseiten	/usr/local/nagios/share/

Das Addon NDOUtils nutzt die in Tabelle 1-24 aufgeführten Speicherorte.

Tabelle 1-24: Installationsorte wichtiger Komponenten der NDOUtils

Komponente	Ort
Konfigurationsdatei ndo2db	/usr/local/nagios/etc/ndo2db.cfg
Konfigurationsdatei ndomod	/usr/local/nagios/etc/ndomod.cfg
Debug-Logs	/usr/local/nagios/var/*.debug

PNP4Nagios nutzt die in Tabelle 1-25 genannten Orte.

Tabelle 1-25: Installationsorte wichtiger Komponenten von PNP4Nagios

Komponente	Ort
Haupt-Konfigurationsdatei	/usr/local/pnp4nagios/etc/config.php
NPCD-Konfigurationsdatei	/usr/local/pnp4nagios/etc/npcd.cfg
Graphendefinitionen und Vorlagen	/usr/local/pnp4nagios/share/
Webserver-Konfigurationsdatei	/etc/apache2/conf.d/pnp4nagios.conf

Das Addon NagVis verwendet die in Tabelle 1-26 aufgeführten Speicherorte.

Tabelle 1-26: Installationsorte wichtiger Komponenten von NagVis

Komponente	Ort
Webserver-Konfiguration	/etc/apache2/conf.d/nagvis.conf
Webseiten	/usr/local/nagvis/share/

Auf die Nagios-Plugins werden wir in den Rezepten in den Kapiteln 5 und 6 genauer eingehen. Das Addon NRPE (Nagios Remote Plugin Executor) werden wir in dem Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)* erläutern.

Siehe auch

- Rezept mit Entscheidungshilfe zur Installation: Rezept 1.1, *Entscheidungshilfe für die Installation*
- Webseite des Nagios-Projekts: *http://www.nagios.org/*
- Download-Seite von Nagios auf Sourceforge: *http://prdownloads.sourceforge.net/sourceforge/nagios/*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Download-Seite der Nagios-Plugins: *http://sourceforge.net/projects/nagiosplug/files/nagiosplug/*
- Quickstart-Dokumentation zur Installation: *http://nagios.sourceforge.net/docs/3_0/quickstart.html*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Dokumentation der Betriebsmodi von PNP4Nagios: *http://docs.pnp4nagios.org/de/pnp-0.6/modes*
- Webseite von NagVis: *http://www.nagvis.org/*
- Webseite des Debian-Projekts: *http://www.debian.org/*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Kapitel zur Konfiguration von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere das Rezept 4.3, *Befehle hinzufügen (command)*, Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* sowie Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Kapitel zur Datenvisualisierung mit PNP4Nagios und NagVis: Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*
- Rezepte zur Paketbasierten Installation von Nagios: Rezept 1.10, *Nagios unter Debian installieren*, Rezept 1.11, *Nagios unter Ubuntu installieren*, Rezept 1.12, *Nagios unter SLES|openSUSE installieren* und Rezept 1.13, *Nagios unter RHEL|CentOS|Fedora installieren*
- Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*

- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezepte zur Installation von NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

1.10 Nagios unter Debian installieren

Problem

Sie möchten Nagios unter Debian paketbasiert installieren.

Lösung

Zunächst beschreiben wir die Installation von Nagios selbst. Darauf aufbauend können Sie in wenigen weiteren Schritten das Addon NDOUtils (Nagios Data Out Utilities) installieren, um die Daten für weitere Addons (insbesondere das später folgende NagVis) in einer Datenbank vorzuhalten. Anschließend werden wir die Installation von PNP4Nagios, einem Add-on zur automatischen Weiterverarbeitung und graphischen Auswertung von Performancedaten, erläutern. Abschließend zeigen wir Ihnen noch die Installation von NagVis, einem Addon, das die Datenbankanbindung über die NDOUtils verwendet und Ihnen ermöglicht, Symbole für Geräte und Dienste auf auf einem beliebigen Hintergrund (etwa einer Karte, einem Grundriss oder einem Foto) anzuordnen.

Der Codename der aktuellen stabilen Version von Debian ist Wheezy (Debian 7). In Wheezyist Nagios als fertiges Paket ab der Version 3.4.1 enthalten.

`apt-get & aptitude`: Als Benutzerschnittstellen zum eigentlich Paketmanager `dpkg` stehen Ihnen sowohl `apt-get` als auch `aptitude` zur Verfügung, die funktional äqqivalent sein sollten. Wir verwenden in diesem Buch aus Gründen der Einheitlichkeit stets `aptitude`.

Basierend auf einer minimalen 64bit-SSH-Server-Installation von Debian können Sie eine laufende Nagios-Umgebung wie folgt einrichten:

Deinstallation: Sofern Sie eine Deinstallation erwarten und dabei sichergehen möchten, dass alle mitinstallierten Pakete wieder entfernt werden, sollten Sie sich an dieser Stelle die Liste der auf Ihrem System mitinstallierten Pakete durch Kopieren notieren. Der Hintergrund dabei ist der, dass bei der Deinstallation über Paketmanager regelmäßig ganze Pakete auf dem System verbleiben (genau dann nämlich, wenn diese in anderen installierten Paketen als optionale Abhängigkeit gelistet sind).

```
root@moni-wheezy-nagios:~# aptitude install nagios3
```

Bestätigen Sie die Installation der aufgeführten Pakete. Während der Installation wird der Name der Samba-Arbeitgruppe (bestätigen Sie hier im Zweifel einfach die Vorgabe) und ein Passwort für den Nagios Administratorzugang `nagiosadmin` abgefragt. Wir werden das hierbei zur Authentifizierung verwendete `http-auth in` Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche* genauer erläutern.

Nagios und Apache laufen bereits nach der Installation. Jetzt müssen Sie nur noch sicherstellen, dass beide Dienste nach einem Neustart automatisch gestartet werden. Mit folgender Anweisung überprüfen Sie das Vorhandensein der Links zu den entsprechenden Startscripten in den dafür vorgesehenen Unterverzeichnissen von */etc/rc?.d* :

```
root@moni-wheezy-nagios:~# find /etc/rc?.d/ -iname "*nagios3"
/etc/rc0.d/K01nagios3
/etc/rc1.d/K01nagios3
/etc/rc2.d/S17nagios3
/etc/rc3.d/S17nagios3
/etc/rc4.d/S17nagios3
/etc/rc5.d/S17nagios3
/etc/rc6.d/K01nagios3
root@moni-wheezy-nagios:~# find /etc/rc?.d/ -iname "*apache2"
/etc/rc0.d/K01apache2
/etc/rc1.d/K01apache2
/etc/rc2.d/S16apache2
/etc/rc3.d/S16apache2
/etc/rc4.d/S16apache2
/etc/rc5.d/S16apache2
/etc/rc6.d/K01apache2
```

Die Nummer im Namen des Ordners gibt dabei den Runlevel an, und der erste Buchstabe im Namen des Verweises spezifiziert, ob der Dienst in diesem Runlevel gestartet (`S` für `start`) oder gestoppt (`K` für `kill`) werden soll.

`insserv`: Wenn ein Service automatisch gestartet werden soll, können Sie dies mit dem Kommando `insserv <service>` konfigurieren.

Konfigurationsschritte: Die adäquaten Schritte für eine manuelle, angepasste Konfiguration können Sie der quellcodebasierenden Nagios-Installation entnehmen (siehe Rezept 1.9, *Nagios aus den Quellen installieren*).

Nach der Installation führt Nagios bereits eine Überwachung des lokalen Systems durch. Sie können das Monitoringsystem über *http://127.0.0.1/nagios3/* oder die systemspezifische IP-Adresse erreichen. Dabei wird das für den Nutzer `nagiosadmin` gesetzte Passwort benötigt.

Oberflächentest von der Kommandozeile: Falls der entfernte Webzugriff auf Ihr System (noch) nicht möglich ist, beispielsweise aufgrund einer Firewall, können Sie auch über *http://127.0.0.1/nagios3/* auf Ihr frisch installiertes Nagios zugreifen. Wenn Sie über keine graphische Oberfläche und bzw. oder keinen lokalen Webbrowser verfügen, können Sie die Erreichbarkeit der Oberfläche durch folgende Anweisung mit dem Kommandozeilen-Werkzeug `telnet` testen:

```
moni-wheezy-nagios:~ # telnet localhost 80
[...]
GET /nagios3
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en"> <head>
<title>Authentication required!</title>
[...]
```

Nun können Sie mit der Installation der NDOUtils (Nagios Data Out Utilities) für den Anschluss von Nagios an einen Datenbankserver fortfahren. In diesem werden später die gesammelten Daten abgelegt. Debian Wheezy enthältdas Paket `ndoutils-nagios3-mysql` ab der Version 1.4b9. Zur Installation rufen wir den Paketmanager Aptitude noch einmal wie folgt auf:

```
root@moni-wheezy-nagios:~# aptitude install ndoutils-nagios3-mysql
```

Der MySQL-Datenbankserver wird hierbei als Abhängigkeit automatisch mit-installiert. Während der Installation wird ein Passwort für den Benutzer `root` des Datenbankservers abgefragt.

MySQL-Passwörter: Wenn Sie das hier gesetzte Passwort für den `root`-Nutzer von MySQL ändern wollen, funktioniert dies folgendermaßen:

```
root@moni-wheezy-nagios:~# mysqladmin -u root -p PASSWORD
```

Um das Passwort eines nicht-administrativen MySQL-Benutzers zu ändern, können Sie sich von der Kommandozeile aus als Benutzer `root` auf dem Server anmelden und das nachfolgend angeführte Kommando nutzen:

```
root@moni-wheezy-nagios:~# mysql -u root -p
mysql> use mysql;
mysql> update user set Password=DBUSERPW('NEWDBUSERPW') WHERE User='DBUSER';
mysql> exit
```

Anschließend startet auf Wunsch eine geführte Installation der NDOUtils. Wir empfehlen Ihnen, den Installationsassistenten zu nutzen. In den nächsten Schritten geben Sie das soeben gesetzte Root-Passwort für MySQL und dann das Passwort für den einzurichtenden MySQL-Benutzer `ndoutils` ein.

Die NDOUtils werden dabei noch nicht mit Nagios verknüpft und wohl deshalb auch nicht automatisch gestartet. Zunächst müssen Sie Nagios also anweisen, die Daten an das für diesen Zweck vorgesehene und mitinstallierte Modul `ndomod` zu übergeben. Hierzu tragen Sie das Modul in der Datei */etc/nagios3/nagios.cfg* als sogenanntes Broker-Modul ein:

```
[...]
# EVENT BROKER MODULE(S)
[...]
broker_module=/usr/lib/ndoutils/ndomod-mysql-3x.o config_file=/etc/nagios3/ndomod.cfg
[...]
```

Dann müssen Sie die NDOUtils selbst noch aktivieren, indem Sie in der Datei */etc/default/ndoutils* die entsprechende Option setzen:

```
ENABLE_NDOUTILS=1
```

Abschließend können Sie NDOUtils starten und einen Neustart von Nagios durchführen, damit die neue Konfiguration aktiv wird:

```
root@moni-wheezy-nagios:~# service ndoutils start
root@moni-wheezy-nagios:~# service nagios3 restart
```

Damit laufen sowohl Nagios als auch NDOUtils und die Daten werden in die angelegte Datenbank geschrieben. Prüfen Sie mit folgenden Anweisungen noch, ob die NDOUtils und der mit diesen installierte MySQL-Server bei einem Neustart automatisch gestartet werden:

```
root@moni-wheezy-nagios:~# find /etc/rc?.d/ -iname "*mysql"
/etc/rc0.d/K02mysql
/etc/rc1.d/K02mysql
/etc/rc2.d/S17mysql
/etc/rc3.d/S17mysql
/etc/rc4.d/S17mysql
/etc/rc5.d/S17mysql
/etc/rc6.d/K02mysql
root@moni-wheezy-nagios:~# find /etc/rc?.d/ -iname "*ndoutils"
/etc/rc0.d/K01ndoutils
/etc/rc1.d/K01ndoutils
/etc/rc2.d/S17ndoutils
/etc/rc3.d/S17ndoutils
/etc/rc4.d/S17ndoutils
/etc/rc5.d/S17ndoutils
/etc/rc6.d/K01ndoutils
```

Abschließend vergewissern Sie sich, dass tatsächlich Daten in die Datenbank geschrieben werden. Dazu lassen Sie sich den Eintrag `status_update_time` in der Tabelle `nagios_programstatus` anzeigen, anhand dessen Sie ablesen können, ob die Nagios-Datenbank kürzlich aktualisiert wurde:

```
root@moni-wheezy-nagios:~# mysql -u root -p
mysql> USE ndoutils;
mysql> SELECT status_update_time FROM nagios_programstatus;
```

```
+---------------------+
| status_update_time  |
+---------------------+
| 2012-12-02 18:59:14 |
+---------------------+
1 row in set (0.00 sec)
mysql> exit
```

Damit wissen Sie, dass Nagios und die NDOUtils laufen und dass Daten in die neu angelegte Datenbank geschrieben werden.

queue send error: In den NDOUtils ab Version 1.5.2 werden Nachrichten nur dann in die Warteschlage weitergeleitet, wenn der Kernel diese nicht annimmt. Falls Sie alles einwandfrei konfiguriert haben und trotzdem keine Daten in Ihre Datenbank geschrieben werden, prüfen Sie, ob die Daten auch korrekt weitergeleitet werden:

```
moni-suse122-nagios:~ # tail -f /var/log/messages | grep ndo2db
[...]
Jan 21 16:49:37 moni-suse122-nagios ndo2db[1893]: Starting ndo2db ..done
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Successfully connected to
MySQL database
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Successfully connected to
MySQL database
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Error: queue send error.
Jan 21 16:49:37 moni-suse122-nagios ndo2db: last message repeated 39 times
[...]
```

Sie können das Problem lösen, indem Sie die Zahl der maximalen Nachrichten erhöhen. Fügen Sie dazu folgende Zeilen in der Datei `/etc/sysctl.conf` hinzu:

```
kernel.msgmax = 131072000
kernel.msgmnb = 131072000
kernel.msgmni = 65536000
```


Installation einer graphischen Auswertung mit PNP4Nagios

In Wheezy haben Sie die Möglichkeit, PNP4Nagios als Addon zur Analyse und Darstellung von Performancedaten über die Paketverwaltung zu installieren. PNP4Nagios ist dabei abVersion 0.6 enthalten und sowohl für Nagios als auch für Icinga verfügbar. Sie können das entsprechende Paket `pnp4nagios` mit folgender Anweisung installieren:

```
root@moni-wheezy-nagios:~# aptitude install pnp4nagios
```

Paketabhängigkeiten: Lassen Sie eventuell entstehe Konflikte zwischen Abhängigkeiten einzelner Pakete durch den Paketmanager `aptitude` auflösen. Wählen Sie im Zweifel die zuerst angebotene Lösung.

In PNP4Nagios werden die gesammelten Daten in sogenannten Round-Robin-Datenbanken (RRD) gespeichert. Die dafür erforderlichen Programme werden über das Paket `rrdtool` automatisch mitinstalliert.

PNP4Nagios unterstützt eine ganze Reihe von Betriebsmodi (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/modes*). Der Modus mit NPCD (Nagios Perfdata C Daemon) wird vom PNP4Nagios-Projekt mit »Dies ist aus Nagios-Sicht die sauberste Art der Verarbeitung« beschrieben. Alternativ können Sie Synchronous Mode installieren, der etwas einfacher zu konfigurieren ist. Im Folgenden werden wir die Installation des »Synchronous Mode« beschreiben. Für die Installation des »Bulk mode mit NPCD« orientieren Sie sich an dem entsprechenden Abschnitt im Rezept 1.5, *Icinga unter Debian installieren*.

Als Nächstes müssen Sie Nagios so konfigurieren, dass zusätzlich zu den reinen Statusinformationen überhaupt eine Verarbeitung der sogenannten Performancedaten stattfindet. Dazu modifizieren Sie die folgenden Zeilen der Datei */etc/nagios3/nagios.cfg*:

```
[...]
# PROCESS PERFORMANCE DATA OPTION
[...]
process_performance_data=1
[...]
# HOST AND SERVICE PERFORMANCE DATA PROCESSING COMMANDS
[...]
host_perfdata_command=process-host-perfdata
service_perfdata_command=process-service-perfdata
[...]
```

Jetzt müssen Sie noch die soeben referenzierten Befehle `process-host-perfdata` und `process-service-perfdata` in der Datei */etc/nagios3/commands.cfg* folgendermaßen anpassen (zu Kommandodefinitionen siehe auch Rezept 4.3, *Befehle hinzufügen (command)*):

```
# 'process-host-perfdata' command definition
define command {
       command_name                    process-host-perfdata
       command_line                    /usr/bin/perl /usr/lib/pnp4nagios/ \
libexec/process_perfdata.pl -d HOSTPERFDATA
}

# 'process-service-perfdata' command definition
define command {
       command_name                    process-service-perfdata
       command_line                    /usr/bin/perl /usr/lib/pnp4nagios/ \
libexec/process_perfdata.pl
}
```

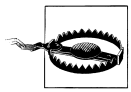
Nun starten Sie den Apache-Webserver und Icinga neu:

```
root@moni-wheezy-nagios:~# service apache2 restart
root@moni-wheezy-nagios:~# service nagios3 restart
```

Nach der jeweils ersten Messung wird im Verzeichnis */var/lib/pnp4nagios/perfdata* ein Unterverzeichnis für jedes Gerät angelegt. In diesem wird für jeden Service dieses Gerätes eine RRD-Datenbank und eine entsprechende XML-Datei erstellt. Überprüfen Sie dies gegebenenfalls nach einigen Minuten, indem Sie sich die Dateien im entsprechenden Ordner *localhost* anzeigen lassen:

```
root@moni-wheezy-nagios:~# ls /var/lib/pnp4nagios/perfdata/localhost/
Current_Load.rrd Current_Users.rrd Disk_Space.rrd _HOST_.rrd HTTP.rrd SSH.rrd
Current_Load.xml Current_Users.xml Disk_Space.xml _HOST_.xml HTTP.xml SSH.xml
```

Sie können PNP4Nagios unter *http://127.0.0.1/pnp4nagios* und alternativ auch unter der systemeigenen IP-Adresse erreichen. Dabei nutzen Sie die unter Nagios gesetzten Zugangsdaten. Sobald die ersten Messungen durchgeführt sind, werden nach einigen Minuten entsprechende Graphen angezeigt. Wundern Sie sich dabei nicht über anfangs leere Graphen. Per Standardeinstellung werden nun für jedes Gerät und jeden Dienst Performancedaten verarbeitet, sofern diese vom Plugin geliefert werden.



Keine Graphen: Wenn PNP4Nagios keine Performancedaten findet, können auch keine Graphen generiert werden. Prüfen Sie bei fehlenden Graphen zuerst, ob von den entsprechenden Check-Plugins tatsächlich Performancedaten erzeugt werden.

Integration von PNP4Nagios mit Nagios

Damit Sie von der Nagios Weboberfläche aus direkt auf die Graphen zugreifen können, sollten Sie die Graphen direkt mit der Anzeige von Geräten und Diensten verzahnen. Legen Sie sich zunächst wie nachfolgend beschrieben eine Vorlage in dem dafür vorgesehenen Verzeichnis (siehe hierzu Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*) */etc/nagios3/conf.d* an, etwa als *pnptemplate.cfg*. Wie diese Konfigurationsoptionen zusammenhängen, wird in den folgenden Rezepten erläutert:

- Rezept 4.1, *Maschinen einbinden (host)* für die Definition von Maschinen
- Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* für die Definition von Services
- Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System* für die Verwendung des Vorlagensystems

In die erstellte Datei schreiben Sie dann die folgenden Konfigurationsanweisungen:

```
define host {
       name pnp-hst
       register                                0
       action_url                              /pnp4nagios/graph?host=$HOSTNAME$' \
class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=_HOST_
}

define service {
       name                                    pnp-svc
       register                                0
       action_url                              /pnp4nagios/graph?host=$HOSTNAME$&srv=$ \
SERVICEDESC$' class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=$SERVICEDESC$
}
```

Anschließend referenzieren Sie diese Definitionen dann in den vorhandenen Definitionen `generic-host` in der Datei */etc/nagios3/conf.d/generic-host_nagios2.cfg*

```
define host {
        name                                    generic-host
        use                                     pnp-hst
```

und die für `generic-service` in der Datei */etc/nagios3/conf.d/generic-service_nagios2.cfg*:

```
define service {
        name                                    generic-service
        use                                     pnp-svc
```

Graph-Integration: Hierbei werden entsprechende Icons für alle Geräte und Services definiert, auch für solche ohne Graphen. Sie haben aber alternativ auch Möglichkeit, die Vorlage `pnp-svc` nur bei den Maschinen bzw. Diensten einzubinden, für die Sie Graphen haben oder anzeigen möchten.

Die oben als Vorlage definierte Konfigurationsoption `action_url` wird jeder Maschine und jedem Dienst ein Graphen-Symbol hinzufügen. Allerdings erst nach dem nächsten Neustart von Nagios, vor dem Sie aber noch eine weitere Anpassung vornehmen sollten. Sie können sich die Graphen nämlich bereits bei einem Darüberfahren mit der Maus (`mouse-over`) anzeigen lassen. Kopieren Sie hierzu wie nachfolgend angeführt die als Vorlage gelieferte Konfigurationsdatei *status-header.ssi* in das Verzeichnis */usr/share/icinga/htdocs/ssi/* und passen Sie die Rechte an:

```
root@moni-wheezy-nagios:~# cp /usr/share/doc/pnp4nagios/examples/ssi/status-header.ssi
/usr/share/nagios3/htdocs/ssi/
root@moni-wheezy-nagios:~# chown 644 /usr/share/nagios3/htdocs/ssi/status-header.ssi
```

Starten Sie nun Icinga neu, damit die geänderten Dateien eingelesen werden:

```
root@moni-wheezy-nagios:~# service nagios3 restart
```

Damit weist die Nagios-Oberfläche nun Symbole für die Graphen auf. Beim Darüberfahren mit der Maus sehen Sie eine Vorschau, durch Klick gelangen Sie auf die Seite von PNP4Nagios mit dem entsprechenden Graphen.

Installation des Visalisierungsaddons NagVis

Mit Hilfe des Addons NagVis können Sie Symbole für Maschinen und Services beliebig anordnen. Durch die Wahl eines entsprechenden Hintergrundes erhalten die Symbole dadurch automatisch einen Mehrwert, etwa durch die Platzierung auf Fotos oder Karten.

In Wheezy ist NagVis ab Version 1.6.6 enthalten. Es lässt sich sowohl für Nagios als auch für Icinga als Addon installieren. Sie können das Paket `nagvis` mit nachfolgend angeführter Anweisung installieren. Des Weiteren benötigen Sie das Paket `php5-mysql`:

```
root@moni-wheezy-nagios:~# aptitude install php5-mysql nagvis
```

Die bei der Installation angebotenen Pakete können Sie bestätigen. In einem anschließenden Dialog wird das von NagVis zu verwendende Monitoring-System abfragt. Bestätigen Sie hier Nagios.

Bei der Installation wurde bereits ein administrativer Zugang `admin` eingerichtet, dessen Passwort mit `admin` vorkonfiguriert ist. Sie sollten das Passwort bei Ihrer ersten Anmeldung ändern, doch hinterlegen Sie zunächst die Zugangsdaten zu der oben über IDO2DB eingebundenen Datenbank in der Datei */etc/nagvis/nagvis.ini.php*:

```
backend="ndomy_1"
[...]
dbhost="localhost"
dbport=3306
dbname="ndoutils"
dbuser="ndoutils"
dbpass="DBUSERPW"
dbprefix="nagios_"
dbinstancename="default"
```

Sie können sich jetzt mit den oben genannten Zugangsdaten anmelden. Sie sollten als Erstes das vorgegebene Administrator Passwort für `admin` ändern. Ansonsten ist die Basis-Einrichtung von NagVis damit komplett. Für den weiteren Umgang mit PNP4Nagios und NagVis lesen Sie bitte die Rezepte zur Datenvisualisierung (Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*).

NagVis-Beispiele: Sie können sich optional mit dem Debian-Paket `nagvis-demos` zusätzlich diverse Anwendungsbeispiele installieren.

Diskussion

Die Ablageorte für diverse Komponenten und Konfigurationsdateien sind für Sie von Bedeutung, damit Sie schnell die richtige Stelle für gewünschte Änderungen finden können. Der Tabelle 1-27 können Sie zunächst entnehmen, an welchen Speicherorten die einzelnen Komponenten der Nagios-Installation hier abgelegt wurden.

Tabelle 1-27: Ablageorte der Komponenten von Nagios

Komponente	Pfad
Programm	/usr/sbin/nagios3
Haupt-Konfigurationsdatei	/etc/nagios3/nagios.cfg
Plugins	/usr/lib/nagios/plugins
Logdatei	/var/log/nagios3/nagios.log
Webserver-Konfiguration	/etc/apache2/conf.d/nagios3.conf
Webseiten	/usr/share/nagios3/

Von den NDOUtils werden die in Tabelle 1-28 aufgeführten Speicherorte genutzt:

Tabelle 1-28: Pfade der wichtigsten Dateien von NDOUtils

Komponente	Pfad
Konfigurationsdatei ndo2db	/etc/nagios3/ndo2db.cfg
Konfigurationsdatei ndomod	/etc/nagios3/ndomod.cfg
Debug-Logs	/var/log/nagios3/*.debug

Das Addon PNP4Nagios hat hier die in Tabelle 1-29 aufgezeigten Installationsorte verwendet.

Tabelle 1-29: Pfade zu den Komponenten von PNP4Nagios

Komponente	Pfad
Konfigurationsdateien	/etc/pnp4nagios/
Logdateien	/var/log/pnp4nagios/
Webserver-Konfiguration	/etc/apache2/conf.d/pnp4nagios.conf
Webseiten	/usr/share/pnp4nagios/html/
Script zur Verarbeitung	/usr/lib/pnp4nagios/libexec/process_perfdata.pl

Das Addon Nagvis hat hier die in Tabelle 1-30 aufgezeigten Installationsorte verwendet.

Tabelle 1-30: Pfade zu den Komponenten von NagVis

Komponente	Pfad
Haupt-Konfigurationsdatei	/etc/nagvis/nagvis.ini.php
Webserver-Konfiguration	/etc/apache2/conf.d/nagvis.conf
Webseiten	/usr/share/nagvis/share/

Bei der Installation wurden automatisch die Plugins aus dem Paket `nagios-plugins` installiert, das die Pakete `nagios-plugins-basic` und `nagios-plugins-standard` umfasst. Auf die enthaltenen Plugins werden wir in den Rezepten der Kapitel 5 und 6 genauer eingehen. Für die Überwachung von System-Updates können Sie das in den Nagios-Plugins enthaltene Plugin `check_apt` verwenden (siehe Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*). Des Weiteren steht das Paket `nagios-snmp-plugins` zur Verfügung.

Unter Debian wird für die Plugins dabei ein eigenes Konfigurationsverzeichnis unter */etc/nagios-plugins* erstellt, das vorgefertigte Kommandodefinitionen für die Einbindung in Nagios zur Verfügung stellt (siehe Rezept 4.3, *Befehle hinzufügen (command)*).

Als weiteres Addon können Sie NRPE (Nagios Remote Plugin Executor) installieren (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*).

Siehe auch

- Rezept mit Entscheidungshilfe zur Installation: Rezept 1.1, *Entscheidungshilfe für die Installation*
- Webseite des Nagios-Projekts: *http://www.nagios.org//*
- Webseite von Nagiosplugins: *http://nagiosplugins.org//*
- Webseite von NagiosGrapher: *http://www.netways.de/de/produkte/nagios_addons/nagiosgrapher/*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Webseite von NagVis: *http://www.nagvis.org/*
- Webseite des Debian-Projekts: *http://www.debian.org/*
- Rezept zur quellenbasierten Installation von Nagios: Rezept 1.9, *Nagios aus den Quellen installieren*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zum Aufbau und strukturierter Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Kapitel zur Konfiguration von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere Rezept 4.3, *Befehle hinzufügen (command)*, Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* sowie Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Kapitel zur Datenvisualisierung mit PNP4Nagios und NagVis: Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*
- Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezept zur Überwachung von Systemupdates: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*
- Rezepte zur Installation von NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

1.11 Nagios unter Ubuntu installieren

Problem

Sie möchten Nagios unter Ubuntu paketbasiert installieren.

Lösung

Wir zeigen zunächst die Installation von Nagios und darauf aufbauend die Installation und Integration mit dem Datenbank-Addon NDOUtils (Nagios Data Out Utilities) zum Vorhalten der Daten für weitere Addons in einer Datenbank. Anschließend werden wir die Installation von PNP4Nagios, einer Erweiterung zur automatischen Weiterverarbeitung und graphischen Auswertung von Performancedaten, erläutern.

Die Installation von NagVis, einem Addon, das die Datenbankanbindung über die NDOUtils verwendet, ist paketbasiert unter Ubuntu zur Zeit noch nicht möglich. Das in Ubuntu 12.04 (Precise Pangolin) mitgelieferte Paket war in unserer Referenzinstallation nicht funktionsfähig. Wir vermuten, dass dieser Fehler mit Erscheinen dieses Buches behoben sein wird. Daher empfehlen wir Ihnen, dass Sie sich bei der paketbasierten Installation von NagVis an dem entsprechenden Abschnitt des Installationsrezeptes für Debian orientieren. Anderenfalls müssen Sie mit der sourcebasierte Installation vorliebnehmen.

In der LTS-Version (Long Term Support) Ubuntu 12.04 (Codename Precise Pangolin) ist Nagios ab der Version 3.2.3 enthalten.

`apt-get & aptitude`: Als Benutzerschnittstellen zum eigentlich Paketmanager `dpkg` stehen Ihnen sowohl `apt-get` als auch `aptitude` zur Verfügung, die funktional äquivalent sein sollten. Wir verwenden in diesem Buch aus Gründen der Einheitlichkeit stets `aptitude`.

Deinstallation: Sofern Sie eine Deinstallation erwarten und dabei sichergehen möchten, dass alle mitinstallierten Pakete wieder entfernt werden, sollten Sie sich an dieser Stelle die Liste der auf Ihrem System mitinstallierten Pakete kopieren. Der Hintergrund dabei ist der, dass bei der Deinstallation über Paketmanager regelmäßig ganze Pakete auf dem System verbleiben (dann nämlich, wenn diese in anderen installierten Paketen als optionale Abhängigkeit gelistet sind).

Basierend auf einer minimalen 64Bit-Server-Installation von Precise Pangolin können Sie mit folgendem Befehl eine laufende Nagios-Umgebung einrichten:

```
root@moni-pangolin-nagios:~# aptitude install nagios3
```

Während der Installation erfolgt dabei zunächst eine Abfrage, wie das Mail-System `postfix` eingerichtet werden soll. Für Testzwecke wählen Sie im Zweifel »Nur lokal« und anschließend die Vorgabe »localdomain«.

Nagios und Apache sollten nun laufen. Abschließend prüfen Sie noch, ob beide Dienste nach dem Booten automatisch gestartet werden. Hierzu suchen Sie in den entsprechenden Ordnern nach Links auf deren Startscripte:

```
root@moni-pangolin-nagios:~# find /etc/rc?.d/ -iname "*nagios3"
/etc/rc0.d/K18nagios3
/etc/rc1.d/K18nagios3
/etc/rc2.d/S30nagios3
/etc/rc3.d/S30nagios3
/etc/rc4.d/S30nagios3
/etc/rc5.d/S30nagios3
/etc/rc6.d/K18nagios3
root@moni-pangolin-nagios:~# find /etc/rc?.d/ -iname "*apache2"
/etc/rc0.d/K09apache2
/etc/rc1.d/K09apache2
/etc/rc2.d/S91apache2
/etc/rc3.d/S91apache2
/etc/rc4.d/S91apache2
/etc/rc5.d/S91apache2
/etc/rc6.d/K09apache2
```

Die Nummer im Namen des Ordners gibt dabei den Runlevel an, und der erste Buchstabe im Namen des Verweises spezifiziert, ob der Dienst in diesem Runlevel gestartet (`S` für `start`) oder gestoppt (`K` für `kill`) werden soll.

`insserv`: Wenn ein Service automatisch gestartet werden soll, können Sie dies bei auf Debian basierenden Distributionen meist mit dem Kommando `insserv <service>` konfigurieren

Der für die Bereitstellung der Weboberfläche mit-installierte Webserver Apache wurde bei der Installation so eingerichtet, dass ein Zugriff für den Webbenutzer `nagiosadmin` unter Eingabe des vergebenen Passwortes auf *http://127.0.0.1/nagios3* beziehungsweise unter der eigenen IP-Adresse möglich ist. Die entsprechende Konfiguration des Apache-Servers finden Sie in */etc/apache2/conf.d/nagios3.conf*.

Konfigurationsschritte: Die adäquaten Schritte für eine manuelle, angepasste Konfiguration können Sie der quellcodebasierenden Nagios-Installation entnehmen (siehe Rezept 1.9, *Nagios aus den Quellen installieren*).

Nach Abschluss der Konfiguration laufen Nagios und Apache und es wird bereits eine Überwachung der lokalen Maschine durchgeführt. An dieser Stelle sollten Sie einen ersten Blick auf die Weboberfläche von Nagios werfen.

Oberflächentest von der Kommandozeile: Falls der entfernte Webzugriff auf Ihr System (noch) nicht möglich ist, beispielsweise aufgrund einer Firewall, können Sie auch über *http://127.0.0.1/nagios3/* auf Ihr frisch installiertes Nagios zugreifen. Wenn Sie über keine graphische Oberfläche und bzw. oder keinen lokalen Webbrowser verfügen, können Sie die Erreichbarkeit der Oberfläche durch folgende Anweisung mit dem Kommandozeilen-Werkzeug `telnet` testen:

```
moni-pangolin-nagios:~ # telnet localhost 80
[...]
GET /nagios3
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en"> <head>
<title>Authentication required!</title>
[...]
```

Installation der Datenbankanbindung über IDOUtils

Mit dem Paket `ndoutils` können Sie nun noch den Anschluss an einen MySQL-Server installieren. In aktuellen Ubuntu-Versionen ist das Paket `ndoutils-nagios3-mysql` ab Version 1.4 verfügbar. Um dieses zu installieren, rufen Sie den Paketmanager Aptitude noch einmal auf:

```
root@moni-pangolin-nagios:~# aptitude install ndoutils-nagios3-mysql
```

Da hierbei `mysql-server` als Abhängigkeit installiert wird, wird für den MySQL-Benutzer `root` ein Passwort abgefragt. Daraufhin starten Sie eine geführte Installation der Datenbank für die NDOUtils. Dann wird das soeben eingerichtete Passwort für den administrativen Datenbank-Benutzer verlangt und anschließend geben Sie ein neues Passwort für den einzurichtenden MySQL-Benutzer `ndoutils` an.

MySQL-Passwörter: Wenn Sie das hier gesetzte Passwort für den `root`-Nutzer von MySQL ändern wollen, funktioniert dies folgendermaßen:

```
root@moni-pangolin-nagios:~# mysqladmin -u root -p PASSWORD
```

Um das Passwort eines nicht-administrativen MySQL-Benutzers zu ändern, können Sie sich von der Kommandozeile aus als Benutzer `root` auf dem Server anmelden und das nachfolgend angeführte Kommando nutzen:

```
root@moni-pangolin-nagios:~# mysql -u root -p
mysql> use mysql;
mysql> update user set Password=DBUSERPW('NEWDBUSERPW') WHERE User='DBUSER';
mysql> exit
```

Die NDOUtils werden dabei noch nicht mit Nagios verknüpft und wohl deshalb auch nicht automatisch gestartet. Zunächst müssen wir Nagios also anweisen, dass es die Daten an das für diesen Zweck vorgesehene und mitinstallierte `ndomod` übergeben soll.

Hierzu tragen Sie das Modul in der Datei */etc/nagios3/nagios.cfg* als sogenanntes Broker-Modul ein:

```
[...]
# EVENT BROKER MODULE(S)
[...]
broker_module=/usr/lib/ndoutils/ndomod-mysql-3x.o config_file=/etc/nagios3/ndomod.cfg
[...]
```

Dann müssen Sie die NDOUtils selbst noch aktivieren, indem Sie in der Datei */etc/default/ndoutils* den entsprechender Schalter setzen:

```
ENABLE_NDOUTILS=1
```

Jetzt prüfen Sie noch, ob die Startscripte für die NDOUtils und den MySQL-Server eingerichtet wurden:

```
root@moni-pangolin-nagios:~# find /etc/rc?.d/ -iname "*ndoutils"
/etc/rc0.d/K20ndoutils
/etc/rc1.d/K20ndoutils
/etc/rc2.d/S20ndoutils
/etc/rc3.d/S20ndoutils
/etc/rc4.d/S20ndoutils
/etc/rc5.d/S20ndoutils
/etc/rc6.d/K20ndoutils
root@moni-pangolin-nagios:~# find /etc/init -iname "mysql*"
/etc/init/mysql.conf
```

Damit die Konfiguration aktiv wird müssen Sie die NDOUtils starten und einen Neustart von Nagios durchführen:

```
root@moni-pangolin-nagios:~# service ndoutils start
root@moni-pangolin-nagios:~# service nagios3 restart
```

Abschließend können Sie jetzt überprüfen, ob tatsächlich Daten in die Datenbank geschrieben werden. Dazu melden Sie sich als `root`-Benutzer von MySQL am Datenbankserver an und überprüfen, ob die Nagios-Datenbank kürzlich aktualisiert wurde. Lassen Sie sich dazu den Eintrag in der Tabelle `nagios_programstatus` anzeigen:

```
root@moni-pangolin-nagios:~#  mysql -u root -p
[...]
mysql> USE ndoutils;
mysql> SELECT status_update_time FROM nagios_programstatus;
+---------------------+
| status_update_time  |
+---------------------+
| 2013-03-21 20:46:47 |
+---------------------+
1 row in set (0.00 sec)
mysql> exit
```

Damit ist die Installation von Nagios und den NDOUtils abgeschlossen. Die Daten werden nun in die neu angelegte Datenbank geschrieben.

Zulassen externer Kommandos

Bei der paketbasierten Standard-Installation von Nagios unter Ubuntu sind unter Umständen keine externen Befehle zugelassen. Diese werden benötigt, um aus der Oberfläche beispielsweise einen Kommentar zu setzen, einen Check manuell anzustoßen oder ein Problem als bekannt zu markieren. Um externe Befehle zu aktivieren, müssen Sie den folgenden Parameter in der Icinga-Konfigurationsdatei */etc/nagios3/nagios.cfg* aktivieren:

```
[...]
# EXTERNAL COMMAND OPTION
[...]
check_external_commands=1
[...]
```

Abschließend müssen Sie noch die Zugriffsrechte für die Kommando-Datei wie folgt anpassen, damit diese nicht beim nächsten Paket-Upgrade überschrieben werden:

```
root@moni-pangolin-nagios:~# service nagios3 stop
root@moni-pangolin-nagios:~# dpkg-statoverride --update --add nagios www-data
2710 /var/lib/nagios3/rw
root@moni-pangolin-nagios:~# dpkg-statoverride --update --add nagios nagios
751 /var/lib/nagios3
root@moni-pangolin-nagios:~# service nagios3 start
```

Sie sollten jetzt externe Kommandos in Nagios absetzen können.

Installation einer graphischen Auswertung mit PNP4Nagios

Jetzt können Sie mit PNP4Nagios ein Addon zur Analyse und Darstellung von Performancedaten installieren. Precise Pangolin (Version 12.04) enthält PNP4Nagios ab Version 0.6.13. Mit folgender Anweisung installieren Sie das Paket `pnp4nagios`:

```
root@moni-pangolin-nagios:~# aptitude install pnp4nagios
```

Paketabhängigkeiten: Lassen Sie einen eventuell entstehenden Konflikt zwischen Abhängigkeiten einzelner Pakete durch den Paketmanager `aptitude` auflösen. Wählen Sie im Zweifel die zuerst angebotene Lösung.

PNP4Nagios speichert die gesammelten Daten in Round-Robin-Datenbanken (RRD). Über das Paket `rrdtool` werden die für diesen Zweck erforderlichen Programme mitinstalliert.

Standardmäßig werden von PNP4Nagios eine ganze Reihe von Betriebsmodi (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/modes*) unterstützt. Der Modus mit NPCD (Nagios Perfdata C Daemon) wird vom PNP4Nagios-Projekt mit »Dies ist aus Nagios-Sicht die sauberste Art der Verarbeitung« beschrieben. Alternativ können Sie Synchronous Mode installieren, der etwas einfacher zu konfigurieren ist. Im Folgenden werden wir deshalb die Installation des »Synchronous Mode« beschreiben. Für die Installation des »Bulk mode mit NPCD« orientieren Sie sich an dem entsprechenden Abschnitt im Rezept 1.6, *Icinga unter Ubuntu installieren*.

Als Nächstes müssen Sie Nagios so konfigurieren, dass zusätzlich zu den reinen Statusinformationen auch die sogenannten Performancedaten verarbeitet werden. Dazu nehmen Sie folgende Modifikationen in der Datei */etc/nagios3/nagios.cfg* vor:

```
[...]
# PROCESS PERFORMANCE DATA OPTION
[...]
process_performance_data=1
[...]
# HOST AND SERVICE PERFORMANCE DATA PROCESSING COMMANDS
[...]
host_perfdata_command=pnp-synchronous-host
service_perfdata_command=pnp-synchronous-service
[...]
```

Die referenzierten Befehle `pnp-synchronous-host` und `pnp-synchronous-service` finden Sie in der bei der Installation von PNP4Nagios erstellten Datei */etc/nagios3/conf.d/pnp4nagios.cfg* (zu Kommandodefinitionen siehe Rezept 4.3, *Befehle hinzufügen (command)*):

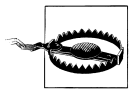
```
define command {
       command_name                            pnp-synchronous-service
       command_line                            /usr/bin/perl /usr/lib/pnp4nagios/ \
libexec/process_perfdata.pl
}

define command {
       command_name                            pnp-synchronous-host
       command_line                            /usr/bin/perl /usr/lib/pnp4nagios/ \
libexec/process_perfdata.pl -d HOSTPERFDATA
}
```

Anschließend starten Sie den Apache-Webserver und Nagios neu:

```
root@moni-pangolin-nagios:~# service apache2 restart
root@moni-pangolin-nagios:~# service nagios3 restart
```

Nach einigen Minuten wird im Verzeichnis */var/lib/pnp4nagios/perfdata* für jedes Gerät ein Verzeichnis angelegt. In diesem Verzeichnis wird dann für jeden Service eine RRD-Datenbank und eine entsprechende XML-Datei erstellt. Prüfen Sie dies mit folgendem Kommando für `localhost`:

```
root@moni-pangolin-nagios:~# ls /var/lib/pnp4nagios/perfdata/localhost/
Current_Load.rrd  Current_Users.rrd  Disk_Space.rrd  _HOST_.rrd  HTTP.rrd
Current_Load.xml  Current_Users.xml  Disk_Space.xml  _HOST_.xml  HTTP.xml
```



Keine Graphen: Wenn PNP4Nagios keine Performancedaten findet, können auch keine Graphen generiert werden. Prüfen Sie bei fehlenden Graphen zuerst, ob von den entsprechenden Check-Plugins tatsächlich Performancedaten erzeugt werden.

An dieser Stelle können Sie PNP4Nagios bereits unter *http://127.0.0.1/pnp4nagios* erreichen und sich nach einiger Zeit die ersten Graphen ansehen. Dabei nutzen Sie die unter Nagios gesetzten Zugangsdaten. Per Standardeinstellung werden nun für jedes Gerät und jeden Dienst Performancedaten verarbeitet, sofern diese vom Plugin geliefert werden.

Integration von PNP4Nagios

Damit Sie von der Weboberfläche aus direkt auf die Graphen zugreifen können, sollten Sie die Graphen direkt mit der Anzeige von Geräten und Diensten verzahnen. Legen Sie sich zunächst wie im Folgenden angeführt eine Vorlage in dem für diesen Zweck vorgesehenen Verzeichnis an (siehe hierzu Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*) */etc/nagios3/conf.d* an, etwa *pnptemplate.cfg* (siehe Rezept 4.1, *Maschinen einbinden (host)* für die Definition von Maschinen, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* für die Definition von Services und Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System* für die Verwendung des Vorlagensystems):

```
define host {
       name                                    pnp-hst
       register                                0
       action_url                              /pnp4nagios/graph?host=$HOSTNAME$' \
class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=_HOST_
}

define service {
       name                                    pnp-svc
       register                                0
       action_url                              /pnp4nagios/graph?host=$HOSTNAME$&srv=$ \
SERVICEDESC$' class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=$SERVICEDESC$
}
```

Anschließend referenzieren Sie diese Definitionen dann in den vorhandenen Definitionen `generic-host` in der Datei */etc/nagios3/conf.d/generic-host_nagios2.cfg*

```
define host {
       name                                    generic-host
       use                                     pnp-hst
[...]
```

und die für `generic-service` in der Datei vim */etc/nagios3/conf.d/generic-service_nagios2.cfg*:

```
define service {
       name                                    generic-service
       use                                     pnp-svc
[...]
```

Graph-Integration: Hierbei werden entsprechende Icons für alle Geräte und Services definiert, auch für solche ohne Graphen. Sie haben aber alternativ auch die Möglichkeit, die Vorlage `pnp-svc` nur bei den Maschinen bzw. Diensten einzubinden, für die Sie Graphen haben oder anzeigen möchten.

Die oben als Vorlage definierte Konfigurationsoption `action_url` wird jeder Maschine und jedem Dienst ein Graphen-Symbol hinzufügen, allerdings erst nach dem nächsten Neustart von Nagios, vor dem Sie aber noch eine weitere Anpassung vornehmen sollten. Sie können sich die Graphen nämlich bereits bei einem Darüberfahren mit der Maus

(mouse-over) anzeigen lassen. Kopieren Sie hierzu die als Vorlage gelieferte Konfigurationsdatei *status-header.ssi* in das Verzeichnis */usr/share/nagios3/htdocs/ssi/* und passen Sie die betreffenden Rechte wie folgt an:

```
root@moni-pangolin-nagios:/# cp /usr/share/doc/pnp4nagios/examples/ssi/status-header.ssi
/usr/share/nagios3/htdocs/ssi/
root@moni-pangolin-nagios:/# chmod 644 /usr/share/nagios3/htdocs/ssi/status-header.ssi
```

Starten Sie nun Nagios neu, damit die geänderten Dateien eingelesen werden:

```
root@moni-pangolin-nagios:~# service nagios3 restart
```

Damit weist die klassische Oberfläche nun Symbole für die Graphen auf. Beim Darüberfahren mit der Maus sehen Sie eine Vorschau, durch Klick gelangen Sie auf die Seite von PNP4Nagios mit dem entsprechenden Graphen. Für den weiteren Umgang mit PNP4Nagios lesen Sie bitte die Rezepte zur Datenvisualisierung (Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*).

Installation des Visalisierungsaddons NagVis

Bei NagVis handelt es sich um ein Addon, mit dem Sie Symbole für Maschinen und Services auf einer Oberfläche beliebig anordnen können. Durch die Wahl eines entsprechenden Hintergrundes erhalten die Symbole dadurch einen Mehrwert, etwa durch die Platzierung auf Fotos oder Karten. Precise Pangolin (Version 12.04) enthält NagVis ab Version 1.4.6, dies war aber leider in unserer Referenzinstallation nicht funktionsfähig.

Wir gehen davon aus das der Fehler mit Erscheinen dieses Buches behoben sein wird. Bei einer paketbasierten Installation von NagVis empfehlen wir Ihnen, dass Sie sich an dem entsprechenden Abschnitt des Installationsrezeptes für Debian orientieren (Rezept 1.10, *Nagios unter Debian installieren*). Anderenfalls müssen Sie mit der sourcebasierten Installation vorlieb nehmen (siehe dazu Rezept 1.9, *Nagios aus den Quellen installieren*). Der Vorteil besteht darin, dass Sie dann auf die aktuelle NagVis-Version zugrückgreifen können.

Diskussion

Die Ablageorte für diverse Komponenten und Konfigurationsdateien sind für Sie von Bedeutung, damit Sie schnell die richtige Stelle für gewünschte Änderungen finden können. Der Tabelle 1-31 können Sie zunächst entnehmen, an welchen Speicherorten die einzelnen Komponenten der Nagios-Installation hier abgelegt wurden.

Tabelle 1-31: Ablageorte der Komponenten von Nagios

Komponente	Pfad
Programm	/usr/sbin/nagios3
Haupt-Konfigurationsdatei	/etc/nagios3/nagios.cfg
Plugins	/usr/lib64/nagios/plugins
Logdatei	/var/log/nagios3/nagios.log
Webserver-Konfiguration	/etc/apache2/conf.d/nagios.conf
Webseiten	/usr/share/nagios/

Von den NDOUtils werden die in Tabelle 1-32 aufgeführten Speicherorte genutzt:

Tabelle 1-32: Pfade der wichtigsten Dateien von NDOUtils

Komponente	Pfad
Konfigurationsdatei ndo2db	/etc/nagios3/ndo2db.cfg
Konfigurationsdatei ndomod	/etc/nagios3/ndomod.cfg
Debug-Logs	/var/log/nagios3/*.debug

Über die Paket-Abhängigkeit werden hier die Plugins aus dem Paket `nagios-plugins` installiert, das die Pakete `nagios-plugins-basic` und `nagios-plugins-standard` umfasst. Auf die enthaltenen Plugins werden wir in den Rezepten der Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen* und Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen* genauer eingehen. Für die Überwachung von System-Updates können Sie das in den Nagios-Plugins enthaltene Plugin `check_apt` verwenden (siehe Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*). Als weitere Pakete mit Plugins stehen Ihnen `nagios-snmp-plugins`, `nagios-plugins-extra`, `gosa-plugin-nagios`, `uwsgi-plugin-nagios` sowie diverse `check-mk-*` zur Verfügung.

Unter Ubuntu wird für die Plugins außerdem ein eigenes Konfigurationsverzeichnis unter */etc/nagios-plugins* erstellt, das vorgefertigte Kommando-Definitionen für die Einbindung in Nagios zur Verfügung stellt (siehe Rezept 4.3, *Befehle hinzufügen (command)*).

Als Erweiterungen für Icinga sind in der Ubuntu-Version neben `pnp4nagios` und `nagvis` auch noch die Pakete `nagiosgrapher`, `nagstamon`, `nagircbot` verfügbar. Damit sind hier über das Paketmanagement mehrere Lösungen für eine graphische Aufbereitung sowie für den Anschluss an einen IRC-Server verfügbar, auf die wir hier aber nicht weiter eingehen werden. Als weiteres Addon können Sie NRPE (Nagios Remote Plugin Executor) installieren (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*).

Siehe auch

- Rezept mit Entscheidungshilfe zur Installation: Rezept 1.1, *Entscheidungshilfe für die Installation*
- Webseite des Nagios-Projekts: *http://www.nagios.org//*
- Webseite von Nagiosplugins: *http://nagiosplugins.org//*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Webseite von NagVis: *http://www.nagvis.org/*
- Webseite von Ubuntu: *http://www.ubuntu.com/*
- Rezept zur quellenbasierten Installation von Nagios: Rezept 1.9, *Nagios aus den Quellen installieren*

- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zum Aufbau und strukturierter Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Kapitel zur Konfiguration von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere das Rezept 4.3, *Befehle hinzufügen (command)*, Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* und Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Kapitel zur Datenvisualisierung mit PNP4Nagios und NagVis: Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*
- Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezept zur Überwachung von Systemupdates: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*
- Rezepte zur Installation von NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

1.12 Nagios unter SLES|openSUSE installieren

Problem

Sie möchten Nagios unter SLES|openSUSE paketbasiert installieren.

Lösung

Wir beschreiben im folgenden zunächst die Installation von Nagios selbst unter der openSUSE LTS Version 12.2, dabei verläuft die Installation unter SLES weitgehend analog Darauf aufbauend können Sie in wenigen weiteren Schritten das Addon NDOUtils (Nagios Data Out Utilities) installieren, um die Daten für weitere Addons und Auswertungen in einer Datenbank vorzuhalten. Anschließend werden wir Ihnen an einem Beispiel die Installation von PNP4Nagios und NagVis zeigen, wozu Sie allerdings weitere Repositories hinzufügen müssen. Alternativ müssten Sie mit der sourcebasierten Installation vorliebnehmen (siehe Rezept 1.9, *Nagios aus den Quellen installieren*) oder zu einer anderen Distribution wechseln (siehe Rezept 1.1, *Entscheidungshilfe für die Installation*).

Installation von Nagios

In den Paketquellen der aktuellen openSUSE Version 12.2 ist Nagios in der Version ab 3.4.1 vorhanden.

Für PNP4Nagios und NagVis sind unter der Standardinstallation von openSUSE 12.2 noch keine Pakete verfügbar. Sie müssen dazu das openSUSE-Repository `server:monitoring` hinzufügen und für NagVis zusätzlich das Repository `server:php:applications`. Wenn Sie wissen, dass Sie diese Addons ebenfalls installieren werden, sollten Sie beide Repositories auch schon vor der Installation von Nagios hinzufügen. Dies gilt ebenfalls für eine Installation unter SUSE LINUX Enterprise.

Yast versus Zypper: Unter openSUSE steht neben dem systemeigenen graphischen Konfigurationstool YAST auch der Paketmanager Zypper zur Verfügung. Beide Tools greifen auf dieselbe Datenbasis zu und können somit alternativ genutzt werden. Im Gegensatz zu Yast deinstalliert Zypper automatisch auch abhängige Pakete, weshalb wir in diesem Buch mit Zypper arbeiten.

Deinstallation: Sofern Sie eine Deinstallation erwarten und dabei sichergehen möchten, dass alle mitinstallierten Pakete wieder entfernt werden, sollten Sie sich an dieser Stelle die Liste der auf Ihrem System mit-installierten Pakete kopieren. Der Hintergrund dabei ist der, dass bei der Deinstallation über Paketmanager regelmäßig ganze Pakete auf dem System verbleiben (dann nämlich, wenn diese in anderen installierten Paketen als optionale Abhängigkeit gelistet sind).

Paket-Quellen: Sie können Repositories unter openSUSE sowohl mit YAST als auch mit Zypper hinzufügen. Die openSUSE-Repositories `server:monitoring` und `server:php:applications` können Sie mit Hilfe von Zypper wie folgt hinzufügen. Dabei müssen Sie den Namen der openSUSE-Distribution gegebenenfalls anpassen:

```
moni-suse122-nagios:~ # zypper ar
http://download.opensuse.org/repositories/server:/monitoring/
openSUSE_12.2/ server-mon
moni-suse122-nagios:~ # zypper ar
http://download.opensuse.org/repositories/server:/php:/applications/
openSUSE_12.2/ server-php-app
```

Basierend auf einer minimalen Serverinstallation von openSUSE in der 64Bit-Server-Variante können Sie mit folgendem Befehl eine laufende Nagios-Umgebung einrichten:

```
moni-suse122-nagios:~ # zypper install nagios
```

Die Abhängigkeiten werden deabei automatisch aufgelöst und die genannten Pakete installiert. Sie müssen nun zunächst ein Passwort für den in der Konfiguration des Web-

servers vorgesehenen Schutz durch `http-auth` (siehe Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*) einrichten:

```
moni-suse122-nagios:~ # htpasswd2 -c /etc/nagios/htpasswd.users nagiosadmin
```

Um die beiden Dienste nach jedem Bootvorgang automatisch zu starten, müssen Sie noch die folgenden Anweisungen ausführen:

```
moni-suse122-nagios:~ # systemctl enable apache2.service
[...]
moni-suse122-nagios:~ # chkconfig -a nagios
[...]
nagios                  0:off  1:off  2:off  3:on   4:off  5:on   6:off
```

Automatischer Start: Wenn ein Service automatisch gestartet werden soll, können Sie dies mit dem Kommando `chkconfig <service> on|off` (RHEL|CentOS|Fedore, SLES|openSUSE) konfigurieren.

Ein Neustart sollte nicht notwendig sein. Sie können beide Dienste wie folgt manuell starten:

```
moni-suse122-nagios:~ # systemctl start apache2.service
moni-suse122-nagios:~ # systemctl start nagios.service
```

Nagios ist nun mit dem Benutzernamen `nagiosadmin` und dem vorher gesetzten Passwort unter *http://127.0.0.1/nagios/* beziehungsweise über die eigene IP-Adresse erreichbar und führt bereits eine Überwachung des lokalen Systems durch. Die entsprechende Apache-Konfiguration finden Sie unter */etc/apache2/conf.d/nagios.conf*.

Konfigurationsschritte: Die adäquaten Schritte für eine manuelle, angepasste Konfiguration können Sie der quellcodebasierenden Nagios-Installation entnehmen (siehe Rezept 1.9, *Nagios aus den Quellen installieren*).

Werfen Sie an dieser Stelle einen ersten Blick auf die Nagios-Weboberfläche. Stellen Sie gegebenenfalls vorher sicher, dass die lokale SUSE-Firewall die für den Apache-Webserver notwendigen Ports (80 beziehungsweise 443) geöffnet hat. Die Firewall-Konfiguration können Sie mittels des SUSE-Tools `yast` anpassen lassen.

Oberflächentest von der Kommandozeile: Falls der entfernte Webzugriff auf Ihr System (noch) nicht möglich ist, beispielsweise aufgrund einer Firewall, können Sie auch über *http://127.0.0.1/nagios/* auf Ihr frisch installiertes Nagios zugreifen. Wenn Sie über keine graphische Oberfläche und bzw. oder keinen lokalen Webbrowser verfügen, können Sie die Erreichbarkeit der Oberfläche durch folgende Anweisung mit dem Kommandozeilen-Werkzeug `telnet` testen:

```
moni-suse122-nagios:~ # telnet localhost 80
[...]
GET /nagios
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en"> <head>
<title>Authentication required!</title>
[...]
```

Installation der Datenbankanbindung über NDOUtils

Als nächsten Schritt können Sie jetzt die (Nagios Data Out Utilities) installieren, die die Anbindung von Nagios an einen Datenbankserver ermöglichen, in den später die Daten geschrieben werden. OpenSUSE 12.2 enthält das Paket `ndoutils` ab der Version 1.5.1. Bei openSUSE werden Datenbanken nicht automatisch mit-installiert, so dass wir erst einen MySQL-Datenbankserver installieren müssen:

```
moni-suse122-nagios:~ # zypper install mysql-community-server
```

Der MySQL-Server läuft nach der Installation noch nicht und hat noch kein Passwort. Richten Sie ihn zunächst als Autostart-Dienst ein und starten Sie den Datenbankserver dann:

```
moni-suse122-nagios:~ # chkconfig mysql on
moni-suse122-nagios:~ # systemctl start mysql.service
```

Anschließend setzen Sie dann das Passwort für den Benutzer `root`:

```
moni-suse122-nagios:~ # mysqladmin -u root password
```

MySQL-Passwörter: Wenn Sie das hier gesetzte Passwort für den `root`-Nutzer von MySQL ändern wollen, funktioniert dies folgendermaßen:

```
moni-suse122-nagios:~ # mysqladmin -u root -p PASSWORD
```

Um das Passwort eines nicht-administrativen MySQL-Benutzers zu ändern, können Sie sich von der Kommandozeile aus als Benutzer `root` auf dem Server anmelden und das nachfolgend angeführte Kommando nutzen:

```
moni-suse122-nagios:~ # mysql -u root -p
mysql> use mysql;
mysql> update user set Password=DBUSERPW('NEWDBUSERPW') WHERE User='DBUSER';
mysql> exit
```

Damit haben Sie einen laufenden Datenbankserver und können nun die NDOUtils installieren:

```
moni-suse122-nagios:~ # zypper install ndoutils
```

Sie müssen nun für die DOUtils eine Datenbank einrichten und konfigurieren. Dazu melden Sie sich zunächst mit als Benutzer `root` am Datenbankserver an und richten die

Datenbank nagios und den MySQL-Benutzer sql_nagios mit entsprechenden Rechten sowie einem Passwort (hier DBPASSWD) ein:

```
moni-suse122-nagios:~ # mysql -u root -p
mysql> CREATE DATABASE nagios;
mysql> GRANT USAGE ON nagios .* TO 'sql_nagios'@'localhost' IDENTIFIED BY 'DBUSERPW' WITH
 MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0;
mysql> GRANT SELECT , INSERT , UPDATE , DELETE, DROP, CREATE VIEW ON nagios.* TO
'sql_nagios'@'localhost';
mysql> FLUSH PRIVILEGES;
mysql> exit
```

Des Weiteren müssen Sie in der Datenbank die von den IDOUtils vorgegebene Struktur anlegen:

```
moni-suse122-nagios:~ # mysql -u root -p nagios
< /usr/share/doc/packages/ndoutils/db/mysql.sql
```

Die Daten für den Datenbankzugriff müssen in der Datei */etc/nagios/ndo2db.cfg* hinterlegt sein. Sie werden zumindest den Wert der Variablen db_pass in das weiter vorne festgelegte Passwort des Datenbankbenutzers nagios ändern müssen (hier ebenso den Benutzernamen in db_user):

```
db_servertype=mysql
db_host=localhost
db_port=3306
db_name=nagios
db_prefix=nagios_
db_user=sql_nagios
db_pass=DBUSERPW
```

Damit Nagios das sogenannte Broker-Modul ndomod zur Übergabe der Daten an die NDOUtils auch nutzt, müssen Sie dies noch durch den folgenden Eintrag in */etc/nagios/nagios.cfg* entsprechend festlegen:

```
[...]
# EVENT BROKER MODULE(S)
[...]
broker_module=/usr/lib/nagios/brokers/ndomod.o config_file=/etc/nagios/ndomod.cfg
[...]
```

Nun teilen Sie dem System die NDOUtils zunächst als neuen Dienst mit:

```
moni-suse122-nagios:~ # chkconfig ndo2db on
```

Vergewissern Sie sich anschließend mit Hilfe der folgenden Anweisung, dass alle notwendigen Dienste für den automatischen Start eingerichtet sind:

```
moni-suse122-nagios:~ # systemctl is-enabled apache2.service
enabled
moni-suse122-nagios:~ # chkconfig nagios
nagios on
moni-suse122-nagios:~ # chkconfig mysql
mysql on
moni-suse122-nagios:~ # chkconfig ndo2db
ndo2db on
```

Als nächsten Schritt starten Sie dann das konfigurierte System neu, damit alle vorgenommenen Änderungen wirksam werden:

```
moni-suse122-nagios:~ # systemctl reboot
```

Abschließend können Sie jetzt nach dem Neustart überprüfen, ob auch tatsächlich Daten in die Datenbank geschrieben werden. Melden Sie sich dazu als MySQL-Benutzer `root` am Datenbankserver an und überprüfen, ob die Nagios-Datenbank kürzlich aktualisiert wurde. Dazu lassen Sie sich den Eintrag der Tabelle `nagios_programstatus` anzeigen:

```
moni-suse122-nagios:~ # mysql -u root -p
mysql> USE nagios;
mysql> SELECT status_update_time FROM nagios_programstatus;
+---------------------+
| status_update_time |
+---------------------+
| 2012-02-08 23:23:52 |
+---------------------+
1 row in set (0.00 sec)

mysql> exit
```

Nun laufen sowohl Nagios als auch NDOUtils und die Daten werden in die angelegte Datenbank geschrieben.

Warteschlangen-Fehler: In den NDOUtils ab Version 1.5.2 werden Nachrichten nur dann in die Warteschlange weitergeleitet, wenn der Kernel diese nicht annimmt. Falls Sie alles einwandfrei konfiguriert haben und trotzdem keine Daten in Ihre Datenbank geschrieben werden, überprüfen Sie, ob die Daten auch korrekt weitergeleitet werden:

```
moni-suse122-nagios:~ # tail -f /var/log/messages | grep ndo2db
[...]
Jan 21 16:49:37 moni-suse122-nagios ndo2db[1893]: Starting ndo2db ..done
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Successfully connected to
MySQL database
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Successfully connected to
MySQL database
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Error: queue send error.
Jan 21 16:49:37 moni-suse122-nagios ndo2db: last message repeated 39 times
[...]
```

Sie können das Problem lösen, indem Sie die Zahl der maximalen Nachrichten erhöhen. Fügen Sie dazu folgende Zeilen in der Datei `/etc/sysctl.conf` hinzu:

```
kernel.msgmax = 131072000
kernel.msgmnb = 131072000
kernel.msgmni = 65536000
```

Installation einer graphischen Auswertung mit PNP4Nagios

PNP4Nagios ist ein Addon zur Analyse und Darstellung von Performancedaten. Um dieses zu installieren, benötigen Sie das `server:monitoring-Repository` von openSUSE, in dem eine aktuelle Version von PNP4Nagios enthalten ist. Sie können das Addon wie folgt installieren:

```
moni-suse122-nagios:~ # zypper install pnp4nagios
```

In PNP4Nagios werden die gesammelten Daten in sogenannten Round-Robin-Datenbanken (RRD) gespeichert. Die dafür erforderlichen Programme werden über das Paket `rrdtool` automatisch mitinstalliert. PNP4Nagios unterstützt eine ganze Reihe von Betriebsmodi (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/modes*). Der Modus mit NPCD (Nagios Perfdata C Daemon) wird vom PNP4Nagios-Projekt mit »Dies ist aus Nagios-Sicht die sauberste Art der Verarbeitung« beschrieben. Alternativ können Sie Synchronous Mode installieren, der etwas einfacher zu konfigurieren ist. Im Folgenden werden wir die Installation des »Synchronous Mode« beschreiben. Für die Installation des »Bulk mode mit NPCD« orientieren Sie sich an dem entsprechenden Abschnitt im Rezept 1.7, *Icinga unter SLES|openSUSE installieren*.

Anschließend müssen Sie Nagios zuerst so konfigurieren, dass zusätzlich zu den reinen Statusinformationen eine Verarbeitung der sogenannten Performancedaten stattfindet. Dazu modifizieren Sie die folgenden Zeilen der Datei */etc/nagios/nagios.cfg*:

```
[...]
# PROCESS PERFORMANCE DATA OPTION
[...]
process_performance_data=1
[...]
# HOST AND SERVICE PERFORMANCE DATA PROCESSING COMMANDS
[...]
host_perfdata_command=process-host-perfdata
service_perfdata_command=process-service-perfdata
[...]
```

Außerdem müssen Sie noch die soeben referenzierten Befehle `process-host-perfdata` und `process-service-perfdata` wie folgt in der Datei */etc/nagios/commands.cfg* anpassen (zu Kommandodefinitionen siehe auch Rezept 4.3, *Befehle hinzufügen (command)*):

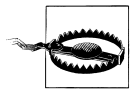
```
# 'process-host-perfdata' command definition
define command {
       command_name process-host-perfdata
       command_line /usr/bin/perl /usr/lib/nagios/plugins/process_perfdata.pl
-d HOSTPERFDATA
}

# 'process-service-perfdata' command definition
define command {
       command_name process-service-perfdata
       command_line /usr/bin/perl /usr/lib/nagios/plugins/process_perfdata.pl
}
```

Starten Sie nun den Webserver und Nagios neu:

```
moni-suse122-nagios:/ # service apache2 restart
moni-suse122-nagios:/ # service nagios restart
```

Im Anschluss an die erste Messung werden nun von Nagios für jedes Gerät Performancedaten im Verzeichnis */var/lib/pnp4nagios/perfdata* abgelegt. Für jeden Service dieses Gerätes wird eine RRD-Datenbank und eine entsprechende XML-Datei erstellt. Überprüfen Sie dies gegebenenfalls nach einigen Minuten, indem Sie sich die Dateien im entsprechenden Ordner *localhost* anzeigen lassen:

```
moni-suse122-nagios:/ # ls /var/lib/pnp4nagios/perfdata/localhost/
Current_Load.rrd Current_Users.rrd _HOST_.rrd HTTP.rrd PING.rrd Root_Partition.rrd
Current_Load.xml Current_Users.xml _HOST_.xml HTTP.xml PING.xml Root_Partition.xml
```



Keine Graphen: Wenn PNP4Nagios keine Performancedaten findet, können auch keine Graphen generiert werden. Prüfen Sie bei fehlenden Graphen zuerst, ob von den entsprechenden Check-Plugins tatsächlich Performancedaten erzeugt werden.

Sie können PNP4Nagios unter *http://127.0.0.1/pnp4nagios* und alternativ auch unter der systemeigenen IP-Adresse erreichen. Nutzen Sie dabei die unter Nagios gesetzten Zugangsdaten. Sobald die ersten Messungen durchgeführt wurden, werden dann entsprechende Graphen angezeigt, wobei dies allerdings einige Minuten dauern kann. Per Standardeinstellung werden nun für jedes Gerät und jeden Dienst Performancedaten verarbeitet, sofern diese vom Plugin geliefert werden.

Installation des Visualisierungsaddons NagVis

Mit Hilfe des Addons NagVis können Sie Symbole für Maschinen und Services beliebig anordnen. Durch die Wahl eines entsprechenden Hintergrundes erhalten die Symbole dadurch automatisch einen Mehrwert, etwa durch die Platzierung auf Fotos oder Karten. Um unter openSUSE auf das NagVis-Paket zugreifen zu können, müssen Sie zusätzlich zum `server:monitoring`-Repository auch das `server:php:applications`-Repository hinzufügen. Dieses enthält NagVis ab der Version 1.7.6.

Mit folgender Anweisung können Sie das Paket `nagvis` und das ebenfalls erforderliche Paket `php5-mysql` installieren:

```
moni-suse122-nagios:~# zypper install php5-mysql nagvis
```

Von NagVis werden unterschiedliche Schnittstellen angeboten, mit denen es auf die Monitoringdaten zugreifen kann. Die `ndomy`-Schnittstelle war ursprünglich die Vorgabe, seit Version 1.6.4-1 wurde diese jedoch durch `mklivestatus` ersetzt. Da die ndomy Schnittstelle über die NDOUtils Datenbank auf die Monitoring-Daten zugreift, empfehlen wir Ihnen diese zu nutzen. Setzen Sie dazu in der Datei */etc/nagvis/nagvis.ini.php* die Konfigurationsoption `backend` auf `ndomy_1` und hinterlegen Sie die Zugangsdaten zu der über NDO2DB eingebundenen Datenbank:

```
backend="ndomy_1"
[...]
dbhost="localhost"
dbport=3306
dbname="nagios"
dbuser="sql_nagios"
dbpass="DBUSERPW"
dbprefix="nagios_"
dbinstancename="default"
```

Abschließend müssen Sie noch die NagVis-Konfiguration in den Webserver einbinden, indem Sie mit dem folgenden Befehl die entsprechende Option für den Apache-Server setzen:

```
moni-suse122-nagios:~ # a2enflag NAGVIS
```

Sie können sich jetzt als Benuzter `admin` mit dem Passwort `admin` unter *http://127.0.0.1/nagvis* anmelden. Dabei sollten Sie dann als Erstes das Administrator-Passwort ändern. Ansonsten ist die Basis-Einrichtung von NagVis damit komplett. Für den weiteren Umgang mit PNP4Nagios und NagVis lesen Sie bitte die Rezepte zur Datenvisualisierung (Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*).

Diskussion

Die Ablageorte für diverse Komponenten und Konfigurationsdateien sind für Sie von Bedeutung, damit Sie schnell die richtige Stelle für gewünschte Änderungen finden können. Der Tabelle 1-33 können Sie zunächst entnehmen, an welchen Speicherorten die einzelnen Komponenten der Nagios-Installation hier abgelegt wurden.

Tabelle 1-33: Ablageorte der Komponenten von Nagios

Komponente	Pfad
Programm	/usr/sbin/nagios
Haupt-Konfigurationsdatei	/etc/nagios/nagios.cfg
Plugins	/usr/lib/nagios/plugins
Logdatei	/var/log/nagios/nagios.log
Webserver-Konfiguration	/etc/apache2/conf.d/nagios.conf
Webseiten	/usr/share/nagios/

Die NDOUtils verwenden die in Tabelle 1-34 aufgeführten Speicherorte:

Tabelle 1-34: Pfade der wichtigsten Dateien von NDOUtils

Komponente	Pfad
Konfigurationsdatei ndo2db	/etc/nagios/ndo2db.cfg
Konfigurationsdatei ndomod	/etc/nagios/ndomod.cfg
Debug-Logs	/var/log/nagios/*.debug

Das Addon PNP4Nagios hat hier die in Tabelle 1-35 aufgezeigten Installationsorte verwendet.

Tabelle 1-35: Pfade zu den Komponenten von PNP4Nagios

Komponente	Pfad
Konfigurationsdateien	/etc/pnp4nagios/
Logdateien	/var/log/pnp4nagios/
Webserver-Konfiguration	/etc/apache2/conf.d/nagios-pnp.conf
Webseiten	/usr/share/pnp4nagios/
Script zur Verarbeitung	/usr/lib/nagios/plugins/process_perfdata.pl

Das Addon NagVis hat hier die in Tabelle 1-36 aufgezeigten Installationsorte verwendet.

Tabelle 1-36: Pfade zu den Komponenten von NagVis

Komponente	Pfad
Haupt-Konfigurationsdatei	/etc/nagvis/nagvis.ini.php
Webserver-Konfiguration	/etc/apache2/conf.d/nagvis.conf
Webseiten	/usr/share/nagvis/share/

Dabei wurden die Plugins aus dem Paket `nagios-plugins` installiert. Auf die enthaltenen Plugins werden wir in den Rezepten der Kapitel 5 und 6 genauer eingehen. Für die Überwachung von System-Updates können Sie das Paket `nagios-plugins-zypper` installieren, das das Plugin `check_zypper` enthält . Weitere Plugins stehen Ihnen als Paket bei Bedarf über `nagios-plugins-extras` und andere Pakete zur Verfügung.

Des Weiteren können Sie das Addon NRPE (Nagios Remote Plugin Executor) installieren (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*).

Siehe auch

- Rezept mit Entscheidungshilfe zur Installation: Rezept 1.1, *Entscheidungshilfe für die Installation*
- Webseite des Nagios-Projekts: *http://www.nagios.org//*
- Webseite von Nagiosplugins: *http://nagiosplugins.org//*
- Webseite von openSUSE: *http://de.opensuse.org/*
- Webseite des server:monitoring Repositories: *http://download.opensuse.org/ repositories/server:/monitoring/openSUSE_12.2/*
- Webseite des server:php:applications Repositories: *http://download.opensuse.org/ repositories/server:/php:/applications/openSUSE_12.2/*

- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Webseite von OpenSuSE: *http://de.opensuse.org/*
- Rezept zur quellenbasierten Installation von Nagios: Rezept 1.9, *Nagios aus den Quellen installieren*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zum Aufbau und strukturierter Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Kapitel zur Konfiguration von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere das Rezept 4.3, *Befehle hinzufügen (command)*, Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* sowie Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Kapitel zur Datenvisualisierung mit PNP4Nagios und NagVis: Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis* Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezepte zur Installation von NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

1.13 Nagios unter RHEL|CentOS|Fedora installieren

Problem

Sie möchten Nagios unter unter RHEL|CentOS|Fedora paketbasiert installieren.

Lösung

Wir zeigen Ihnen, wie Sie Nagios unter CentOS installieren. Die Installation unter RHEL und Fedora ist weitgehend analog. Darauf aufbauend werden wir die Installation von optionalen Addons, die ebenfalls als Pakete angeboten werden, erläutern. Nachfolgend zeigen wir Ihnen, wie Sie das Datenbank-Addon NDOUtils installieren, mit dem Sie die Daten in einer Datenbank für weitere Addons vorhalten können. Wir werden anschließend die Installation des Addons PNP4Nagios zur graphischen Auswertung der gesammelten Performancedaten beschreiben. Für das Addon NagVis ist leider in den von uns gewählten Repositories kein Paket verfügbar. Wir empfehlen Ihnen, bei Installation von NagVis auf die sourcebasierte Installation zurückzugreifen (siehe Rezept 1.9, *Nagios aus den Quellen installieren*).

Installation Nagios

Nagios ist nicht Teil der offiziellen RHEL (Red Hat Enterprise Linux)-Distribution und damit also ebenfalls nicht in CentOS enthalten. Um Nagios paketorientiert auf einem auf RHEL-basierenden System zu installieren, müssen Sie als Paketquelle Extra Packages for Enterprise Linux (EPEL) einbinden. EPEL-Pakete basieren auf Fedora-Paketen, sind allerdings so konfiguriert, dass hierdurch auf RHEL-basierenden Systemen keine vorhandenen Pakete ersetzt werden.

Mit dem Kommando `yum repolist` können Sie überprüfen ob das EPEL-Repository bereits in Ihr System eingebunden ist. Falls das nicht der Fall ist, können Sie dies nachholen, indem Sie zuerst den zugehörigen GPG-Schlüssel in Ihrer RPM-Datenbank installieren und dann das Repository hinzufügen:

```
[root@moni-centos63-nagios ~]# rpm --import http://fedoraproject.org/static/0608B895.txt
[root@moni-centos63-nagios ~]# rpm -ivh http://ftp-stud.hs-esslingen.de/pub/epel/6/i386/
epel-release-6-8.noarch.rpm
[root@moni-centos63-nagios ~]# yum repolist
[...]
 * epel: mirrors.n-ix.net
[...]
epel Extra Packages for Enterprise Linux 6 - x86_64 8,326
[...]
```

Das EPEL-Repository enhielt in unserem Fall Nagios ab der Version 3.4.3. Installieren Sie zunächst das Paket `nagios` inklusive der dem System bekannten Abhängigkeiten:

```
[root@moni-centos63-nagios ~]# yum install nagios
```

Deinstallation: Sofern Sie eine Deinstallation erwarten und dabei sichergehen möchten, dass alle mitinstallierten Pakete wieder entfernt werden, sollten Sie sich an dieser Stelle die Liste der auf Ihrem System mitinstallierten Pakete kopieren. Der Hintergrund dabei ist der, dass bei der Deinstallation über Paketmanager regelmäßig ganze Pakete auf dem System verbleiben (dann nämlich, wenn diese in anderen installierten Paketen als optionale Abhängigkeit gelistet sind).

Dabei wird als Paket `httpd` der Webserver Apache2 mit-installiert. Nagios enthält ein Web-Interface, das standardmäßig eine Authentifizierung über den Webserver mittels `http-auth` erfordert (siehe auch Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*). Richten Sie daher zunächst einen entsprechenden Benutzer `nagiosadmin` mit einem Passwort ein:

```
[root@moni-centos63-nagios ~]# htpasswd -c /etc/nagios/passwd nagiosadmin
```

Sie habe damit die Voraussetzungen erfüllt, die erforderlich sind, um die Weboberfläche ans Laufen zu bringen. Damit diese die Daten von Nagios anzeigen kann, muss der Webserver auf die entsprechenden Verzeichnisse zugreifen dürfen. Ein Weg, um dies zu erreichen, besteht darin, den Benutzer, unter dem der Webserver läuft, in die Benutzergruppe `nagios` aufzunehmen. Sie erreichen dies folgendermaßen:

```
[root@moni-centos63-nagios ~]# usermod -a -G nagios apache
```

Falls Sie über das Netzwerk von außen auf den Webserver zugreifen möchten, lassen Sie Port 80 beziehungsweise 443 in der lokalen Firewall zu, indem Sie in der Datei */etc/sysconfig/iptables* die entsprechende Zeile oben in den Abschnitt ":OUTPUT ACCEPT [0:0]" eintragen:

```
[...]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j ACCEPT
[...]
```

Nagios führt Tests über sogenannte Plugins durch. Über die Abhängigkeiten im Paketmanagement wurden noch keine Plugins installiert. Die automatisch eingerichteten Tests auf `localhost` würden infolgedessen Fehlermeldungen liefern. Installieren Sie daher die über das Paketmanagement verfügbaren Plugins:

```
[root@moni-centos63-nagios ~]# yum install nagios-plugins-all
```

Damit wären die grundlegenden Komponenten prinzipiell einsatzfähig, jedoch wird das standardmäßig aktivierte SELinux nicht automatisch für die Belange von Nagios (und später betrachteten Addons) angepasst. Betrachtungen zur Anpassung des SELinux für Nagios und die einzelnen Module sind nicht Bestandteil dieses Buches. Ändern Sie deshalb bitte im Zweifel den folgenden Wert in der Datei */etc/selinux/config*, um das SELinux in einen Modus zu schalten, in dem es alle Operationen zulässt und lediglich eine entsprechende Protokollierung vornimmt:

```
[...]
#SELINUX=enforcing
SELINUX=permissive
[...]
```

Damit diese Änderung wirksam wird, müssen Sie das System neu starten. In unserem Fall waren der Webserver und Nagios noch deaktiviert. Sie können den automatischen Start der beiden Anwendungen wie folgt sicherstellen:

```
[root@moni-centos63-nagios ~]# chkconfig httpd on
[root@moni-centos63-nagios ~]# chkconfig nagios on
```

Autostart: Wenn ein Service automatisch gestartet werden soll, können Sie dies mit dem Kommando `chkconfig <service> on|off` konfigurieren.

Starten Sie jetzt das System neu:

```
[root@moni-centos63-nagios ~]# reboot
```

Rufen Sie dann den URL *http://127.0.0.1/nagios* auf beziehungsweise setzen Sie für `localhost` die entsprechende IP-Adresse ein. Nagios läuft und führt bereits Tests am lokalen System durch. Der abgefragte Benutzername wäre dann `nagiosadmin` mit dem oben von Ihnen gesetzten Passwort.

Oberflächentest von der Kommandozeile: Falls der entfernte Webzugriff auf Ihr System (noch) nicht möglich ist, beispielsweise aufgrund einer Firewall, können Sie auch über *http://127.0.0.1/nagios/* auf Ihr frisch installiertes Nagios zugreifen. Wenn Sie über keine graphische Oberfläche und bzw. oder keinen lokalen Webbrowser verfügen, können Sie die Erreichbarkeit der Oberfläche durch folgende Anweisung mit dem Kommandozeilen-Werkzeug `telnet` testen:

```
[root@moni-centos63-nagios ~]# telnet localhost 80
[...]
GET /nagios
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en"> <head>
<title>Authentication required!</title>
[...]
```

Installation der Datenbankanbindung über NDOUtils

Die EPEL-Quelle enthält die NDOUtils (Nagios Data Out Utilities) in unserem Fall ab der Version 1.4. Die NDOUtils ermöglichen die Anbindung von Nagios an einen Datenbankserver, in den später die Daten geschrieben werden. Beginnen Sie damit, einen lokalen MySQL-Datenbankserver zu installieren:

```
[root@moni-centos63-nagios ~]# yum install mysql-server
```

Damit haben Sie den Datenbankserver installiert, allerdings ist er weder gestartet noch mit einem Passwort versehen. Starten Sie ihn deshalb zunächst und vergeben Sie dann ein Passwort für den MySQL-Benutzer `root`:

```
[root@moni-centos63-nagios ~]# service mysqld start
[root@moni-centos63-nagios ~]# mysqladmin -u root password 'DBROOTPW'
```

MySQL-Passwörter: Wenn Sie das hier gesetzte Passwort für den `root`-Nutzer von MySQL ändern möchten, funktioniert dies folgendermaßen:

```
[root@moni-centos63-nagios ~] mysqladmin -u root -p PASSWORD
```

Um das Passwort eines nicht-administrativen MySQL-Benutzers zu ändern, können Sie sich von der Kommandozeile aus als Benutzer `root` auf dem Server anmelden und das nachfolgend angeführte Kommando nutzen:

```
[root@moni-centos63-nagios ~] mysql -u root -p
mysql> use mysql;
mysql> update user set Password=DBUSERPW('NEWDBUSERPW') WHERE User='DBUSER';
mysql> exit
```

Nun können Sie sich mit diesem Passwort am Server anmelden und eine Datenbank `nagios` sowie einen Benutzer `sql_nagios` mit Passwort (hier DBUSERPW) erstellen und den Benutzer mit entsprechenden Rechten versehen:

```
[root@moni-centos63-nagios ~]# mysql -u root -p
mysql> CREATE DATABASE nagios;
mysql> GRANT ALL ON nagios.* TO 'sql_nagios'@'localhost' IDENTIFIED BY 'DBUSERPW';
mysql> FLUSH PRIVILEGES;
mysql> exit
```

Installieren Sie nun das Paket `ndoutils`:

```
[root@moni-centos63-nagios ~]# yum install ndoutils-mysql
```

Diese Datenbank initialisieren Sie nun mittels der beim Paket `ndoutils` mitgelieferten Vorlage:

```
[root@moni-centos63-nagios ~]# mysql -u sql_nagios -p nagios <
/usr/share/doc/ndoutils-mysql-1.4/db/mysql.sql
```

Damit die NDOUtils auf die Datenbank zugreifen können, müssen die Zugangsdaten in der Datei */etc/nagios/ndo2db.cfg* hinterlegt sein. Passen Sie hier die erforderlichen Zeilen an (zumindest `db_pass`, hier auch `db_user`):

```
db_servertype=mysql
db_host=localhost
db_port=3306
db_name=nagios
db_prefix=nagios_
db_user=sql_nagios
db_pass=DBUSERPW
```

Damit Nagios das Modul der NDOUtils lädt, tragen Sie nun noch die folgende Zeile in die Datei */etc/nagios/nagios.cfg* ein:

```
[...]
# EVENT BROKER MODULE(S)
[...]
broker_module=/usr/lib64/nagios/brokers/ndomod.so config_file=/etc/nagios/ndomod.cfg
[...]
```

Der MySQL-Server und die NDOUtils sind in unserem Fall noch nicht für den automatischen Start eingerichtet. Dies können Sie wir folgt nachholen:

```
[root@moni-centos63-nagios ~]# chkconfig mysqld on
[root@moni-centos63-nagios ~]# chkconfig ndo2db on
```

Nun müssen Sie nur noch den ndo2db-Agent und Nagios neu starten, damit die vorgenommenen Änderungen wirksam werden.

```
[root@moni-centos63-nagios ~]# service ndo2db start
[root@moni-centos63-nagios ~]# service nagios restart
```

Jetzt können Sie überprüfen, ob die Ergebnisse auch in die Datenbank geschrieben werden. Schauen Sie dazu nach, ob der Eintrag `status_update_time` aus der Tabelle `nagios_programstatus` Einträge enthält:

```
[root@moni-centos63-nagios ~]# mysql -u root -p
mysql> USE nagios;
mysql> SELECT status_update_time FROM nagios_programstatus;
```

```
+---------------------+
| status_update_time  |
+---------------------+
| 2013-01-28 16:47:54 |
+---------------------+
1 row in set (0.00 sec)

mysql> exit
```

Damit haben Sie die Datenbankanbindung für Nagios über NDOUtils erfolgreich installiert, konfiguriert und getestet. Die Daten werden jetzt in die angelegte Datenbank geschrieben.

Warteschlangen-Fehler: In den NDOUtils ab Version 1.5.2 werden Nachrichten nur dann in die Warteschlange weitergeleitet, wenn der Kernel diese nicht annimmt. Falls Sie alles einwandfrei konfiguriert haben und trotzdem keine Daten in Ihre Datenbank geschrieben werden, prüfen Sie, ob die Daten auch korrekt weitergeleitet werden:

```
moni-suse122-nagios:~ # tail -f /var/log/messages | grep ndo2db
[...]
Jan 21 16:49:37 moni-suse122-nagios ndo2db[1893]: Starting ndo2db ..done
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Successfully connected to
MySQL database
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Successfully connected to
MySQL database
Jan 21 16:49:37 moni-suse122-nagios ndo2db: Error: queue send error.
Jan 21 16:49:37 moni-suse122-nagios ndo2db: last message repeated 39 times
[...]
```

Sie können das Problem lösen, indem Sie die Zahl der maximalen Nachrichten erhöhen. Fügen Sie dazu die folgenden Zeilen in der Datei `/etc/sysctl.conf` hinzu:

```
kernel.msgmax = 131072000
kernel.msgmnb = 131072000
kernel.msgmni = 65536000
```


Installation einer graphischen Auswertung mit PNP4Nagios

In unserem Fall enthält das EPEL-Repository PNP4Nagios ab Version 0.4.16. Installieren Sie das Paket `pnp4nagios`, um damit Graphen zeichnen zu lassen:

```
[root@moni-centos63-nagios ~]# yum install pnp4nagios
```

PNP4Nagios speichert Daten in sogenannten Round-Robin-Datenbanken (RRD). Die dafür nötigen Programme wurden über das Paket `rrdtool` mitinstalliert.

Standardmäßig unterstützt PNP4Nagios eine ganze Reihe von Betriebsmodi (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/modes*). Der Modus mit NPCD (Nagios Perfdata C Daemon) wird vom PNP4Nagios-Projekt mit »Dies ist aus Nagios-Sicht die sauberste Art der Verarbeitung« beschrieben. Alternativ können Sie Synchronous Mode installieren, der etwas einfacher zu konfigurieren ist. Im Folgenden werden wir die Installation des

»Synchronous Mode« beschreiben. Für die Installation des »Bulk mode mit NPCD« orientieren Sie sich an dem entsprechenden Abschnitt im Rezept 1.8, *Icinga unter RHEL|CentOS|Fedora installieren*.

Damit Nagios zusätzlich zu den reinen Status-Informationen auch die sogenannten Performance-Daten (Leistungsdaten) verarbeitet, müssen Sie einen entsprechenden Schalter in der Datei */etc/nagios/nagios.cfg* von `0` auf `1` setzen und das Kommentarzeichen am Anfang der Zeilen entfernen, die spezifizieren, welche Kommandos Nagios dabei verwenden soll:

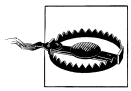
```
[...]
# PROCESS PERFORMANCE DATA OPTION
[...]
process_performance_data=1
[...]
# HOST AND SERVICE PERFORMANCE DATA PROCESSING COMMANDS
[...]
host_perfdata_command=process-host-perfdata
service_perfdata_command=process-service-perfdata
[...]
```

Darüber hinaus passen Sie die vorhandenen Kommandodefinitionen für die Kommandos `process-host-perfdata` und `process-service-perfdata` in der Datei */etc/nagios/objects/commands.cfg* wie folgt an (zu Kommandodefinitionen siehe auch Rezept 4.3, *Befehle hinzufügen (command)*):

```
# 'process-host-perfdata' command definition
define command {
       command_name                    process-host-perfdata
       command_line                    /usr/bin/perl /usr/libexec/pnp4nagios/ \
process_perfdata.pl -d HOSTPERFDATA
}

# 'process-service-perfdata' command definition
define command {
       command_name                        process-service-perfdata
       command_line                        /usr/bin/perl /usr/libexec/pnp4nagios/ \
process_perfdata.pl
}
```

Starten Sie nun noch den Webserver und Nagios neu:

```
[root@moni-centos63-nagios ~]# service httpd restart
[root@moni-centos63-nagios ~]# service nagios restart
```



Keine Graphen: Wenn PNP4Nagios keine Performancedaten findet, können auch keine Graphen generiert werden. Prüfen Sie bei fehlenden Graphen zuerst, ob von den entsprechenden Check-Plugins tatsächlich Performancedaten erzeugt werden.

Nachdem Nagios erneut Tests durchgeführt hat (dies kann ein paar Minuten dauern), können Sie sich die ersten Graphen unter dem URL *http://127.0.0.1/pnp4nagios* ansehen. Nutzen Sie dabei die unter Nagios gesetzten Zugangsdaten. Die grundlegende Installa-

tion und Konfiguration von PNP4Nagios haben Sie damit abgeschlossen. Es werden nun für jedes Gerät und jeden Dienst Performancedaten verarbeitet, sofern diese von dem Plugin geliefert werden.

Integration von PNP4Nagios

Damit Sie von der Weboberfläche aus direkt auf die Graphen zugreifen können, sollten Sie die Graphen direkt mit der Anzeige von Geräten und Diensten verzahnen. Passen Sie dazu die Vorlagen in der Datei */etc/nagios/objects/templates.cfg* an und referenzieren Sie diese Definitionen dann in den vorhandenen Definitionen `generic-host` und `generic-service` (siehe hierzu Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*, Rezept 4.1, *Maschinen einbinden (host)* für die Definition von Maschinen, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* für die Definition von Services und Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System* für die Verwendung des Vorlagensystems):

```
[...]
# HOST TEMPLATES
[...]
define host {
       name                                    generic-host
       use                                     pnp-hst
[...]
define host {
       name                                    pnp-hst
       register                                0
       action_url                              /pnp4nagios/graph?host=$HOSTNAME$' \
class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=_HOST_
}
[...]
# SERVICE TEMPLATES
[...]
define service {
       name                                    generic-service
       use                                     pnp-svc
[...]
define service {
       name                                    pnp-svc
       register                                0
       action_url                              /pnp4nagios/graph?host=$HOSTNAME$&srv=$ \
SERVICEDESC$' class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=$SERVICEDESC$
}
[...]
```

Graph-Integration: Hierbei werden entsprechende Icons für alle Geräte und Services definiert, auch für solche ohne Graphen. Sie haben alternativ aber auch die Möglichkeit, die Vorlage `pnp-svc` nur bei den Maschinen bzw. Diensten einzubinden, für die Sie Graphen haben oder anzeigen möchten.

Die oben als Vorlage definierte Konfigurationsoption `action_url` wird jeder Maschine und jedem Dienst ein Graphen-Symbol hinzufügen. Allerdings erst nach dem nächsten Neustart von Nagios, vor dem Sie aber noch eine weitere Anpassung vornehmen sollten. Sie können sich die Graphen nämlich bereits bei einem Darüberfahren mit der Maus (`mouse-over`) anzeigen lassen. Kopieren Sie hierzu die als Vorlage gelieferte Konfigurationsdatei *status-header.ssi* in das Verzeichnis */usr/share/nagios3/htdocs/ssi/* und passen Sie die entsprechenden Rechte wie folgt an:

```
[root@moni-centos63-nagios ~]# cp /usr/share/doc/pnp4nagios-0.6.16/contrib/ssi/status-
header.ssi /usr/share/nagios/html/ssi/
[root@moni-centos63-nagios ~]# chmod 644 /usr/share/nagios/html/ssi/status-header.ssi
```

Starten Sie nun Nagios neu, damit die geänderten Dateien eingelesen werden:

```
[root@moni-centos63-nagios ~]# service nagios restart
```

Damit weist die klassische Oberfläche nun Symbole für die Graphen auf. Beim Darüberfahren mit der Maus sehen Sie eine Vorschau, durch Klick gelangen Sie auf die Seite von PNP4Nagios mit dem entsprechenden Graphen. Für den weiteren Umgang mit PNP4Nagios lesen Sie bitte die Rezepte zur Datenvisualisierung (Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*).

Installation des Visalisierungsaddons NagVis

Mit Hilfe des NagVis-Addons können Sie Symbole für Maschinen und Services auf einer Oberfläche beliebig anordnen. Diese erhalten durch die Wahl eines entsprechenden Hintergrundes einen Mehrwert, etwa durch die Platzierung auf Fotos oder Karten. Das Addon ist nicht Bestandteil der von uns genutzten Paketquellen. Sie können hier auf die quellbasierte Installation (siehe Rezept 1.9, *Nagios aus den Quellen installieren*) ausweichen.

Diskussion

Die mit dem Paket `nagios` installierten Komponenten nutzen die in Tabelle 1-37 aufgezeigten Pfade.

Tabelle 1-37: Ablageorte der Komponenten von Nagios

Komponente	Pfad
Programm	/usr/sbin/nagios
Haupt-Konfigurationsdatei	/etc/nagios/nagios.cfg
Plugins	/usr/lib64/nagios/plugins
Logdatei	/var/log/nagios/nagios.log
Webserver-Konfiguration	/etc/httpd/conf.d/nagios.conf
Webseiten	/usr/share/nagios/html/

Das Addon NDOUtils verwendet die in Tabelle 1-38 aufgeführten Dateien und Ordner.

Tabelle 1-38: Pfade der wichtigsten Dateien von NDOUtils

Komponente	Pfad
Konfigurationsdatei ndo2db	/etc/nagios/ndo2db.cfg
Konfigurationsdatei ndomod	/etc/nagios/ndomod.cfg
Debug-Logs	/var/log/ndoutils/*.debug

Das Addon PNP4Nagios benutzt die in Tabelle 1-39 aufgezeigten Installationsorte.

Tabelle 1-39: Pfade zu den Komponenten von PNP4Nagios

Komponente	Pfad
Konfigurationsdateien	/etc/pnp4nagios/
Logdateien	/var/log/pnp4nagios/
Webseiten	/usr/share/nagios/html/pnp4nagios/
Script zur Verarbeitung	/usr/libexec/pnp4nagios/process_perfdata.pl

Auf die in den Nagiosplugins enthaltenen Plugins werden wir in den Rezepten der Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen* und Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen* genauer eingehen. Für die Überwachung von System-Updates können Sie das Plugin `check_updates` aus dem Paket `nagios-plugins-check-updates` nutzen. Es stehen Ihnen auch noch diverse anderen Plugins mit der Paketbezeichnung `nagios-plugins-*` zur Verfügung. Als weiteres Addon können Sie NRPE (Nagios Remote Plugin Executor) installieren (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*).

Siehe auch

- Rezept mit Entscheidungshilfe zur Installation: Rezept 1.1, *Entscheidungshilfe für die Installation*
- Webseite des Nagios-Projekts: *http://www.nagios.org//*
- Webseite von Nagiosplugins: *http://nagiosplugins.org//*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Webseite von CentOS: *http://www.centos.org/*
- Rezept zur quellenbasierten Installation von Nagios: Rezept 1.9, *Nagios aus den Quellen installieren*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zum Aufbau und strukturierter Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*

- Kapitel zur Konfiguration von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere das Rezept 4.3, *Befehle hinzufügen (command)*, Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* sowie Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Kapitel zur Datenvisualisierung mit PNP4Nagios und NagVis: Kapitel 9, *Datenvisualisierung mit PNP4Nagios und NagVis*
- Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezepte zur Installation von NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

1.14 Migrationen und Aktualisierungen von Nagios/Icinga

Problem

Sie möchten

- Nagios/Icinga aktualisieren (Aktualisierung) oder
- von dem einen zum anderen wechseln (Migration) und/oder
- von einer alten Maschine auf eine neue migrieren (Migration).

Lösung

In allen genannten Fällen benötigen Sie zunächst Sicherungskopien aller relevanten Konfigurationsdateien und angepassten Dateien (siehe Rezept 3.9, *Sicherungen der Konfigurationsdateien anlegen*). Sofern beide Installationen auf einer Maschine sind, ist dies besonders wichtig, da Sie hierbei eventuell vorhandene Dateien überschreiben. Migrieren Sie hingegen auf eine neue Maschine, können Sie die alte laufen lassen, bis Sie das neue Monitoring-System zu Ihrer Zufriedenheit eingerichtet haben.

Generell sollten Sie die Konfiguration dabei als eine zweigeteilte betrachten. Zum einen die Konfigurationsdateien für das Nagios/Icinga-System selbst, also die Einstellungen zu Datenbankanbindung, die Webserver-Konfiguration, der Pfad zu den Plugins, die Plugins selbst und so weiter. Wir werden diesen Teil im Folgenden System-Konfiguration nennen. Zum anderen sind die Dateien mit den Angaben zu überwachender Maschinen und Diensten zu nennen, die wir Monitoring-Konfiguration nennen.

Bei kleineren Aktualisierungen von Nagios/Icinga, also bei Versionsprüngen, bei denen die erste Zahl der Versionsnummer gleich bleibt, müssen Sie in der Regel nur die neue Version über die alte installieren und keine weiteren Änderungen vornehmen. Beachten Sie beim Installieren neuer Versionen aber immer die Installations- und gegebenenfalls Aktualisierungshinweise, insbesondere Datenbankschema-Aktualisierungen. Sollten bei einer Aktualisierung einmal Änderungen an den Konfigurationsdateien nötig werden, dann wird dies dort angeführt sein. Aber selbst bei einer von uns durchgeführten Migration von Nagios Version 2 auf Version 3 waren keine Anpassungen der Monitoring-Konfiguration erforderlich. Das Format dieser Konfigurationsdateien ist bereits seit vielen Jahren das gleiche.

Ablauf

Damit vollziehen Sie Wechsel also prinzipiell immer in den folgenden Schritten:

1. Icinga / Nagios auf Quell- und Zielserver beenden
   Beginnen Sie damit, Icinga / Nagios und die IDOtools / NDOtools zu beenden, beispielsweise mit `service icinga stop` und anschließend `service ido2db stop`. Dies gilt auch für eventuell installierte Addons wie PNP4Nagios und NagVis.
2. Alte Konfiguration auf dem Quellserver sichern
   Sie möchten hier vermutlich insbesondere Ihre Nagios/Icinga-Konfigurationsdateien, zusätzlich installierte und angepasste Plugins, sowie eventuell installierte Addons (siehe dazu Rezept 3.9, *Sicherungen der Konfigurationsdateien anlegen*) und eventuell auch Logs und Ihre historischen Daten übernehmen.
3. Backup entpacken
   Entpacken Sie das Backup in ein separates Verzeichnis auf dem Zielserver.
4. Alte Konfiguration auf dem Zielsystem einspielen
   Suchen Sie die erforderlichen Dateien und Verzeichnisse aus dem Backup heraus. Sichern Sie jede zu überschreibende Konfigurationsdatei auf dem Zielsystem (beispielsweise mit dem Kommando `mv DATEINAME DATEINAME.ORIG`) und kopieren Sie erst anschließend Dateien aus dem Backup schrittweise an die jeweils benötigte Stelle. Prüfen Sie, ob es grundsätzliche Veränderungen in den Konfigurationsdateien gibt, so dass Sie diese manuell anpassen müssen. Dies wird bei den Hauptkonfigurationsdateien *icinga.cfg* / *nagios.cfg* beziehungsweise ido2db.cfg / ndo2db.cfg meistens der Fall sein. Prüfen Sie auch, ob Anpassungen an Ihrer *resource.cfg* notwendig sind.
5. Neues System testen
   Planen Sie einen ausführlichen Test ein, um gegebenenfalls auftretende Probleme zu erkennen und zu beseitigen. Betreiben Sie das alte System solange weiter, bis Ihr neues Monitoring-System zu Ihrer Zufriedenheit läuft.

Bei einem entsprechend modifizierten System und bei einer ausgelagerten Datenbank können sich diese Schritte allerdings immer noch als recht komplex erweisen.

Anpassung vs. Übernahme

Eine passende System-Konfiguration wird bei der Installation eines neuen Systems (auf Wunsch) erstellt. Entsprechend erhalten Sie also im Zweifel immer zunächst ein lauffähiges System, in dem Sie dann aber gegebenenfalls Ihre spezifischen Änderungen nachpflegen müssen. Ob es einfacher ist, Ihre vorherige System-Konfiguration an das neue System anzupassen oder die von Ihnen vorgenommenen Änderungen nochmals in der neuen Konfiguration durchzuführen, hängt von dem Versionssprung und den von Ihnen durchgeführten Anpassungen ab.

Bei einer Migration zwischen den beiden Alternativen Nagios und Icinga müssen Sie insbesondere beachten, dass diese an vielen Stellen der System-Konfiguration unterschiedliche Standardwerte verwenden. Dies ist etwa bei den verwendeten Benutzernamen (`nagiosadmin` im Gegensatz zu `icingaadmin`) und Verzeichnisnamen (*/etc/nagios* im Gegensatz zu */etc/icinga*) sowie Dateinamen (*nagios.cfg* im Gegensatz zu *icinga.cfg*) der Fall. Da diese dann wieder häufig innerhalb der Konfigurationsdateien referenziert werden, bestehen entsprechend viele Unterschiede zwischen den Konfigurationsdateien beider Systeme. Typischerweise sind die Anpassungen an der System-Konfiguration aber nicht sehr zahlreich. Deshalb ist es insbesondere bei einer Migration zwischen Nagios/Icinga häufig einfacher, die von Ihnen vorgenommenen Änderungen in der neuen Konfiguration erneut durchzuführen. Aufgrund der erwähnten Umbenennungen müssen Sie hier sonst mit hunderten von Änderungen rechnen.

Änderungen die Sie möglicherweise an der System-Konfiguration vorgenommen haben, könnten insbesondere folgende sein:

- Anpassungen in der *icinga.cfg* / *nagios.cfg* beziehungsweise *ido2db.cfg* / *ndo2db.cfg*
  Zwar haben Sie mit der bei einer Installation generierten Konfiguration wieder ein lauffähiges System inklusive der mit-eingerichteten Erweiterungen, aber sämtliche Einstellungen sind dabei auf die (gegebenenfalls neuen) Standardwerte gesetzt worden. Wenn Sie hier Anpassungen vorgenommen haben, sollten Sie die entsprechenden Optionen prüfen und gegebenenfalls wieder anpassen. Dies kann insbesondere die Protokollierung, die Pfade zur Monitoring-Konfiguration, die Verwendung der eingebetteten Perl-Umgebung und Performance-Optionen betreffen.
- Protokolldateien und historische Daten
  Hier sollten Sie eine Überprüfung auf Logs (Reporting-Funktionalität) sowie historische Daten (Statehistory, Downtimes und Scheduling-Informationen in der Datei *retention.dat*) durchführen.
- Anpassungen in der *resource.cfg*
  Wenn Sie eigene Benutzer-Makros verwenden, müssen Sie sicherstellen, dass diese in der neuen Version verfügbar sind. Dabei kann es sich beispielsweise um hinterlegte Passwörter oder zusätzlich eingerichtete Plugin-Verzeichnisse handeln. Falls Sie Ihre alte Konfigurationsdatei übernehmen, achten Sie insbesondere darauf, ob sich der typischerweise hier hinterlegte Pfad zu den Plugins geändert hat.

- Zusätzlich installierte Plugins
  Wenn Sie neben den Standard-Plugins weitere Plugins installiert haben, müssen Sie diese gegebenenfalls auch auf dem neuen System bereitstellen.

Da Sie prinzipiell überall Änderungen vornehmen können, gibt es entsprechend viele Möglichkeiten, welche Aspekte hier beachtet werden müssen. Wenn Sie beispielsweise die zur Anzeige genutzten HTML-Seiten angepasst haben, dann müssen Sie diese auch hier berücksichtigen. Es können aber auch externe Komponenten betroffen sein, wie beispielsweise die Datensicherung (siehe hierzu Rezept 3.9, *Sicherungen der Konfigurationsdateien anlegen*).

Diskussion

Die Migration der eigentlichen Kernaufgabe kann Ihnen also wie dargestellt unter Umständen einige Arbeit bereiten, dürfte aber ansonsten kein Problem darstellen. Problematisch ist allerdings in einigen Fällen die Übernahme von gespeicherten Daten. Wenn Sie etwa von einem Graph-Addon auf ein anderes migrieren, werden Sie Ihre alten Daten unter Umständen nur über mühselige und sehr versionsspezifische Handarbeit in das neue System bekommen. Prüfen Sie hierzu zunächst, ob vielleicht ein entsprechendes Migrations-Script oder etwas Ähnliches zur Verfügung steht.

Ansonsten können Sie auch überlegen die Alt-Daten im alten System parallel vorzuhalten. Dazu lassen Sie das alte System laufen, bis Sie ausschließlich auf das neue System umsteigen können. Wenn Sie einen solchen Parallel-Betrieb planen, sollten Sie im alten System allerdings die Prüf-Intervalle auf hohe Werte setzen. Anderenfalls erzeugen die zwei parallel laufenden Monitoring-Systeme unnötige Last und verfälschen damit auch die erhobenen Daten.

Siehe auch

- Rezept zur Sicherung der Konfigurationsdateien: Rezept 3.9, *Sicherungen der Konfigurationsdateien anlegen*

1.15 Icinga unter Debian deinstallieren

Problem

Sie haben nach Rezept 1.5, *Icinga unter Debian installieren* Icinga auf Ihrem Debian System installiert. Nun wollen Sie dieses und möglicherweise zusätzlich installierte Addons wieder von Ihrem System entfernen.

Lösung

Die Deinstallation von Icinga erfolgt analog zur Installation, nur in umgekehrter Reihenfolge.

apt-get autoremove: Zwischen den Alternativen `apt-get` und `aptitude` besteht bezüglich der Deinstallation ein erwähnenswerter Unterschied: `aptitude` entfernt automatisch alle abhängigen Pakete, die nicht noch von einem anderen installierten Paket abhängig sind. Das Kommando `apt-get` wird diesen Paketen hingegen nur eine entsprechende Markierung hinzufügen, die Deinstallation der so markierten Pakete erfolgt allerdings erst beim nächsten Aufruf von `apt-get` (gegegebenfalls manuell direkt im Anschluss als `apt-get autoremove`).

Als erstes deinstallieren Sie NagVis, PNP4Nagios und IDOUtils mit Hilfe folgender Anweisung:

```
root@moni-wheezy-icinga:~# aptitude purge nagvis pnp4nagios icinga-idoutils
```

Die bei der Installation automatisch installierten Pakete werden nun auch automatisch wieder deinstalliert. Während der Deinstallation werden Sie gefragt, ob Sie die Datenbank und die darin gespeicherten Daten ebenfalls löschen möchten. Wenn Sie neben den Programmen auch die Daten löschen möchten, bestätigen Sie dies und geben Sie anschließend Ihr Root-Passwort für MySQL ein. Nun können Sie mit der Deinstallation von MySQL fortfahren:

```
root@moni-wheezy-icinga:~# aptitude purge mysql-server
```

Entfernen Sie dann Icinga selbst:

```
root@moni-wheezy-icinga:~# aptitude purge icinga
```

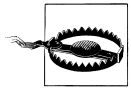
Wenn Sie den Apache2-Webserver ebenfalls nicht mehr benötigen, können Sie auch diesen abschließend entfernen:

```
root@moni-wheezy-icinga:~# aptitude remove apache2
```

Damit wurde ein Großteil der installierten Pakete wieder von Ihrem System entfernt.

Diskussion

Die mittels des Paketmanagements bequem installierten Anwendungen lassen sich auf ebenso bequeme Art und Weise wieder deinstallieren.



remove/purge: Bei der Verwendung von `purge` werden auch Konfigurationsdateien entfernt, so dass sämtliche vorgenommenen Anpassungen potentiell verloren sind. Entsprechend müssen dann bei einer Neuinstallation alle Einstellungen, wie beispielsweise Zugangsdaten zur Datenbank, erneut vorgenommen werden. Soll anschließend wieder ein System installiert werden, bietet sich deshalb die Operation `remove` statt `purge` an (wird sowohl von `apt-get` als auch von `aptitude` verstanden). Dieses Vorgehen

und die Übernahme der Konfigurationsdateien wird im Rezept zur Aktualisierung genauer beschrieben (siehe Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*).

Paketmanager-Deinstallation: Bei der Deinstallation über Paketmanager bleiben neben Konfigurationsdateien regelmäßig ganze Pakete zurück. Dies betrifft alle Pakete, die durch andere verbleibende Pakete als »empfohlen« referenziert werden. Wenn wirklich alle Pakete wieder entfernt werden sollen, ist ein manueller Abgleich nötig. Wenn Sie bei der Installation die Liste der mit-installierten Pakete abgespeichert haben, können Sie hier gleich die ganze Liste zur Deinstallation übergeben.

Siehe auch

- Webseite des Icinga-Projekts: *https://www.icinga.org/*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Webseite von NagVis: *http://www.nagvis.org/*
- Rezepte zur Installation von Icinga unter Debian: Rezept 1.5, *Icinga unter Debian installieren*
- Rezept zur Migration und Aktualisierung von Nagios/Icinga: Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*

1.16 Icinga unter Ubuntu deinstallieren

Problem

Sie möchten ein nach Rezept 1.6, *Icinga unter Ubuntu installieren* über das Paketmanagement-System installiertes Icinga und möglicherweise ebenfalls installierte Addons wieder vom System entfernen .

Lösung

Die Deinstallation erfolgt analog zur Installation, jedoch gegebenenfalls in umgekehrter Reihenfolge.

`apt-get autoremove`: Zwischen den Alternativen `apt-get` und `aptitude` besteht bezüglich der Deinstallation ein erwähnenswerter Unterschied: `aptitude` entfernt automatisch alle abhängigen Pakete, die nicht noch von einem anderen installierten Paket abhängig sind. Das Kommando `apt-get` wird diesen Paketen hingegen nur eine entsprechende Markierung hinzufügen, die Deinstallation der so markierten Pakete erfolgt allerdings erst beim nächsten Aufruf von `apt-get` (gegegebenfalls manuell direkt im Anschluss als `apt-get autoremove`).

Zunächst entfernen Sie alle Addons. Wenn Sie PNP4Nagios installiert haben, deinstallieren Sie dieses mit folgender Anweisung:

```
root@moni-oneiric-icinga:~# aptitude purge pnp4nagios
```

Sofern Sie die Datenbankanbindung mit den IDOUtils installieren haben, entfernen Sie nun das Paket `icinga-idoutils`:

```
root@moni-oneiric-icinga:~# aptitude purge icinga-idoutils
```

Das Paket stellt dabei einen Assistenten zur Verfügung, mit dem sich die Datenbank entfernen lässt. Sofern Sie diese nicht mehr benötigen, geben Sie das während der Installation vergebene Passwort für den MySQL-Benutzer `root` ein. Im Zweifel verwenden Sie die vom Assistenten angebotenen Vorgaben.

Im nächsten Schritt deinstallieren Sie den MySQL-Server, sofern Sie diesen nicht anderweitig benötigen:

```
root@moni-oneiric-icinga:~# aptitude purge mysql-server
```

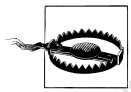
Schließlich entfernen Sie das Icinga selbst:

```
root@moni-oneiric-icinga:~# aptitude purge icinga
```

Damit sind viele der installierten Pakete wieder vom System entfernt.

Diskussion

Die mittels des Paketmanagements bequem installierten Anwendungen lassen sich auf ebenso bequeme Art und Weise wieder Deinstallieren.



`remove/purge`: Bei der Verwendung von `purge` werden auch Konfigurationsdateien entfernt, so dass sämtliche vorgenommenen Anpassungen potentiell verloren sind. Entsprechend müssen dann bei einer Neuinstallation alle Einstellungen, wie beispielsweise Zugangsdaten zur Datenbank, erneut vorgenommen werden. Wenn anschließend wieder ein System installiert werden soll, bietet sich daher die Operation `remove` anstelle von `purge` an (wird sowohl von apt-get als auch von aptitude verstanden). Dieses Vorgehen und die Übernahme der Konfigurationsdateien wird im Rezept zur Aktualisierung genauer beschrieben (siehe Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*).

Siehe auch

- Webseite des Icinga-Projekts: *https://www.icinga.org/*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Rezept zur Installation von Icinga unter Debian: Rezept 1.6, *Icinga unter Ubuntu installieren*
- Rezept zur Migration und Aktualisierung von Nagios/Icinga: Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*

1.17 Icinga unter SLES|openSUSE deinstallieren

Problem

Sie haben Icinga auf SLES|openSUSE nach den Installationsanweisungen von Rezept 1.7, *Icinga unter SLES|openSUSE installieren* installiert und wollen dieses wieder deinstallieren.

Lösung

Im Folgenden zeigen wir Ihnen, wie Sie Icinga und alle mitinstallierten Pakete, sofern diese nicht zwischenzeitlich von anderen Paketen benötigt werden, wieder deinstallieren können.

YAST versus Zypper: Unter openSUSE steht neben dem systemeigenen graphischen Konfigurationstool YAST auch der Paketmanager Zypper zur Verfügung. Beide Tools greifen auf dieselbe Datenbasis zu und können somit alternativ genutzt werden. Im Gegensatz zu Yast deinstalliert Zypper automatisch auch abhängige Pakete, weshalb wir hier mit Zypper arbeiten.

Sofern auch die Datenbankanbindung installiert ist, deinstallieren Sie zunächst die IDOUtils:

```
moni-suse-icinga:~ # zypper remove --clean-deps icinga-idoutils-mysql
```

Falls Sie den MySQL-Server nicht anderweitig benötigen, können Sie diesen anschließend mit folgender Anweisung deinstallieren:

```
moni-suse-icinga:~ # zypper remove --clean-deps mysql-community-server
```

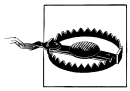
Nun deinstallieren Sie Icinga mit folgender Anweisung:

```
moni-suse-icinga:~ # zypper remove --clean-deps icinga
```

Alle wesentlichen Komponenten sind damit entfernt.

Diskussion

Die mittels des Paketmanagements bequem installierten Anwendungen lassen sich auch auf ebenso bequeme Art und Weise wieder deinstallieren.



RPM-Überbleibsel: Durch die Installation angelegte Verzeichnisse, Konfigurationdateien und Scripte werden nicht restlos entfernt. Modifizierte Dateien erhalten beispielsweise allesamt die Endung `.rpmsave` und verbleiben. Außerdem wird die angelegte Datenbank nicht gelöscht. Die betreffenden Dateien müssen dann bei Bedarf manuell gesucht und gelöscht werden.

Siehe auch

- Webseite des Icinga-Projekts: *https://www.icinga.org/*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Rezept zur Installation von Icinga unter openSUSE: Rezept 1.7, *Icinga unter SLES|openSUSE installieren*
- Rezept zur Migration und Aktualisierung von Nagios/Icinga: Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*

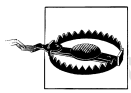
1.18 Icinga unter RHEL|CentOS|Fedora deinstallieren

Problem

Sie möchten ein nach Rezept 1.8, *Icinga unter RHEL|CentOS|Fedora installieren* installiertes Icinga wieder deinstallieren.

Lösung

Wir zeigen Ihnen im Folgenden die Deinstallation von Icinga. Der Paketmanager Yum entfernt bei der Deinstallation nur die Pakete automatisch mit, die ohne die zu entfernenden Pakete nicht mehr funktionsfähig wären. Dadurch werden im Zweifel eher viele Pakete auf Ihrem System verbleiben. Wir zeigen Ihnen hier deshalb an einen Beispiel die Deinstallation über Angabe aller Paketnamen .



Deinstallation mit Yum: Sofern Sie sich bei der Installation die Liste der mitinstallierten Pakete kopiert haben, können Sie diese verwenden, um wirklich alle Pakete wieder zu entfernen. Problemlos ist dies allerdings nur dann möglich, wenn Sie nach der Installation dieser Software nicht noch weitere Pakete installiert haben. Anderenfalls werden unter Umständen Teile dieser weiteren Installationen ebenfalls deinstalliert. Sie können dies jedoch leicht feststellen, da Ihnen Yum aufgrund von Abhängigkeiten zu deinstallierende Pakete gesondert und vor der Sicherheitsabfrage anzeigt:

```
[...]
Removing:
[...]
Removing for dependencies:
[...]
```

Der Abschnitt »Removing for dependencies:« ist ansonsten nicht vorhanden. Achten Sie darauf!

Bevor Sie mit der Deinstallation beginnen, stoppen Sie bitte die auf Ihrem System laufenden Dienste:

```
[root@moni-centos63-icinga ~]# service icinga stop
[root@moni-centos63-icinga ~]# service ido2db stop
[root@moni-centos63-icinga ~]# service mysqld stop
```

Wenn Sie sich bei der Installation die Liste der Pakete kopiert haben, können Sie diese hier verwenden, um auch alle Pakete wieder zu entfernen. Starten Sie mit der Deinstallation des MySQL-Servers und der IDOUtils, hier als Beispiel anhand der Paketliste aus der Installation:

```
[root@moni-centos63-icinga ~]# yum remove mysql mysql-server perl-DBD-MySQL perl-DBI
icinga-idoutils-libdbi-mysql libdbi-dbd-mysql
```

Deinstallieren Sie dann das Paket `nagios-plugins` und die mit diesem installierten Pakete, wieder anhand der Paketliste aus der Installation:

```
[root@moni-centos63-icinga ~]# yum remove nagios-plugins fping mysql-libs
```

Anschließend deinstallieren Sie außerdem PHP und Icinga. :

```
[root@moni-centos63-icinga ~]# yum remove icinga apr apr-util apr-util-ldap
fontconfig gd httpd httpd-tools libX11 libX11-common libXau libXpm libpng
libxcb mailcap icinga-common icinga-gui icinga-doc
```

Damit haben Sie dann alle Pakete deinstalliert, die Sie ursprünglich installiert hatten.

Diskussion

Um alle installierten Pakete wieder zu deinstallieren, ist es erforderlich, auch jedes Paket wieder für die Deinstallation anzugeben.

Siehe auch

- Webseite des Icinga-Projekts: *https://www.icinga.org/*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Rezept zur Installation von Icinga unter CentOS: Rezept 1.8, *Icinga unter RHEL|CentOS|Fedora installieren*

1.19 Nagios unter Debian deinstallieren

Problem

Sie möchten ein über das Paketmanagement-System nach Rezept 1.10, *Nagios unter Debian installieren* installiertes Nagios wieder vom System entfernen.

Lösung

Im Folgenden zeigen wir Ihnen, wie Sie Nagios und alle mitinstallierten Pakete und Addons, sofern diese nicht zwischenzeitlich von anderen Paketen benötigt werden, wieder deinstallieren können. Die Deinstallation verläuft analog zur Installation, im Wesentlichen jedoch in umgekehrter Reihenfolge.

apt-get autoremove: Zwischen den Alternativen `apt-get` und `aptitude` besteht bezüglich der Deinstallation ein erwähnenswerter Unterschied: `aptitude` entfernt automatisch alle abhängigen Pakete, die nicht noch von einem anderen installierten Paket abhängig sind. Das Kommando `apt-get` wird diesen Paketen hingegen nur eine entsprechende Markierung hinzufügen, die Deinstallation der so markierten Pakete erfolgt allerdings erst beim nächsten Aufruf von `apt-get` (gegebenenfalls manuell direkt im Anschluss als `apt-get autoremove`).

Deinstallieren Sie zunächst alle installierten Addons.

Sofern Sie wie beschrieben NagVis, PNP4Nagios und die NDOUtils installiert haben, deinstallieren Sie zunächst dann diese.

```
root@moni-wheezy-nagios:~# aptitude purge nagvis pnp4nagios ndoutils-nagios3-mysql
```

Das Paketmanagent-System erkennt, welche Pakete automatisch mit-installiert wurden und entfernt diese auch automatisch wieder. Bei der Deinstallation werden einige Fragen dazu gestellt, ob die erstellte NDOUtils-Datenbank und alle darin gespeicherten Daten gelöscht werden sollen. Dabei muss das Root-Passwort für MySQL eingegeben werden. Wenn Nagios vollständig gelöscht werden soll, können Sie dies entsprechend bestätigen. Im Zweifelsfall belassen sie Sie es hier bei den vorgegebenen Einstellungen.

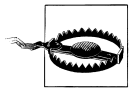
Anschließend können Sie Nagios durch folgende Anweisung deinstallieren:

```
root@moni-wheezy-nagios:~# aptitude purge nagios3
```

Damit sind alle wesentlichen Komponenten wieder entfernt.

Diskussion

Die mittels des Paketmanagements bequem installierten Anwendungen lassen sich auch auf ebenso bequeme Weise wieder deinstallieren.



remove/purge: Bei der Verwendung von `purge` werden auch Konfigurationsdateien entfernt, so dass sämtliche vorgenommenen Anpassungen potentiell verloren sind. Entsprechend müssen dann bei einer Neuinstallation alle Einstellungen, wie beispielsweise Zugangsdaten zur Datenbank, erneut vorgenommen werden. Wenn anschließend wieder ein System installiert werden soll, bietet sich deshalb die Operation `remove` anstelle von `purge` an (wird sowohl von apt-get als auch von aptitude verstanden). Dieses Vorgehen und die Übernahme der Konfigurationsdateien wird im Rezept zur Aktualisierung genauer beschrieben (siehe Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*).

Paketmanager-Deinstallation: Bei der Deinstallation über Paketmanager bleiben neben Konfigurationsdateien regelmäßig ganze Pakete zurück. Dies betrifft alle Pakete, die durch andere verbleibende Pakete als »empfohlen« referenziert werden. Wenn wirklich alle Pakete wieder entfernt

werden sollen, ist ein manueller Abgleich nötig. Wenn Sie bei der Installation die Liste der mit-installierten Pakete abgespeichert haben, dann können Sie hier gleich die ganze Liste zur Deinstallation übergeben.

Siehe auch

- Webseite des Nagios-Projektes: *http://www.nagios.org/*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Webseite von NagVis: *http://www.nagvis.org/*
- Rezept zur Installation von Nagios unter Debian: Rezept 1.10, *Nagios unter Debian installieren*
- Rezept zur Migration und Aktualisierung von Nagios/Icinga: Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*

1.20 Nagios unter Ubuntu deinstallieren

Problem

Sie möchten ein über das Paketmanagement-System installiertes Nagios (siehe Rezept 1.11, *Nagios unter Ubuntu installieren*) wieder von Ihrem System entfernen.

Lösung

Die Deinstallation verläuft analog zur Installation, jedoch gegebenenfalls in umgekehrter Reihenfolge.

`apt-get & aptitude`: Als Benutzerschnittstellen zum eigentlichen Paketmanager `dpkg` stehen Ihnen sowohl `apt-get` als auch `aptitude` zur Verfügung, die funktional adäquat sein sollten. Wir verwenden hier aus Gründen der Einheitlichkeit stets `aptitude`.

Zunächst deinstallieren Sie alle Addons. Mit folgendem Befehl deinstallieren Sie Nagios-Grapher:

```
root@moni-lucid-nagios:~# aptitude purge nagiosgrapher
```

Sofern Sie wie beschrieben NagVis, PNP4Nagios und NDOUtils installiert haben, deinstallieren Sie zunächst dann diese.

```
root@moni-pangolin-icinga:~# aptitude purge nagvis pnp4nagios ndoutils-nagios3-mysql
```

Das Paketmanagent-System erkennt, welche Pakete automatisch mit-installiert wurden und entfernt diese auch automatisch wieder. Bei der Deinstallation werden einige Fragen dazu gestellt, ob die erstellte NDOUtils-Datenbank und alle darin gespeicherten Daten

gelöscht werden sollen. Dabei muss das Root-Passwort für MySQL eingegeben werden. Wenn Nagios vollständig gelöscht werden soll, können Sie dies bestätigen. Im Zweifelsfall belassen Sie es hier bei den vorgegebenen Einstellungen.

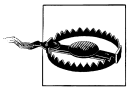
Anschließend können Sie Nagios deinstallieren:

```
root@moni-lucid-nagios:~# aptitude purge nagios3
```

Damit sind alle wesentlichen Komponenten wieder entfernt.

Diskussion

Die mittels des Paketmanagements bequem installierten Anwendungen lassen sich auch ebenso bequem wieder deinstallieren.



`remove/purge`: Bei der Verwendung von `purge` werden auch Konfigurationsdateien entfernt, so dass sämtliche vorgenommenen Anpassungen potentiell verloren sind. Entsprechend müssen dann bei einer Neuinstallation alle Einstellungen, wie beispielsweise Zugangsdaten zur Datenbank, erneut vorgenommen werden. Wenn anschließend wieder ein System installiert werden soll, bietet sich deshalb die Operation `remove` anstelle von `purge` an (wird sowohl von apt-get als auch von aptitude verstanden). Dieses Vorgehen und die Übernahme der Konfigurationsdateien wird im Rezept zur Aktualisierung genauer beschrieben (siehe Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*).

Paketmanager-Deinstallation: Bei der Deinstallation über Paketmanager bleiben neben Konfigurationsdateien regelmäßig ganze Pakete zurück. Dies betrifft alle Pakete, die durch andere verbleibende Pakete als »empfohlen« referenziert werden. Falls wirklich alle Pakete wieder entfernt werden sollen, ist ein manueller Abgleich nötig. Wenn Sie bei der Installation die Liste der mit-installierten Pakete abgespeichert haben, können Sie hier gleich die ganze Liste zur Deinstallation übergeben.

Siehe auch

- Webseite des Nagios-Projektes: *http://www.nagios.org/*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Webseite von PNP4Nagios: *http://www.pnp4nagios.org/*
- Webseite von NagVis: *http://www.nagvis.org/*
- Rezept zur Installation von Nagios unter Ubuntu: Rezept 1.11, *Nagios unter Ubuntu installieren*
- Rezept zur Migration und Aktualisierung von Nagios/Icinga: Rezept 1.14, *Migrationen und Aktualisierungen von Nagios/Icinga*

1.21 Nagios unter SLES|openSUSE deinstallieren

Problem

Sie wollen das über das Paketmanagement-System installierte Nagios nach Rezept 1.12, *Nagios unter SLES|openSUSE installieren* unter openSUSE wieder deinstallieren.

Lösung

Im Folgenden zeigen wir Ihnen, wie Sie Nagios und alle mit-installierten Pakete, sofern diese nicht zwischenzeitlich von anderen Paketen benötigt werden, wieder deinstallieren können.

YAST versus Zypper: Unter openSUSE steht neben dem systemeigenen graphischen Konfigurationstool YAST auch der Paketmanager Zypper zur Verfügung. Beide Tools greifen auf dieselbe Datenbasis zu und können somit alternativ genutzt werden. Im Gegensatz zu Yast deinstalliert Zypper automatisch auch abhängige Pakete, weshalb wir hier mit Zypper arbeiten.

```
moni-suse-nagios:~ # zypper remove --clean-deps ndoutils
```

Sofern Sie den MySQL-Server ebenfalls mitinstalliert hatten und nicht anderweitig benötigen, können Sie diesen als Nächstes entfernen:

```
moni-suse-nagios:~ # zypper remove --clean-deps mysql-community-server
```

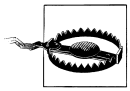
Abschließend deinstallieren Sie Nagios:

```
moni-suse-nagios:~ # zypper remove --clean-deps nagios
```

Damit sind alle wesentlichen Komponenten wieder entfernt.

Diskussion

Die mittels des Paketmanagements bequem installierten Anwendungen lassen sich auch auf ebenso bequeme Art und Weise wieder deinstallieren.



RPM-Überbleibsel: Durch die Installation angelegte Verzeichnisse, Konfigurationdateien und Scripte werden nicht restlos entfernt. Modifizierte Dateien erhalten beispielsweise allesamt die Endung `.rpmsave` und bleiben erhalten. Außerdem wird die angelegte Datenbank nicht gelöscht. Die entsprechenden Dateien müssen dann bei Bedarf manuell gesucht und gelöscht werden.

Siehe auch

- Webseite des Nagios-Projektes: *http://www.nagios.org/*

- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Rezept zur Installation von Nagios unter openSUSE: Rezept 1.12, *Nagios unter SLES|openSUSE installieren*

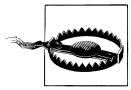
1.22 Nagios unter RHEL|CentOS|Fedora deinstallieren

Problem

Sie möchten ein nach Rezept 1.13, *Nagios unter RHEL|CentOS|Fedora installieren* installiertes Nagios und möglicherweise ebenfalls installierte Addons wieder von Ihrem System entfernen.

Lösung

Im Folgenden zeigen wir Ihnen die Deinstallation von Nagios. Der Paketmanager Yum entfernt bei der Deinstallation nur die Pakete automatisch mit, die ohne die zu entfernenden Pakete nicht mehr funktionsfähig wären. Dadurch werden im Zweifel eher viele Pakete auf Ihrem System verbleiben. Wir zeigen Ihnen hier deshalb die Deinstallation über Angabe aller Paketnamen .



Deinstallation mit Yum: Sofern Sie sich bei der Installation die Liste der mitinstallierten Pakete kopiert haben, können Sie diese verwenden, um wirklich alle Pakete wieder zu entfernen. Problemlos ist dies jedoch nur dann möglich, wenn Sie nach der Installation dieser Software nicht noch weitere Pakete installiert haben. Anderenfalls werden unter Umständen Teile dieser weiteren Installationen ebenfalls deinstalliert. Sie können dies jedoch leicht feststellen, daIhnen Yum aufgrund von Abhängigkeiten zu deinstallierende Pakete gesondert und vor der Sicherheitsabfrage anzeigt

```
[...]
Removing:
[...]
Removing for dependencies:
[...]
```

Der Abschnitt »Removing for dependencies:« ist ansonsten nicht vorhanden. Achten Sie darauf!

Bevor Sie mit der Deinstallation beginnen, stoppen Sie bitte die auf Ihrem System laufenden Dienste:

```
[root@moni-centos63-nagios ~]# service nagios stop
[root@moni-centos63-nagios ~]# service ndo2db stop
[root@moni-centos63-nagios ~]# service mysqld stop
```

Wenn Sie sich bei der Installation die Liste der Pakete kopiert haben, können Sie diese hier verwenden, um auch alle Pakete wieder zu entfernen. Starten Sie mit PNP4Nagios, sofern Sie dieses installiert hatten:

```
[root@moni-centos63-nagios ~]# yum remove pnp4nagios cairo dejavu-fonts-common
dejavu-lgc-sans-mono-fonts dejavu-sans-mono-fonts fontpackages-filesystem
libXft libXrender libthai pango php-gd pixman rrdtool rrdtool-perl
```

Da nur die ausgewählten Pakete deinstalliert werden, können Sie hier die folgende Abfrage nach der Deinstallation bestätigen . Als Nächstes können Sie den MySQL-Server deinstallieren, in unserem Beispiel wieder anhand der Paketliste aus der der Installation:

```
[root@moni-centos63-nagios ~]# yum remove mysql mysql-server perl-DBD-MySQL perl-DBI
```

Wieder werden lediglich die angegebenen Pakete deinstalliert. Als Nächstes deinstallieren Sie die NDOUtils und die damit installiert Pakete, im Beispiel erneut anhand der Paketliste aus der Installation:

```
[root@moni-centos63-nagios ~]# yum remove ndoutils ndoutils-mysql
```

Deinstallieren Sie dann das Paket `nagios-plugins` und die mit diesem installierten Pakete, wieder anhand der Paketliste aus der Installation:

```
[root@moni-centos63-nagios ~]# yum remove nagios-plugins bind-libs bind-utils
dmidecode fping libgssglue libtalloc libtdb libtirpc lm_sensors lm_sensors-libs
mysql-libs net-snmp-libs net-snmp-utils postgresql-libs qstat rpcbind samba-client
samba-common samba-winbind-clients
```

Nun deinstallieren Sie weiter PHP und Nagios, auch hier wieder mit den anderen Paketen, die damit mit-installiert wurden:

```
[root@moni-centos63-nagios ~]# yum remove php php-cli php-common nagios apr apr-util
apr-util-ldap fontconfig gd httpd httpd-tools libX11 libX11-common
libXau libXpm libpng libxcb mailcap mailx nagios-common
```

Damit haben Sie dann alle Pakete deinstalliert, die Sie ursprünglich installiert hatten.

Diskussion

Um alle installierten Pakete wieder zu deinstallieren, ist es erforderlich, jedes einzelne Paket wieder für die Deinstallation anzugeben.

Siehe auch

- Webseite des Nagios-Projektes: *http://www.nagios.org/*
- Projekt-Seite der Nagios-Plugins: *http://nagiosplugins.org/*
- Rezept zur Installation von Nagios unter CentOS: Rezept 1.13, *Nagios unter RHEL|CentOS|Fedora installieren*

KAPITEL 2

Basiskonfiguration der Systeme

Nach der Basis-Installation, also nach der Installation des Monitoring-Systems selbst (siehe Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*), werden Sie sowohl die Monitoring-Maschine als auch die zu überwachenden Geräte weiter anpassen wollen. Zunächst wird dies auch teilweise notwendig sein, damit Sie die gewünschten Prüfungen nach Ihren Vorstellungen überhaupt durchführen können. Darüber hinaus möchten wir Ihnen Maßnahmen zur Absicherung der Zugriffe und zur Sicherung Ihrer Installation empfehlen.

In diesem Kapitel gehen wir zunächst auf generelle Konfigurationsschritte für beteiligte Systeme ein. Sie können auf dieses Kapitel auch durchaus erst bei Bedarf zurückgreifen. Wir verweisen Sie in späteren Rezepten auf entsprechende zugrundeliegende Rezepte.

2.1 Absichern der Webschnittstelle zum Monitoring-Server mit SSL

Problem

Sie möchten sicherstellen, dass keine Unbefugten den administrativen Netzwerkverkehr zum Monitoring-System abhören und dabei leicht an sensible Informationen (insbesondere Passwörter) gelangen. Dafür wollen Sie die Verbindungen zur Webschnittstelle Ihres Monitoring-Servers mit SSL absichern.

Lösung

In der Apache2-Standardkonfiguration findet jeglicher Verkehr zwischen dem Webserver und dem Browser-Client erst einmal unverschlüsselt statt. Um dieses Sicherheitsproblem zu beheben, beschreiben wir im Folgenden, wie Sie die Kommunikation zum Server auf verschlüsselte Verbindungen mit SSL (Secure Socket Layer) beziehungsweise HTTPs (HyperText Transfer Protocol Secure) umstellen.

Wir betrachten an dieser Stelle lediglich die Verbindungen zur Weboberfläche des Monitoring-Systems, da es insbesondere bei dieser Sinn macht, sie vom Internet aus erreichbar zu machen. Die internen Verbindungen, die Nagios/Icinga für die Durchführung von Tests und/oder die Kommunikation mit Addons (etwa Datenbank auf eigener Maschine) aufbaut, sollten nur dann unverschlüsselt bleiben, sofern diese über private Management-Netze geleitet werden (siehe hierzu Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*). Wenn Sie allerdings nicht ausschliessen können, dass der Netzwerkverkehr über öffentliche Netze geleitet wird, dann sollten Sie alle Punkt-zu-Punkt Verbindungen von Anfang an verschlüsselt anlegen.

Im Folgenden zeigen wir Ihnen am Beispiel des standardmäßig genutzten Webservers Apache2, wie Sie die Verbindungen zu den Weboberflächen von Nagios/Icinga beziehungsweise deren Addons, wie beispielsweise Icinga-Web, PNP4Nagios und NagVis, verschlüsseln lassen können.

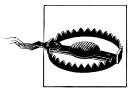
SSL-Zertifikat erstellen

Ein Webserver benötigt zur Unterstützung von SSL ein RSA-Schlüsselpaar (RSA steht für Rivest, Shamir und Adleman) und ein Zertifikat für den öffentlichen Schlüssel. Das SSL-Zertifikat muss von einer Zertifizierungsstelle, der Certification Authority (CA), ausgestellt worden sein, deren öffentlicher Signierungsschlüssel sich bereits in den Standardbrowsern befindet.

In Ermangelung eines entsprechenden Zertifikats kann aber jederzeit auch eine eigene Zertifizierungsstelle erstellt und verwendet werden. Die Schritte hierzu beschreiben wir Ihnen im Folgenden. Das resultierende Zertifikat wird es Ihnen ermöglichen, die Verbindungen zu verschlüsseln, es wird dabei allerdings zu Warnungen im Browser führen, da die selbst erstellte Zertifizierungsstelle dem Browser unbekannt ist.

CACert: Wenn Sie noch keine Zertifizierungsstelle verwenden und die Kosten für eine kommerzielle Zertifizierung scheuen, versuchen Sie es einmal bei CACert (*https://www.cacert.org/*). Diese Zertifizierungsstelle arbeitet gemeinschaftsbasiert. Sie können hier kostenfrei teilhaben und über den Import der CACert-Zertifikate Browser-Warnungen unterbinden.

Erstellen Sie zunächst eine sogenannte Certificate Signing Request (kurz CSR, daher im Folgenden die Dateiendung), also eine formal abgebildete Anfrage zur Unterschrift eines Zertifikates durch eine Zertifizierungsstelle. Das zu unterschreibende Zertifikat wird dabei automatisch erstellt und ist in der resultierenden Datei enthalten. Verwenden Sie dafür die folgende Anweisung:



RSA-Schlüssellänge: Sie sollten einen RSA-Schlüssel mit einer Länge von mindestens 2048 Bit, besser 3072 Bit, erstellen. Kürzere Schlüssel gelten nicht mehr als sicher.

```
root@moni:~# openssl req -newkey rsa:2048 > /etc/ssl/private/server.csr
Generating a 2048 bit RSA private key
.............++++++
..................++++++
writing new private key to 'privkey.pem'
Enter PEM pass phrase:SECRETPASSPHRASE
Verifying - Enter PEM pass phrase:SECRETPASSPHRASE
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Berlin
Locality Name (eg, city) []:Berlin
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Nagios/Icinga Kochbuch
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:myserv.mynet.de
Email Address []:webmaster@myserv.mynet.de

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:totalgeheim
An optional company name []:
```

FQDN: Wir empfehlen Ihnen gleich den FQDN (Full Qualified Domain Name) Ihres Monotoring-Servers als CN (Common Name) einzusetzen. Damit vermeiden Sie Probleme, sollte der Rechner später eine neue IP-Adresse bekommen. Wenn der FQDN nicht über DNS aufgelöst wird, dann tragen Sie ihn in die lokale */etc/hosts* Datei ein.

Die von uns hier vorgenommenen Eingaben haben wir für Sie hervorgehoben. Funktional entscheidend ist dabei vornehmlich der Common Name, der den FQDN (Fully Qualified Domain Name) der Maschine oder ersatzweise deren IP enthalten muss. Die übrigen Informationen betreffen das Zertifikat angezeigt und sollten natürlich auch möglichst aussagekräftige Werte enthalten.

Dabei wurde des Weiteren ein privater Schlüssel angelegt, der zunächst im lokalen Verzeichnis als Datei *privkey.pem* erstellt wird. Dieser wird dann als CA verwendet, um das Zertifikat weiter unten zu unterschreiben. Verschieben Sie ihn in das Konfigurationsverzeichnis von `ssl`, in unserer Referenzinstallation ist dies */etc/ssh/private*:

```
root@moni:~# mv privkey.pem /etc/ssl/private/
```

Sie müssen den privaten Schlüssel im PEM-Format (Privacy Enhanced Mail) nun noch umwandeln und ohne die bisher verwendete Passphrase speichern, die dabei entsprechend noch einmal abgefragt wird:

```
root@moni:~# openssl rsa -in /etc/ssl/private/privkey.pem -out
/etc/ssl/private/server.key
Enter pass phrase for /etc/ssl/private/privkey.pem:
writing RSA key
```

Jetzt können Sie den CSR unterschreiben und damit das zu verwendende Zertifikat erstellen:

```
root@moni:~# openssl x509 -in /etc/ssl/private/server.csr -out
/etc/ssl/certs/server.crt -req -signkey /etc/ssl/private/server.key -days 370
Signature ok
subject=/C=DE/ST=Berlin/L=Berlin/O=Nagios/Icinga
Kochbuch/CN=myserv.mynet.de/emailAddress=webmaster@myserv.mynet.de
Getting Private key
```

Das resultierende Zertifikat liegt damit als Datei */etc/ssl/certs/server.crt* vor und kann nun verwendet werden. Da bei einem selbstsignierten Zertifikat keine automatische Überprüfung der Richtigkeit durchgeführt werden kann, lassen Sie sich für einen manuellen Abgleich nun den Fingerprint des Zertifikates ausgeben:

```
root@moni:~# openssl x509 -in /etc/ssl/certs/server.crt -fingerprint -sha1 -noout
SHA1 Fingerprint=29:EE:EC:27:D8:1E:10:F8:0E:89:13:A7:32:8D:B8:DB:42:16:A2:4B
```

Sie werden diesen später benötigen, wenn Sie zum ersten Mal die verschlüsselten Seiten aufrufen.

Einbindung des Zertifikates

Zunächst müssen Sie dem Webserver den erstellten Schlüssel mitteilen. Über die Standardkonfiguration ist bereits eine Datei vorbereitet, die sich in unserem Referenzsystem unter */etc/apache2/sites-available/default-ssl* befindet. Passen Sie hier die Anweisungen für Zertifikat und Schlüssel an die erstellten Dateien an:

```
[...]
# SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
# SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
 SSLCertificateFile /etc/ssl/certs/server.crt
 SSLCertificateKeyFile /etc/ssl/private/server.key
[...]
```

Normalerweise ist diese Konfiguration nicht aktiviert. Stellen Sie sicher, dass der Webserver sie auch verarbeitet:

```
root@moni:/etc/apache2# a2ensite default-ssl
Enabling site default-ssl.
[...]
```

Beschränkung auf SSL-Verbindungen: Sie können die bisher erlaubten, unverschlüsselten Verbindungen vollständig unterbinden. Hierzu können Sie entweder die komplette Definition für unverschlüsselte Verbindungen deaktivieren oder gezielt für einzelne Bereiche vorgeben, dass diese nur per SSL erreichbar sein sollen. Um alle unverschlüsselten Verbindungen (betrifft gegebenenfalls nicht nur Nagios/Icinga) zu unterbinden, deaktivieren Sie das Profil `default`:

```
root@moni:/etc/apache2# a2dissite default
Site default disabled.
[...]
```

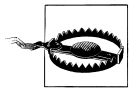
Um hingegen nur für Nagios/Icinga SSL als obligatirisch festzulegen, entfernen Sie das Kommentarzeichen vor der Anweisung `SSLRequireSSL` in der entsprechenden Apache-Konfigurationsdatei (bei uns */etc/apache2/conf.d/icinga.conf*):

```
[...]
<Directory "/usr/local/icinga/sbin">
SSLRequireSSL
[...]
<Directory "/usr/local/icinga/share/">
SSLRequireSSL
[...]
```

Nach diesen Änderungen müssen Sie gegebenenfalls den Webserver neu starten, wie im Folgenden beschrieben wird.

Um SSL generell zu aktivieren, müssen Sie nun noch das entsprechende Modul des Apache2 aktivieren und den Webserver neu starten, damit die entsprechenden Änderungen wirksam werden:

```
root@moni:~# a2enmod ssl
Enabling module ssl.
[...]
root@moni:~# service apache2 restart
[...]
```



APACHE_SERVER_FLAGS: Unter openSUSE müssen Sie, um SSL zu aktivieren, zusätzlich in der Apache2-Konfigurationsdatei */etc/sysconfig/apache2* das SSL-Flag für den Apache-Server wie folgt setzen:

```
[...]
APACHE_SERVER_FLAGS="ICINGA ICINGAWEB KOHANA2 PNP4NAGIOS SSL"
[...]
```

Diese Anpassung können Sie am einfachsten automatisiert mit dem Kommando `a2enflag SSL` vornehmen lassen.

Wenn Sie jetzt den URL *https://localhost/icinga* aufrufen, werden Sie zunächst wegen des unbekannten Zertifikats gewarnt. Lassen Sie sich bei einem Zugriff aus dem Internet unbedingt die Details des Zertifikates anzeigen und vergleichen Sie den dort angegebenen SHA1 (Secure Hash Algorithm)-Fingerprint mit dem vorher für das Zertifikat ausgelesenen. Abbildung 2-1 zeigt, wie der entsprechende Dialog im Firefox-Browser aussieht.

Diskussion

Mit den beschriebenen Schritten können Sie jetzt SSL-verschlüsselt auf Ihr Monitoring-System zugreifen. Das betrifft auch das möglicherweise installierte Icinga-Web beziehungsweise die Addons PNP4Nagios und NagVis. Wenn Sie auch bei den Addons den Zugriff auf SSL beschränken möchten, müssen Sie hier jeweils noch die Direktive `SSLRequireSSL` einfügen.

Abbildung 2-1: Ansicht des Zertifikates im Browser

Um zum Beispiel Icinga-Web nur über SSL zuzulassen, modifizieren Sie in der entsprechenden Konfigurationsdatei Ihres Webservers (bei uns */etc/apache2/conf.d/icinga-web.conf*) die beiden Directory-Direktiven wie folgt:

```
[...]
<Directory /usr/local/icinga-web/lib/ext3>
SSLRequireSSL
[...]
<Directory /usr/local/icinga-web/pub>
SSLRequireSSL
[...]
```

Wenn Sie das Addon NagVis verwenden und auf SSL beschränken möchten, müssen Sie in der entsprechenden Konfigurationsdatei (bei uns */etc/apache2/conf.d/nagvis.conf*) nur die entsprechende Directory-Anweisung wie folgt modifizieren:

```
[...]
<Directory "/usr/local/nagvis/share">
SSLRequireSSL
[...]
```

Um PNP4Nagios gegebenenfalls auf SSL einzuschränken, müssen Sie ebenfalls nur eine Directory-Anweisung anpassen.

```
[...]
<Directory "/usr/local/pnp4nagios/share">
SSLRequireSSL
[...]
```

Auf diese Weise haben Sie Nagios/Icinga und die dazugehörigen Komponenten bei Zugriffen von außen gegen ein direktes Abhören abgesichert. Beachten Sie allerdings, dass Sie wie beschrieben den Fingerprint des Zertifikates abgleichen müssen, da Sie anderenfalls anfällig für sogenannte Man-In-The-Middle (MITM)-Angriffe sind.

Man-In-The-Middle: Mit MITM wird eine Vielzahl von Netzwerkangriffen umschrieben, bei denen der Angreifer versucht, die Kommunikation zwischen zwei Systemen über sich umzuleiten. Im Falle von SSL baut der Angreifer zwei unabhängige SSL-verschlüsselte Verbindungen zu den abzuhörenden Systemen auf und leitet den Datenverkehr zwischen beiden Systemen weiter. Die Kommunikation erfolgt dann zwar verschlüsselt über das Netzwerk, liegt allerdings beim Angreifer im Klartext vor. Um dies zu verhindern müsste sich mindestens ein System dem anderen gegenüber eindeutig authentifizieren. Dies kann über eine zentrale, vertrauenswürdige Zertifizierungsstelle geschehen oder eben über eine manuelle Prüfung des Zertifikates, wie in unserem Fall über den Fingerprint desselbigen.

Siehe auch

- Rezept zur Planung der Netzwerkanbindung des Monitoring-Servers: Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*
- Gemeinschaftsbasierende Zertifizierungsstelle CA-Cert: *https://www.cacert.org/*
- Hilfeseiten des `openssl`-Kommandos: `man openssl`

2.2 Plugins zur Ausführung mit administrativen Rechten vorbereiten

Problem

Einige Plugins müssen, um zu funktionieren, bei der Ausführung mit erweiterten Rechten ausgestattet sein. Daher möchten Sie vielleicht ein oder mehrer Plugins mit root-Rechten versehen.

Lösung

Unter den Standard-Plugins befinden sich bereits mehrere Plugins, die bei einem Aufruf als nicht-priviligierter Benutzer nicht funktionieren. Dies sind `check_icmp` (siehe Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)*) und `check_dhcp` (siehe

Rezept 6.6, *DHCP überwachen (check_dhcp)*). Beide Plugins führen Operationen auf der Netzwerkschnittstelle aus, die bei den Voreinstellungen dem administrativen Benutzer `root` vorbehalten sind. Wir zeigen Ihnen im Folgenden die unterschiedlichen Mechanismen zur Rechte-Eskalation `sudo` und `setuid` und diskutieren ihre Vor- und Nachteile.

Testen von Plugins: Eine sehr gute Anleitung zum Testen von Plugins finden Sie im Icinga Wiki unter *https://wiki.icinga.org/display/testing/Icinga+Plugin+Testing*.

Setzen und Nutzen des setuid-Bits

Bei `setuid` setzen Sie ein bestimmtes Bit in der Rechte-Maske einer Datei, so dass anschließend jede Ausführung als Programm mit den Rechten des Inhabers durchgeführt wird. Dieser Mechanismus läuft also vollständig über das Dateisystem und die dort hinterlegten Einstellungen zu Eigentümer und Ausführungsrechten.

Dazu muss die Datei zunächst dem Benutzer (in unserem Fall `root`) gehören, mit dessen Rechten sie ausgestattet werden soll. Den Besitzer und das setuid-Bit können Sie mit den folgenden Anweisungen festlegen:

```
root@moni:~# chown root /usr/local/icinga/libexec/check_icmp
root@moni:~# chmod u+s /usr/local/icinga/libexec/check_icmp
```

Update der Plugins: Bei der Neuinstallation oder Aktualisierung der Nagiosplugins wird das hier gesetzte setuid-Bit wieder überschrieben. Unter Debian/Ubuntu können Sie dies vermeiden, indem Sie die Rechte durch das Kommando `dpkg-stateoverride` setzen:

```
root@moni:~# dpkg-statoverride --update --add root nagios 4750
/usr/local/icinga/libexec/check_icmp
```

Dieses Problem besteht nicht bei der im folgenden Abschnitt beschriebenen Lösung mit `sudo`.

Jeder zur Ausführung berechtigte Benutzer kann dieses Programm weiterhin ausführen, aber es wird dann eben mit den Rechten des Besitzers laufen. Wie Sie sehen, ist dies eine sehr einfache Konfiguration, die auch keine Anpassungen an den Konfigurationsdateien bedingt.

Konfiguration und Aufruf über sudo

Bei `sudo` rufen Sie ein Programm hingegen explizit über das Hilfsprogramm `sudo` auf, dass ihm die gewünschten erweiterten Rechte einräumt. Dazu müssen Sie gegebenenfalls zunächst die Anwendung `sudo` selbst installieren. In unserem Debian-Referenzsystem können Sie hierzu, sofern die Debian-Repositories korrekt eingebunden sind, das Kommando `aptitude update && aptitude install sudo` nutzen.

Mit dem Kommando `visudo` lässt sich dann jeweils die Datei */etc/sudoers* bearbeiten, in der Sie Anwendungen für die Ausführung mit `sudo` konfigurieren können. Fügen Sie in

dieser Datei folgende Zeile hinzu, wobei Sie den Nutzer `icinga` beziehungsweise `nagios` anpassen müssen, je nachdem, unter welchem die Plugins laufen:

```
Cmnd_Alias                                  ROOTPLUGIN = /usr/local/icinga/libexec/ \
check_dhcp, /usr/local/icinga/libexec/check_icmp
icinga                                      ALL=(root)NOPASSWD:ROOTPLUGIN
```

Parameter einschränken: Sie können das Kommando zusätzlich absichern, indem Sie mit `sudo` die aufzurufenden Parameter einschränken. Im Fall von `check_dhcp` könnten Sie beispielsweise den Aufruf wie folgt auf nur einen spezifizierten Test einschränken:

```
Cmnd_Alias ROOTPLUGIN = /usr/local/icinga/libexec/check_dhcp -i eth0 -m
00:16:36:fb:7a:f6
icinga ALL=(root)NOPASSWD:ROOTPLUGIN
```

Der Eintrag ermöglicht ausschliesslich dem Nutzer `icinga`, die Plugins `check_dhcp` und `check_icmp` mit root-Rechten und ohne Eingabe eines Passwortes aufzurufen. Dazu müssen Sie dem Befehlsaufruf dann allerdings ein *sudo* voranstellen, also etwa:

```
define command {
       command_name                         check_icmp
       command_line                         sudo $USER1$/check_icmp -H $HOSTADDRESS$ \
                                            $ARG1$
}
```

Diskussion

Die nötigen Rechte über den sudo-Mechanismus zu vergeben, mag Ihnen zunächst aufwendig erscheinen. Er stellt dafür aber eine zentrale Stelle zur Verwaltung der entsprechenden Rechte bereit, an der Sie einsehen können, welchen Nutzern eine Ausführung erlaubt ist. Bei der setuid-Variante müssten Sie hierzu vergleichsweise umständlich die Ausführungsrechte der Datei für einzelne Nutzer aufschlüsseln. Des Weiteren werden sudo-Aufrufe jeweils in der Datei */var/log/auth.log* protokolliert, wohingegen Sie bei der setuid-Variante keine Logs erhalten. Wenn es Ihnen vorrangig um eine niedrige Systemlast geht, sollten Sie deshalb das setuid-Bit nutzen, während Sie bei Sicherheitsansprüchen gegebenenfalls die sudo-Variante bevorzugen sollten.

Siehe auch

- Rezept zur Überwachung der Erreichbarkeit mit dem `ping`-Kommando: Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)*
- Rezept zur Überwachung eines DHCP-Dienstes: Rezept 6.6, *DHCP überwachen (check_dhcp)*
- Anleitung zum Testen von Plugins: *https://wiki.icinga.org/display/testing/Icinga+Plugin+Testing*

2.3 SNMP auf Linux-Maschinen vorbereiten

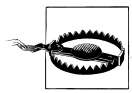
Problem

Sie möchten einen SNMP-Server unter Linux installieren und mit Icinga/Nagios nutzen.

Lösung

SNMP beschreibt generisch ein Verfahren, mit dem Daten für den Abruf vorgehalten oder anlassabhängig versendet werden. Bei letzterem spricht man dann von sogenannten `traps`. Für jedes Datum ist dabei ein Zahlencode für den gezielten Zugriff vorgesehen, der die Position des Datums in einem Baum beschreibt. Prinzipiell wird hier dann zwischen den SNMP-Versionen 1, 2c und 3 unterschieden. Die relevanten Unterschiede sind folgende:

- 2c führt gegenüber 1 größere Zahlenräume ein, die für den Abruf von Zählern moderner Geräte unbedingt erforderlich sind.
- 3 wartet gegenüber 2c mit einer erweiterten Benutzerverwaltung und der Möglichkeit der Verschlüsselung der Authentifizierung (dabei stehen als Protokolle wahlweise MD5 und SHA1 zur Verfügung) und optional auch der Datenübertragung auf (dabei stehen als Protokolle DES und AES zur Verfügung). Für die Authentifizierung stehen Ihnen wahlweise die Algorithmen SHA1 (Secure Hash Algorithm) und MD5 (Message Digest Algorithm 5) zur Verfügung, sowie optional für die Datenübertragung auch DES (Data Encryption Standard) und AES (Advanced Encryption Standard).



SNMP-Version: Entsprechend sollten Sie bei jedem Gerät die jeweils größte unterstützte Version verwenden. Sofern Sie Version 1 oder 2c verwenden müssen, weil beispielsweise Ihre Geräte Version 3 nicht unterstützten, sollten Sie unbedingt auf eine anderweitige Sicherung der Verbindung achten. Ein potentieller Angreifer, dem es gelingt, den Datenverkehr mitzulesen, erhält sonst auch Zugriff auf Ihre SNMP-Passwörter.

Die Installation des SNMP-Servers

Um über SNMP auf Daten zugreifen zu können, benötigen Sie einen laufenden und eingerichteten SNMP-Dienst (typischerweise `snmpd`) sowie einen entsprechenden Client für den Zugriff. Für den Versand von traps benötigen Sie gegebenenfalls zusätzlich eine entsprechende Serverkomponente (typischerweise als `snmptrapd` beim `snmpd` mitgeliefert). Der SNMP-Server aus dem Projekt Net-SNMP ist in jeder der hier erläuterten Distributionen als Paket enthalten. Sie können ihn (hier wie immer bezogen auf unsere Referenzplattform Debian) wie folgt installieren:

```
root@moni:~# aptitude install snmpd
```

Dabei wird der Server mit einer Standardkonfiguration installiert, für den Autostart eingerichtet und gestartet.

Die Konfiguration des SNMP-Servers

Die Konfigurationsdatei */etc/snmp/snmpd.conf* enthält alle für den SNMP-Server wesentlichen Konfigurationsoptionen. Die Standardkonfiguration beschränkt den Server zunächst auf das lokale Interface und richtet einen nicht verschlüsselten Zugang ein.

SNMPD Startparameter: Neben der Hauptkonfigurationsdatei für den SNMP-Daemon, die sich meist unter `/etc/snmp/snmpd.conf` finden lässt, gibt es noch eine weitere Konfigurationsdatei, in der Sie die SNMPD-Startparameter anpassen können. Diese finden Sie distributionsspezifisch an folgenden Orten:

	Hauptkonfigurationsdatei	Startparameter
Debian|Ubuntu	/etc/snmp/snmpd.conf	/etc/default/snmpd
SLES|openSUSE	/etc/snmp/snmpd.conf	/etc/sysconfig/net-snmp
RHEL|CentOS|Fedora	/etc/snmp/snmpd.conf	/etc/sysconfig/snmpd

Damit Sie sich nicht gleich mit allen der in der Standard-Datei definierten Konfigurationsoptionen beschäftigen müssen, schlagen wir vor, dass Sie sich eine Kopie von der Originaldatei machen und dann zunächst eine neue Datei */etc/snmp/snmpd.conf* erstellen, die nur die nachfolgend angeführten Zeilen enthält. Passphrasen und Ort müssen Sie dabei entsprechend anpassen:

```
agentAddress udp:161,udp6:161

createUser icinga SHA "SNMP-PASSPHRASE" AES
rouser icinga

sysLocation Berlin
sysContact root@localhost
```


Die erste Anweisung definiert, auf welchen Schnittstellen der Server agiert. In diesem Fall horcht er auf allen Schnittstellen auf dem Standard-SNMP-Port 161/UDP. Wenn Sie nicht IPv6 verwenden, können Sie das Komma und den zweiten Teil weglassen.

Begrenzung: Sie können hier den Zugriff auf bestimmte Netzwerk-Schnittstellen begrenzen. Geben Sie dazu in der Anweisung die IP-Adresse der Schnittstelle an, wie hier zum Beispiel 192.168.10.12:

```
agentAddress udp:192.168.1.1:161
```

Die zweite Anweisung definiert einen SNMP-Benutzer `icinga` mit der Passphrase `SNMP-PASSPHRASE` für das Authentifikations-Protokoll SHA1. Standardwert wäre hier das ältere und nicht mehr als sicher geltende MD5. Außerdem wird das Protokoll AES für die optionale Verbindungsverschlüsselung unter Nutzung der gleichen Passphrase spezifiziert (sie können dementsprechend alternativ eine getrennte Passphrase angeben).

Benutzerverwaltung: Beachten Sie bitte, dass Änderungen an diesen Benutzerdaten nicht ganz trivial sind. Unserer Erfahrung nach müssen Sie manuell die mit `usmUser` beginnenden Zeilen in der Datei */var/lib/snmp/snmpd.conf* entfernen, bevor geänderte Daten wirksam übernommen werden. Das Hinzufügen neuer Benutzer funktioniert allerdings ohne Weiteres.

Mit der dritten Anweisung erhält der Nutzer lesenden Zugriff auf alle vorhandenen Daten. Die letzten beiden Anweisungen im hier angeführten Beispiel geben Ort und Kontaktperson der Installation an und sind bezüglich der Funktionalität optional.

Wenn Sie anstelle oder zusätzlich zum verschlüsselten Zugriff über Version 3 von SNMP einen Zugriff über Version 1 oder 2 erlauben möchten, verwenden Sie eine Zeile wie die folgende:

```
rocommunity PASSWORD localhost
```

Damit definieren Sie einen nur-lesenden Zugriff mit dem Passwort `PASSWORD` für die lokale Maschine. Für den Zugriff von einer anderen Maschine aus müssten Sie `localhost` durch die IP-Adresse der Maschine ersetzen. Aus Sicherheitsgründen empfehlen wir Ihnen allerdings, Zugriffe wenn immer möglich nur über Version 3 des Protokolls zuzulassen.

Es gibt weitere Konfigurationsoptionen, mit denen Sie den SNMP-Server an Ihre Bedürfnisse anpassen können. Eine davon, die Überwachung von Prozessen, stellen wir Ihnen im Rezept zur Überwachung von Hypervisoren (siehe Rezept 8.3, *Überwachung von Hypervisoren und virtuellen Maschinen*) noch näher vor. Alle Möglichkeiten können Sie ansonsten auch auf den Seiten des SNMP-Projektes nachlesen (siehe *http://www.net-snmp.org/*).

Neustart und Test des SNMP-Servers

Starten Sie nun den SNMP-Server neu, damit die Änderungen wirksam werden:

```
root@moni:~# service snmpd restart
```

Um den Zugang zu testen, benötigen Sie auf der abfragenden Maschine die entsprechenden Dienst-Programme, die im Paket `snmp` enthalten sind. Sofern Sie sich auf der Monitoring-Maschine befinden, werden diese vermutlich bereits installiert sein. Ansonsten können Sie dies manuell nachholen:

```
root@moni:~# aptitude install snmp
```

Paketverwaltung: Denken Sie vor dem Installieren neuer Pakete daran, die Paketlisten des Paketmanagers zu aktualisieren, sofern dies nicht automatisiert vor der Installation geschiet Unter Debian nutzen Sie dazu folgende Anweisung:

```
root@moni:~# aptitude update
```

Testen Sie jetzt die Funktion des Servers, indem Sie sich die ersten zehn von allen verfügbaren Werten anzeigen lassen:

```
root@moni:~# snmpwalk -u icinga -l authPriv -a SHA -A "SNMP-PASSPHRASE" -x AES -X
"SNMP-PASSPHRASE" localhost .1 | head
iso.3.6.1.2.1.1.1.0 = STRING: "Linux moni 2.6.32-5-amd64 #1 SMP Thu Nov 3 03:41:26 UTC
2011 x86_64"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (17372550) 2 days, 0:15:25.50
iso.3.6.1.2.1.1.4.0 = STRING: "root@localhost"
iso.3.6.1.2.1.1.5.0 = STRING: "moni"
iso.3.6.1.2.1.1.6.0 = STRING: "Berlin"
iso.3.6.1.2.1.1.7.0 = INTEGER: 72
iso.3.6.1.2.1.1.8.0 = Timeticks: (0) 0:00:00.00
iso.3.6.1.2.1.1.9.1.2.1 = OID: iso.3.6.1.6.3.10.3.1.1
iso.3.6.1.2.1.1.9.1.2.2 = OID: iso.3.6.1.6.3.11.3.1.1
```

Beachten Sie, dass die vollständige Liste der Ihnen hier zurückgegebenen Werte ist in der Regel mehrere Tausend Zeilen lang ist. Mit dem Befehl `snmpget` können Sie gezielt auf einzelne Werte zugreifen, zum Beispiel nur die Zeile mit der von Ihnen vorher eingepflegten Kontaktadresse:

```
root@moni:~# snmpget -u icinga -l authPriv -a SHA -A "SNMP-PASSPHRASE"
-x AES -X "SNMP-PASSPHRASE" localhost SNMPv2-MIB::sysContact.0
SNMPv2-MIB::sysContact.0 = STRING: root@localhost
```

Diskussion

Soweit scheint SNMP dem `simple` im Namen auch gerecht zu werden. Tatsächlich handelt es sich dabei aber um ein sehr mächtiges Werkzeug. Es erlaubt Ihnen etwa auch diese Datensammlung zu erweitern, so dass Sie nicht nur die bereits vorhandenen Werte verwenden, sondern quasi nach belieben Werte hinzufügen können. Im Rezept 2.4, *SNMP mit MIBs verwenden* stellen wir Ihnen die notwendigen Schritte zur Einbindung externer MIBs vor, die solche weiteren Objekte beschreiben. In Rezept 2.5, *Plugins unter Linux über SNMP ausführen* zeigen wir Ihnen weiter, wie Sie Ihren SNMP-Server um beliebige externe Kommandos erweitern können.

Über SNMP zur Verfügung gestellte OIDs (Object IDentifier) werden dann auch häufig von Plugins genutzt, um entsprechende Werte zu ermitteln und auszuwerten. Bei den Standard-Plugins, die wir Ihnen in diesem Buch beschreiben, sind dies etwa `check_ifstatus` (siehe Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)*) und `check_ifoperstatus` (siehe Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*). Die Einbindung einer Abfrage bezogen auf einen beliebigen OID können Sie bequem mit dem generischen Standard-Plugin `check_snmp` realisieren (siehe Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*).

Häufig werden Sie sich jedoch die zusammenfassende Auswertung mehrerer OIDs wünschen. Externe Plugins tun häufig genau das, wie beispielsweise die SNMP-Plugins von Manubulon, deren Einbindung und Nutzung wir Ihnen in Rezept 7.5, *Einbinden externer*

Plugins am Beispiel von Manubulons SNMP-Plugins zeigen. Wie Sie selbst nach Wunsch mehrere Plugin-Aufrufe aggregieren können, werden wir in Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*) erläutern.

Siehe auch

- Webseiten des net-snmp-Projektes: *http://www.net-snmp.org/*
- Anwenungsbeispiele für net-snmp: *http://www.net-snmp.org/docs/man/snmpd.examples.html*
- Hilfeseiten des `snmpwalk`-Kommandos: `man snmpwalk`
- Hilfeseiten des `snmpcmd`-Kommandos: `man snmpcmd`
- Weitere Rezepte zur Einrichtung des SNMP-Dienstes unter Linux: Rezept 2.4, *SNMP mit MIBs verwenden* und Rezept 2.5, *Plugins unter Linux über SNMP ausführen*
- Rezepte zu Standard-Plugins die SNMP nutzen: Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)*, Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)* und Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*
- Rezepte zu externen und eigenen Plugins, die SNMP nutzen: Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*

2.4 SNMP mit MIBs verwenden

Problem

Sie arbeiten mit SNMP (Simple Network Management Protocol), aber die nummerisch dargestellten OIDs (Object IDentifier) bereiten Ihnen Schwierigkeiten.

Lösung

Die Zahlenkolonnen, die SNMP als Identifikatoren verwendet, sind für Menschen eher schwer zu handhaben. Die bisher aufgeführten sind dabei noch vergleichsweise kurz, daher hier zum Vergleich einmal ein eher langes Beispiel, das sich ebenfalls in der Standard-Konfiguration findet:

```
iso.3.6.1.6.3.16.1.5.2.1.6.10.115.121.115.116.101.109.111.110.108.121.8.1.3.6.1.2.1.25.1
= INTEGER: 1
```

Aus diesem Grund gibt es für SNMP auch eine Namensauflösung, bei der die OIDs in beschreibende Zeichenketten umgewandelt werden. Diese Zeichenketten sind in den sogenannten MIB-Dateien (Management Information Base) enthalten. Der Ablageort für diese MIB-Dateien ist bei unserer Referenzinstallation (siehe Rezept 1.4, *Icinga aus den Quellen installieren*) */usr/share/mibs*, und nach der Installation des `snmpd` enthält diese auch bereits die Dateien für die unterstützte Funktionalität. Wenn Sie SNMP unter Windows nach Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten* verwenden, sind die MIBs für die mitinstallierten Komponenten bereits vorhanden und die folgenden Schritte für diese Maschine zunächst nicht nötig erforderlich.

Aktivierung der Namensauflösung

Damit `snmp` das Verzeichnis aber überhaupt einliest, müssen Sie zunächst die folgende Anweisung in der Konfigurationsdatei */etc/snmp/snmp.conf* auskommentieren:

```
[...]
#mibs :
```

Deaktivierung der Namenauflösung: Wenn Sie die Namensauflösung zu einem spätereren Zeitpunkt deaktivieren wollen, entfernen Sie einfach das gesetzte Kommentarzeichen wieder.

Installation der Standard-MIBs (Debian)

Unter Debian 6 sind die vorhandenen MIB-Dateien aus Lizenzgründen zunächst unvollständig und verweisen auf nicht vorhandene Dateien. Um die fehlenden Dateien zu installieren, müssen Sie hier deshalb die `non-free` genannten Teile der Installationsquellen freischalten. Fügen Sie den Zeilen in der Datei */etc/apt/sources.list* deshalb diese Endung wie im folgenden Beispiel hinzu:

```
deb http://ftp.de.debian.org/debian/ wheezy main non-free
deb-src http://ftp.de.debian.org/debian/ wheezy main non-free

deb http://security.debian.org/ wheezy/updates main non-free
deb-src http://security.debian.org/ wheezy/updates main non-free

deb http://ftp.de.debian.org/debian/ wheezy-updates main non-free
deb-src http://ftp.de.debian.org/debian/ wheezy-updates main non-free
```

Jetzt können Sie das Paket `snmp-mibs-downloader` installieren, das bereits während des Installationslaufs die erforderlichen MIB-Dateien herunterlädt und installiert:

```
root@moni:~# aptitude update
root@moni:~# aptitude install snmp-mibs-downloader
```

Mit folgendem Kommando können Sie die MIBs auf den aktuellen Stand bringen:

```
root@moni:~# download-mibs
```

Test der Namensauflösung

Wenn Sie nun etwa die Anweisung aus dem Test in Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* wiederholen, werden Sie feststellen, dass aus den Zahlenkolonnen automatisch textuelle Beschreibungen geworden sind:

```
root@moni:~# snmpwalk -u icinga -l authNoPriv -a SHA -A "SNMP-PASSPHRASE"
localhost .1 | head
SNMPv2-MIB::sysDescr.0 = STRING: Linux moni 2.6.32-5-amd64 #1 SMP Thu Nov 3 03:41:26 UTC
2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (17995188) 2 days, 1:59:11.88
SNMPv2-MIB::sysContact.0 = STRING: admin@example.org
SNMPv2-MIB::sysName.0 = STRING: moni
SNMPv2-MIB::sysLocation.0 = STRING: Berlin
SNMPv2-MIB::sysServices.0 = INTEGER: 72
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORID.1 = OID: SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance
SNMPv2-MIB::sysORID.2 = OID: SNMP-MPD-MIB::snmpMPDCompliance
```

OIDs übersetzen

Mit dem Befehl `snmptranslate` können Sie nun Übersetzungen der OIDs zwischen den Varianten numerisch und textuell durchführen lassen. Bei den Übersetzungen verwenden Sie die Ausgabeoption `-On` für nummerisch beziehungsweise `-Of` für textuell:

```
root@moni:~# snmptranslate .1.3.6.1.2.1.1.4.0 -Of
.iso.org.dod.internet.mgmt.mib-2.system.sysContact.0

root@moni:~# snmptranslate SNMPv2-MIB::sysContact.0 -On
.1.3.6.1.2.1.1.4.0
```

Wie Sie hier am Beispiel von `snmptranslate` sehen, verstehen die SNMP-Kommandos für OIDs auch textuelle Repräsentationen. Das bedeutet, dass Sie auch bei `snmpwalk` und `snmpget` über diese spezifizieren können, was angezeigt werden soll.

SNMP-Performance: SNMP arbeitet intern immer mit der nummerischen Repräsentation und konvertiert diese anhand von (nicht indexierten) Text-Dateien. Dieser Vorgang ist vergleichsweise arbeitsintensiv, so dass Sie bei automatisierten Vorgängen besser die Zahlen verwenden und auf diese Weise die bei der automatischen Verarbeitung eine unnötige Konvertierung umgehen.

Die Kommandos lassen sich durch weitere Optionen modifizieren, die Sie den entsprechenden Hilfen sowie der für alle SNMP-Befehle geltenden generischen Hilfe zu `snmpcmd` entnehmen können.

MIBs nachpflegen

Hersteller nutzen und pflegen häufig eigene MIBs, um auf diese Weise zusätzliche Daten ihrer Geräte zur Verfügung zu stellen. Cisco als bekannter Hersteller von Netzwerk-Gerä-

ten etwa hat für seine Geräte jeweils spezifische MIBs, die Kunden über den Service zugänglich gemacht werden. Wenn Sie bei SNMP-Abfragen trotz installierter Standard-MIBs wieder nur Zahlenkolonnen sehen, müssen Sie sich die für die Geräte passenden MIBs besorgen.

MIB-Abhängigkeiten: Achten Sie beim Einpflegen neuer MIBs auf Abhängigkeiten. Häufig sind neben einer gerätespezifischen Datei zusätzlich eine oder mehrere herstellerspezifische Datei(en) nötig. Anderenfalls führen nicht erfüllte Abhängigkeiten in den MIBs zu lästigen Warnungen bei der Ausführung der SNMP-Kommandos. Diese können unter Umständen auch auf SNMP basierende Befehle in Nagios/Icinga stören.

Die Installation zusätzlicher MIBs ist sehr von Ihrem System abhängig. Unter Linux wird bei unserer Referenzinstallation typischerweise ein Ordner wie */usr/share/snmp* (siehe Rezept 1.4, *Icinga aus den Quellen installieren*) verwendet, um die MIBs als Textdateien aufzunehmen. Diese werden dann gegebenenfalls automatisch vom SNMP-Server eingelesen, sofern Sie die Konfiguration wie oben beschrieben vorgenommen haben. Unter Windows ist uns keine Möglichkeit bekannt, mit der die als Textdateien vorliegenden MIBs auf vergleichbare Weise integriert werden könnten. Auf Lösungen mit entsprechenden DLL-Dateien zur Erweiterung werden wir hier nicht näher eingehen, sondern wir betrachten nur die mitgelieferten MIBs.

Diskussion

Mit den hier gezeigten Befehlen können Sie die Vielfalt der über SNMP bereitgestellten Möglichkeiten bereits weitgehend nutzen. Wenn Sie genauere Informationen zu einem bestimmten SNMP-Datum benötigen, können Sie die Option `-Td` von `snmptranslate` nutzen. Mittels dieser zeigt Ihnen der Befehl die genauen Spezifikationen aus der passenden MIB-Datei an, wie Sie im folgenden Beispiel (hier wieder anhand des typischerweise von Ihnen eingepflegten Datums `sysContact`) sehen können:

```
root@moni:~# snmptranslate SNMPv2-MIB::sysContact.0 -Td
SNMPv2-MIB::sysContact.0
sysContact OBJECT-TYPE
 -- FROM SNMPv2-MIB, RFC1213-MIB
 -- TEXTUAL CONVENTION DisplayString
 SYNTAX OCTET STRING (0..255)
 DISPLAY-HINT "255a"
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION "The textual identification of the contact person for
 this managed node, together with information on how
 to contact this person. If no contact information is
 known, the value is the zero-length string."
::= { iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) system(1) sysContact(4) 0 }
```

Wie Sie Abfragen im Hinblick auf einzelne OIDs in Nagios/Icinga einbinden können, werden wir in Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)* erläutern.

Siehe auch

- Hilfeseiten des `snmpget`-Kommandos: `man snmpget`
- Hilfeseiten des `snmpwalk`-Kommandos: `man snmpwalk`
- Hilfeseiten des `snmptranslate`-Kommandos: `man snmptranslate`
- Hilfeseiten des `snmpcmd`-Kommandos: `man snmpcmd`
- Rezepte zur Installation des SNMP-Dienstes unter Linux und Windows: Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* und Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*
- Rezept zur Erstellung generischer SNMP-Abfragen: Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*

2.5 Plugins unter Linux über SNMP ausführen

Problem

Sie möchten auf einer Linux-Maschine Plugins lokal ausführen und die Ergebnisse über SNMP (Simple Network Management Protocol) bereitstellen.

Lösung

Wir zeigen Ihnen im Folgenden, wie Sie SNMP erweitern und damit beliebige Programme ausführen lassen können. Im Kontext dieses Buches geht es dabei um die Möglichkeit des Ausführens von Plugins über den SNMP-Server, wie wir Ihnen hier am Beispiel `check_apt` (siehe Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*) demonstrieren werden. Sie sollten zu diesem Zweck den Agenten `snmpd` bereits wie in Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* beschrieben installiert haben.

Das entsprechende Plugin muss zu einer Ausführung auf der entsprechenden Maschine verfügbar sein. Installieren Sie deshalb die Plugins auf der jeweiligen Maschine. Wenn die Plugins verfügbar sind, können Sie `check_apt` über die folgende Zeile in der Konfigurationsdatei */etc/snmp/snmpd.conf* einbinden:

```
[...]
extend check_apt /usr/local/icinga/libexec/check_apt
[...]
```

Nach dieser Änderung ist ein Neustart des SNMP-Servers erforderlich, damit die entsprechende Konfigurationsoption diesem auch mitgeteilt wird. Führen Sie den Neustart etwa wie folgt durch (hier am Beispiel von auf Debian basiertem Linux mit dem Befehl `service`):

```
root@moni:~# service snmpd restart
```

Nun können Sie das Plugin über den Abruf der entsprechenden OIDs in der `NET-SNMP-EXTEND-MIB` ausführen und sich die Ergebnisse anzeigen lassen:

```
icinga@moni:~$ snmpwalk -On -v 3 -l authNoPriv -u icinga -a sha -A "PASSPHRASE" 127.0.0.1
NET-SNMP-EXTEND-MIB::nsExtendObjects
.1.3.6.1.4.1.8072.1.3.2.1.0 = INTEGER: 1
.1.3.6.1.4.1.8072.1.3.2.2.1.2.9.99.104.101.99.107.95.97.112.116 = STRING: /usr/local/icin
ga/libexec/check_apt
.1.3.6.1.4.1.8072.1.3.2.2.1.3.9.99.104.101.99.107.95.97.112.116 = STRING:
.1.3.6.1.4.1.8072.1.3.2.2.1.4.9.99.104.101.99.107.95.97.112.116 = STRING:
.1.3.6.1.4.1.8072.1.3.2.2.1.5.9.99.104.101.99.107.95.97.112.116 = INTEGER: 5
.1.3.6.1.4.1.8072.1.3.2.2.1.6.9.99.104.101.99.107.95.97.112.116 = INTEGER: shell(2)
.1.3.6.1.4.1.8072.1.3.2.2.1.7.9.99.104.101.99.107.95.97.112.116 = INTEGER: run-on-read(1)
.1.3.6.1.4.1.8072.1.3.2.2.1.20.9.99.104.101.99.107.95.97.112.116 = INTEGER: permanent(4)
.1.3.6.1.4.1.8072.1.3.2.2.1.21.9.99.104.101.99.107.95.97.112.116 = INTEGER: active(1)
.1.3.6.1.4.1.8072.1.3.2.3.1.1.9.99.104.101.99.107.95.97.112.116 = STRING: APT OK:
0 packages available for upgrade (0 critical updates).
.1.3.6.1.4.1.8072.1.3.2.3.1.2.9.99.104.101.99.107.95.97.112.116 = STRING: APT OK:
0 packages available for upgrade (0 critical updates).
.1.3.6.1.4.1.8072.1.3.2.3.1.3.9.99.104.101.99.107.95.97.112.116 = INTEGER: 1
.1.3.6.1.4.1.8072.1.3.2.3.1.4.9.99.104.101.99.107.95.97.112.116 = INTEGER: 0
.1.3.6.1.4.1.8072.1.3.2.4.1.2.9.99.104.101.99.107.95.97.112.116.1 = STRING: APT OK:
0 packages available for upgrade (0 critical updates).
```

Um ein einzelnes Datum wie etwa den Status-Code in Nagios/Icinga einzubinden, können Sie dann das Plugin `check_snmp` (siehe Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*) nutzen. Alternativ können Sie mit einem Wrapper (siehe Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*) mehrere Werte wie etwa den numerischen und textuellen Status auslesen und zu einem Rückgabewert zusammenfassen.

Diskussion

Benutzerdefinierte Erweiterungen des Agenten `snmpd` sind ein mächtiges Werkzeug, um beliebige Tests auf mit SNMP ausgerüsteten Zielsystemen durchzuführen. Wir empfehlen Ihnen auf allen zu überwachenden Linux- und Solaris-Maschinen neben dem `snmpd` auch die Standard-Plugins zu installieren. Erstellen Sie sich dann eine passende Konfigurationsdatei für den `snmpd`, die die für Sie relevanten Plugins wie oben beschrieben einbindet. Anschließend müssen Sie dann nach der Installation nur noch diese Konfigurationsdatei auf alle Maschinen verteilen, um die entsprechenden Einbindungen in Nagios/Icinga vornehmen zu können. In Rezept 8.1, *Überwachung von Unix-/Linux-Servern* ist ein entsprechender Satz an Plugins aufgeführt.

Siehe auch

- Rezept zur Installation des SNMP-Dienstes unter Linux: Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*
- Rezept zur Überwachung von Aktualisierungen: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*
- Hilfeseiten des `snmpwalk`-Kommandos: `man snmpwalk`
- Rezept zur Erstellung generischer SNMP-Abfragen: Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*
- Rezept mit Beispiel für ein erweitertes Wrapper-Script für die Zusammenfassung von Prüfungen: Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*
- Rezept zur Überwachung von Linux-Maschinen: Rezept 8.1, *Überwachung von Unix-/Linux-Servern*

2.6 SNMP auf Windows-Maschinen vorbereiten

Problem

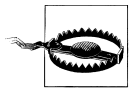
Sie möchten einen SNMP-Server unter Windows installieren und mit Nagios/Icinga nutzen.

Lösung

SNMP beschreibt generisch ein Verfahren, mit dem sich Daten für den Abruf vorhalten oder anlassabhängig versenden lassen. Bei letzterem spricht man dann von sogenannten `traps`. Für jedes Datum ist dabei ein Zahlencode für den gezielten Zugriff vorgesehen, der dabei die Position des Datums in einem Baum beschreibt. Prinzipiell wird dabei zwischen den SNMP Versionen 1, 2c und 3 unterschieden. Die relevanten Unterschiede sind folgende:

- 2c führt gegenüber Version 1 größere Zahlenräume ein, die für den Abruf von Zählern moderner Geräte unbedingt erforderlich sind.
- 3 führt gegenüber 2c eine erweiterte Benutzerverwaltung und die Möglichkeit der Verschlüsselung der Authentifizierung (dabei stehen als Protokolle wahlweise MD5 und SHA1 zur Verfügung) und optional auch der Datenübertragung (dabei stehen als Protokolle DES und AES zur Verfügung) ein.

SNMP Version 3 wird leider unter Windows standardmäßig nicht unterstützt. Daher können Sie hier erstmal nur auf SNMP Version 2c zugreifen. Desweiteren ist SNMP ab Windows Server 2012 auf den Status `depreciated` gesetzt, dass heißt SNMP soll in Zukunft nicht mehr seitens des Betriebssystems unterstützt werden. Lesen Sie dazu bitte auch den Diskussionsteil dieses Rezeptes.



SNMP-Version: Entsprechend sollten Sie bei jedem Gerät die jeweils höchste unterstützte Version verwenden. Sofern Sie Version 1 oder 2c verwenden müssen, weil beispielsweise Ihre Geräte Version 3 nicht unterstützten, sollten Sie auf eine anderweitige Sicherung der Verbindung achten. Ein potentieller Angreifer, dem es gelingt, den Datenverkehr mitzulesen, erhält auch Zugriff auf Ihre SNMP-Passwörter.

Die Installation des SNMP-Servers

In allen aktuellen Windows-Versionen (seit Windows NT beziehungsweise Windows 20XX) ist eine SNMP (Simple Network Management Protocol)-Komponente enthalten, die Sie lediglich nachinstallieren und konfigurieren müssen.

Zur Aktivierung rufen Sie zunächst in der Systemsteuerung die Funktion `Windows-Funktionen aktivieren oder deaktivieren` auf. Die folgende Abbildung 2-2 zeigt die entsprechende Auswahl, hier für ein System vom Typ Windows Server 2008 R2.

Abbildung 2-2: Systemsteuerung: Aktivierung von Windows-Komponenten

Im nun geöffneten Servermanager wählen Sie dann `Features` und anschließend `Features hinzufügen` aus. Abbildung 2-3 zeigt Ihnen die entsprechende Auswahl.

Im nächsten Schritt erhalten Sie eine Liste von aktivierbaren Features (siehe Abbildung 2-4). Wählen Sie erst den Menüpunkt `SNMP-Dienst` und aktivieren Sie dann im sich öffnenden Untermenü wieder `SNMP-Dienst`. Die Funktion `WMI SNMP-Anbieter` benötigen Sie für Nagios/Icinga nicht.

Abbildung 2-3: Windows-Funktionen: Features hinzufügen

Windows Management Instrumentation: Mit WMI stellt Windows eine alternative Schnittstelle zu SNMP zur Verfügung. Über diese lassen sich ebenfalls Systemeigenschaften über das Netzwerk abfragen. Die Tests werden hier von einem zentralen Windows-Server, dem WMI-Proxy, durchgeführt und können auch an dieser Stelle über NRPE von Ihrem Monitoring-Server ausgelöst werden. Alternativ können Sie WMI-Abfragen auch über NSClient++ durchführen. Wie werden hier allerdings nicht weiter auf WMI eingehen, da Microsoft soweit uns bekannt WMI mittelfristig nicht weiter unterstützen wird.

Wählen Sie dann `Weiter` → `Installieren` → `Schliessen`, um die Installation abzuschließen. Danach sollten Sie ein Windows Update durchführen und Ihr System neu starten. Nun ist die Installation der SNMP-Komponente abgeschlossen.

Die Konfiguration des SNMP-Servers

Zur SNMP-Konfiguration rufen Sie den Servermanager nun erneut auf. Wie in Abbildung 2-5 im hinteren Fenster zu sehen, wählen Sie dann in dem Menü `Konfiguration` → `Dienste` den Eintrag für `SNMP-Dienst`. Über das Kontextmenü rufen Sie dann den Menupunkt `Eigenschaften` auf. In dem folgenden Fenster (in Abbildung 2-5 das vordere Fenster auf der rechten Seite) tragen Sie dann mindestens unter dem Menüpunkt `Sicherheit` die Community-Strings für die Authentifizierung ein und fügen die IP-Adresse Ihres Monitoring-Servers als berechtigten Host hinzu. Unter dem Menüpunkt `Agent` sollten Sie dann noch abschließend die Felder für `Kontakt` und `Ort` ausfüllen, sowie die später zu überwachenden Services auswählen.

Die Installation und Konfiguration von SMNP ist unter Windows 7 im Wesentlichen identisch.

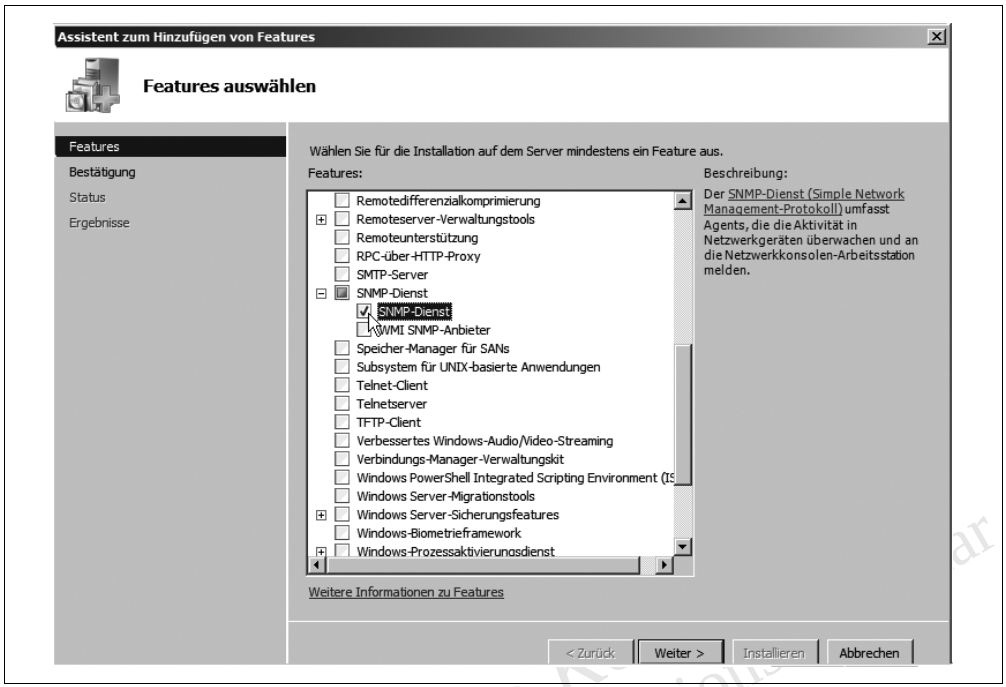


Abbildung 2-4: Windows-Funktionen: SNMP-Dienst auswählen

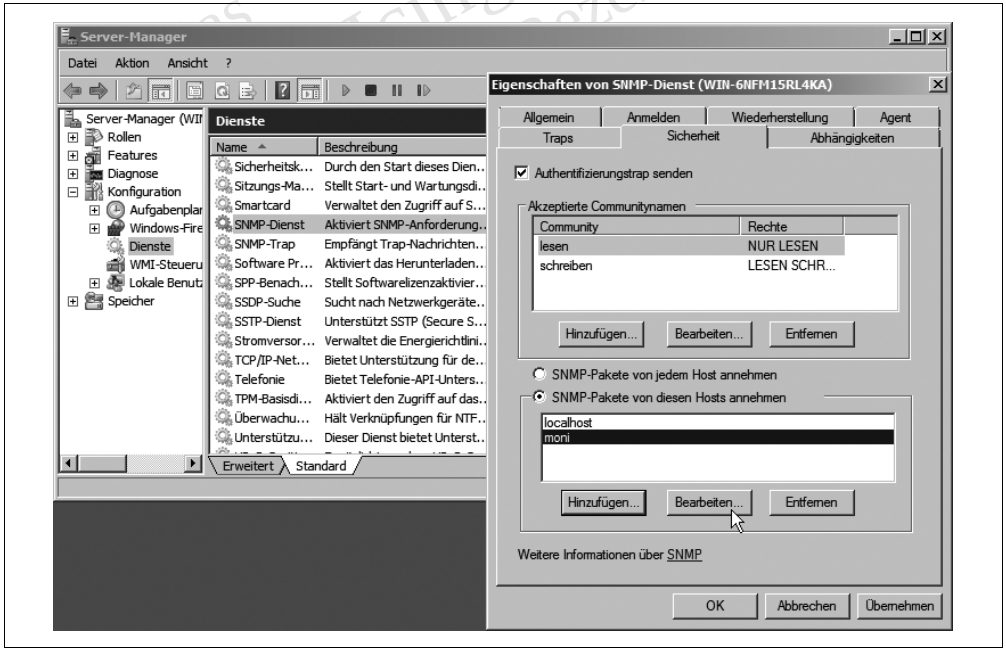


Abbildung 2-5: Windows-Funktionen: SNMP-Dienst konfigurieren

Test des SNMP-Servers

Wenn Sie die SNMP-Funktionalität wie beschrieben installiert haben, können Sie den Zugang von Ihrer Monitoring-Maschine nun wie folgt testen (hier am Beispiel eines Windows 7-Rechners). Beachten Sie dabei, dass Windows SNMP Version 3 standardmäßig nicht unterstützt:

```
icinga@moni:~$ snmpwalk -v2c -c COMMUNITY 10.85.58.111 | head
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: Intel64 Family 6 Model 2 Stepping 3
AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7601 Multiprocessor Free)
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.311.1.1.3.1.1
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (173699717) 20 days, 2:29:57.17
SNMPv2-MIB::sysContact.0 = STRING: root@Tso7
SNMPv2-MIB::sysName.0 = STRING: Tso7
SNMPv2-MIB::sysLocation.0 = STRING:
SNMPv2-MIB::sysServices.0 = INTEGER: 76
IF-MIB::ifNumber.0 = INTEGER: 19
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
```

Die vollständige Liste der Ihnen hier zurückgegebenen Werte ist in der Regel mehrere Tausend Zeilen lang. Mit dem Befehl `snmpget` können Sie gezielt auf einzelne Werte zugreifen, zum Beispiel nur die Zeile mit der von Ihnen vorher eingepflegten Kontaktadresse:

```
icinga@moni:~$ snmpget -v2c -c COMMUNITY 10.85.58.111 SNMPv2-MIB::sysContact.0
SNMPv2-MIB::sysContact.0 = STRING: root@Tso7
```

Wenn die Maschine nicht reagiert, überprüfen Sie gegebenenfalls vorhandene Zugangsbeschränkungen. Eine auf dem Endgerät oder im Netzwerk eventuell vorhandene Firewall könnte die Verbindung blockieren. Sie müssten in diesem Fall eine Kommunikation über Port upd/161 zulassen.

Diskussion

Sie können auf die beschriebene Weise recht einfach einen SNMP-Dienst installieren. Die passenden MIBs (siehe Rezept 2.4, *SNMP mit MIBs verwenden*) sind dabei bereits enthalten. Unter Windows können Sie so allerdings zunächst keine Verschlüsselung nutzen, da SNMP hier nur bis einschliesslich Version 2c unterstützt wird. Weil der `community string` im Klartext übertragen wird, sollten Sie diese Funktion in der hier genannten Art und Weise nur in vertrauenswürdigen Netzwerken verwenden (siehe hierzu Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*) und auf eine Nutzung in öffentlichen Netzen verzichten. Zusätzlich sollten Sie den Zugriff auf den SNMP-Server wie vorher beschrieben auf die erforderlichen Rechner beschränken.

Sie können alternativ erwägen, SNMP über eine externe Software mit Unterstützung für SNMP Version 3 zu realisieren. Dies gilt insbesondere für Windows Server 2012 und folgende Versionen, da die SNMP-Komponente hier, genauso wie WMI, als `depreciated` gesetzt ist. Dies bedeutet, das SNMP in Zukunft seitens des Betriebssystems nicht mehr

unterstützt werden soll. Derzeit lässt sich SNMP allerdings noch aktivieren, indem Sie die SNMP-Tools-Erweiterung nachträglich installieren.

Desweiteren lässt sich der SNMP-Dienst unter Windows nicht so leicht erweitern wie unter Linux. Im Rezept 2.5, *Plugins unter Linux über SNMP ausführen* zeigen wir für Linux, wie Sie Pugins und auch andere beliebige Kommandos über SNMP ausführen lassen können. Für Windows ist uns kein vergleichbarer Weg zur Erweiterung der mitgelieferten SNMP-Komponente bekannt. Wenn Sie auf Windows-Maschinen Plugins ausführen und abfragen möchten, so empfehlen wir Ihnen die Nutzung von NSClient++ (siehe Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*). Die mit dem SNMP-Dienst verfügbaren Informationen bieten Ihnen aber bereits für viele Fragestellungen die nötigen Informationen.

Über SNMP zur Verfügung gestellte OIDs werden entsprechend häufig von Plugins genutzt, um aus den Werten Status für Nagios/Icinga zu generieren. Bei den Standard-Plugins, die wir Ihnen in diesem Buch beschreiben, sind dies etwa `check_ifstatus` (siehe Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)*) und `check_ifoperstatus` (siehe Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*). Die Einbindung einer Abfrage im Hinblick auf eine beliebige OID können Sie bequem mit dem generischen Standard-Plugin `check_snmp` realisieren (siehe Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*).

Häufig werden Sie sich jedoch die zusammenfassende Auswertung mehrerer OIDs wünschen. Externe Plugins tun häufig genau das, wie beispielsweise die SNMP-Plugins von Manubulon, deren Einbindung und Nutzung wir Ihnen in Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins* zeigen. Wie Sie selbst nach Wunsch mehrere Plugin-Aufrufe aggregieren können, werden wir in Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen* erläutern. Die Einbindung von Windows-Rechnern in Nagios/Icinga beschreiben wir zusammenfassend in Rezept 8.2, *Überwachung von Windows-Maschinen*.

Siehe auch

- Rezept zur Verwendung von MIB: Rezept 2.4, *SNMP mit MIBs verwenden*
- Rezept zur Planung der Netzwerkanbindung des Monitoring-Servers: Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*
- Rezept zur Ausführung von Plugins über SNMP: Rezept 2.5, *Plugins unter Linux über SNMP ausführen*
- Rezept zur Installation von NRPE und NSClient unter Linux: Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezepte zu Standard-Plugins, die SNMP nutzen: Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)* und Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*

- Rezept zur Erstellung generischer SNMP-Abfragen: Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*
- Rezepte zu externen und eigenen Plugins, die SNMP nutzen: Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*
- Rezepte zur Überwachung von Windows-Maschinen: Rezept 8.2, *Überwachung von Windows-Maschinen*

2.7 NRPE auf Unix-Maschinen vorbereiten

Problem

Sie möchten auf Ihren Maschinen einen NRPE-Server unter Unix installieren und das System für die Überwachung durch Nagios/Icinga vorbereiten.

Lösung

Um eine Maschine mit NRPE (Nagios Remote Plugin Executor) zu überwachen, müssen Sie auf jedem zu überwachenden Gerät einen NRPE-Dienst installieren. Dieser antwortet dann auf die NRPE-Anfragen Ihres Monitoring-Servers. Wir zeigen Ihnen im Folgenden exemplarisch die Installation und Konfiguration des NRPE-Dienstes auf einer Linux-Maschine.

MacOSX, Solaris, BSD und Co: Der Quellcode von NRPE kompiliert auf zahlreichen Systemen. So können Sie den Agenten nicht nur unter Linux, sondern auch unter MacOSX, Windows (mit Cygwin), Solaris, BSD (NetBSD, FreeBSD, OpenBSD), AIX, HP-UX, IRIX, SCO Unix, OpenVMS und SunOS einsetzen.

Die Installation des NRPE-Dienstes

Wir empfehlen Ihnen die paketbasierte Installation des NRPE-Dienstes. In allen hier betrachteten Distributionen gibt es fertige Pakete für die Installation .

In auf Debian basierenden Distributionen heißt dieses Paket `nagios-nrpe-server` und unter openSUSE ist das Paket als `nagios-nrpe` benannt. Zusätzlich sollten Sie das Paket mit den `nagios-plugins` auf dem zu überwachenden Server installieren. Die Installation können Sie durch das Kommando `aptitude install nagios-nrpe-server nagios-plugins` beziehungsweise bei SLES|openSUSE durch das Kommando `zypper install nagios-nrpe nagios-plugins` automatisch vornehmen lassen. Der NRPE-Dienst wird in beiden Fällen nach der Installation automatisch gestartet.

Unter RHEL|CentOS|Fedora benötigen Sie für NRPE das EPEL- oder das RepoForge-Repostory. Die Pakete heißen hier `nrpe` beziehungsweise `nagios-nrpe`. Der NRPE-Dienst

muss in unserem Fall allerdings manuell gestartet beziehungsweise durch das Kommando `chkconfig nrpe on` für den automatischen Start konfiguriert werden.

Wenn Sie keine sourcebasierte Installation durchführen wollen, können Sie jetzt gleich in den Konfigurationsteil dieses Rezeptes springen. Falls Sie trotzdem eine sourcebasierte Installation durchführen möchten, können Sie den Sourcecode über den Link *http://sourceforge.net/projects/nagios/files/nrpe-2.x/* beziehen. Speichern Sie auf dem zu überwachenden Zielrechner das Paket im Verzeichnis */usr/local/src/* und entpacken es.

NRPE 2.12: Die NRPE-Version 2.12 hat sich über viele Jahre bewährt. Falls Sie Probleme mit aktuelleren Versionen von NRPE haben, dann versuchen Sie es mit dieser .

Anschließend erstellen Sie einen Nutzer `icinga` beziehungsweise `nagios` mit einer möglichst hohen Nutzer-ID und kompilieren den Quellcode wie folgt:

```
root@deenes:~# cd /usr/local/src/
root@deenes:/usr/local/src# tar xzvf nrpe-2.14.tar.gz
root@deenes:/usr/local/src# cd nrpe-2.14
root@deenes:/usr/local/src/nrpe-2.14# useradd -u 5666 -d /usr/local/nagios -c "Nagios user"
nagios
root@deenes:/usr/local/src/nrpe-2.14# ./configure --sysconfdir=/etc/nagios --enable-ssl
root@deenes:/usr/local/src/nrpe-2.14# make all
```

Anschließend müssen Sie noch die Vorlage für die Konfigurationsdatei an die richtige Stelle kopieren und dem Nutzer `nagios` beziehungsweise `icinga` noch die entsprechenden Zugriffsrechte erteilen:

```
root@deenes:/usr/local/src/nrpe-2.14# mkdir -p /etc/nagios && cp sample-config/nrpe.cfg
/etc/nagios/
root@deenes:/usr/local/src/nrpe-2.14# chown nagios:nagios /etc/nagios/nrpe.cfg
```

Nach erfolgter Kompilierung finden Sie `nrpe` im Verzeichnis */usr/local/nagios/bin/*, das Server-Plugin `check_nrpe` (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*) in */usr/local/nagios/libexec/* sowie die Konfigurationsdatei */etc/nagios/nrpe.cfg*. Sofern Sie den NRPE-Dienst manuel starten, müssen Sie Ihrem Monitoring-Server berechtigen, auf den NRPE-Dienst zuzugreifen. Modifizieren Sie in der Konfigurationsdatei als Erstes das Schlüsselwort `allowed_hosts` und tragen hier die IP-Adresse Ihres Monitoring-Servers ein. Bei Bedarf können Sie auch weitere Geräte, durch Kommata separiert, hinzufügen.

```
# ALLOWED HOST ADDRESSES
allowed_hosts=SERVERIP
```

Anschließend können Sie den NRPE-Dienst mit folgendem Kommando manuell starten:

```
root@deenes:~# /usr/local/nagios/bin/nrpe -c /etc/nagios/nrpe.cfg -d
```

Wir empfehlen Ihnen allerdings, den NRPE-Dienst am besten über einen Inetd-Daemon zu starten. Dies bedeutet, dass NRPE bei jeder Anfrage automatisch gestartet und wieder

beendet wird. Fügen Sie dazu als Erstes folgende Zeile in der Datei */etc/services* hinzu, sofern diese nicht bereits vorhanden ist:

```
nrpe 5666/tcp # Nagios Remote Plugin Executor
```

Die folgende Konfiguration ist davon abhängig, ob Sie `inetd` oder `xinetd` einsetzten. Wenn bei Ihnen `inetd` zum Einsatz kommt, fügen Sie nun folgende Zeile in die Konfigurationsdatei */etc/inetd.conf* ein:

```
nrpe stream tcp nowait nagios /usr/sbin/tcpd /usr/local/nagios/bin/nrpe
-c /etc/nagios/nrpe.conf --inetd
```

Im Falle von `xinetd` legen Sie eine Datei */etc/xinetd.d/nrpe* mit nachfolgendem Inhalt an. Eine Vorlage zum Anpassen finden Sie in dem Verzeichnis */usr/local/src/nrpe-2.14/sample-config*. Dabei entspricht die Variable SERVERIP im Parameter `only_from` der IP-Adresse Ihres Monitoring-Servers. Beachten Sie, dass alle in der folgenden inetd-Konfiguration definierten Parameter die entsprechenden Werte in der NRPE-Konfigurationsdatei überschreiben:

```
# /etc/xinetd.d/nrpe
# default: on
# description: NRPE
service nrpe
{
        flags = REUSE
        socket_type = stream
        wait = no
        user = nagios
        server = /usr/local/nagios/bin/nrpe
        server_args = -c /etc/nagios/nrpe.cfg --inetd
        log_on_failure += USERID
        disable = no
        only_from = 127.0.0.1 SERVERIP
}
```


Nach Anpassung der Konfiguration müssen Sie nun *inetd* mit dem Kommando `/etc/init.d/inetd reload` beziehungsweise *xinetd* mit dem Kommando `/etc/init.d/xinetd reload` neu starten. Sie können wie folgt prüfen, ob der NRPE-Dienst nun bereit ist, Anfragen auf dem Port tcp/5666 zu beantworten:

```
root@deenes:~# netstat -lnt | grep ':5666'
tcp 0 0 0.0.0.0:5666 0.0.0.0:* LISTEN
```

Abschließend stellen Sie sicher, dass eine eventuell vorhandene Firewall sowohl `ping` (icmp echo request) als auch die Kommunikation über den NRPE port tcp/5666 vom Monitoring-Server aus zulässt. Die Anpassung der lokalen Firewall ist bei openSUSE in jedem Fall notwendig.

Die Konfiguration des NRPE-Dienstes

Bereits ohne weitere Konfiguration sollten Sie nun von Ihrer Monitoring-Maschine aus auf die ersten Daten zugreifen können. Testen Sie wie folgt den prinzipiellen Zugriff von

der Monitoring-Maschine durch einen Aufruf des Plugins `check_nrpe` (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*):

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nrpe -H CLIENTIP -t 60 -p 5666
NRPE v2.14
```

An der Antwort können Sie erkennen, ob eine erfolgreiche Kommunikation mit dem installierten NRPE-Dienst stattgefunden hat. Wenn Sie hier keine Meldung oder eine Fehlermeldung erhalten, überprüfen Sie die Verbindung. Prüfen Sie gegebenenfalls ebenfalls noch einmal Ihre Firewall-Konfiguration und vergewissern Sie sich, dass die bei der Installation angegebene IP-Adresse des Monitoring-Servers korrekt in der Konfigurationsdatei hinterlegt ist.

`CHECK_NRPE: Error - Could not complete SSL handshake`: Diese Fehlermeldung weist auf eine Inkompatibilität zwischen den verwendeten SSL-Versionen hin oder auf den Umstand, dass der NRPE-Client ohne SSL mit dem NRPE-Server kommuniziert, obwohl dieser es voraussetzt, oder auch umgekehrt. Sie kann aber auch eine andere Ursache haben, zum Beispiel die, dass der NRPE-Daemon seine Konfigurationsdatei nicht findet oder deren Zugriffsrechte unzureichend gesetzt sind.

Im Folgenden werden wir genauer auf die Konfigurationsdatei `/etc/nagios/nrpe.cfg` eingehen. Der NPRE-Dienst verbietet standardmäßig die Übergabe von Parametern über das Netzwerk. Diese Sperre und das Filtern von Sonderzeichen lässt sich mit der nachfolgenden Option außer Kraft setzten. Allerdings raten wir Ihnen aus Sicherheitsgründen dringend dazu, die Sperre beizubehalten. Es ist besser, Sie setzen diese Sektion nur temporär ein, um neue Kommandos zu testen:

```
# COMMAND ARGUMENT PROCESSING
dont_blame_nrpe=1
```

Sie sollten bevorzugt Alias-Kommandos nutzen. In diesen hinterlegen Sie auf dem zu überwachenden System das auszuführende Kommando, alle benötigter Parameter und die zu übergebenden Argumente.

Puppet: Für die Verteilung von der NRPE-Konfiguration eignen sich Automatisierungstools wie beispielsweise Puppet.

Im Folgenden sehen Sie die mitgelieferten Alias-Kommandos, die auf häufig genutzte Standard-Plugins zur Überwachung lokaler Dienste verweisen. Auf diese gehen wir in Rezept 8.1, *Überwachung von Unix-/Linux-Servern* genauer ein. Dort erläutern wir auch, wie Sie weitere Alias-Kommandos selbst definieren können:

```
# COMMAND DEFINITIONS
command[check_users]=/usr/lib/nagios/plugins/check_users -w 5 -c 10
command[check_load]=/usr/lib/nagios/plugins/check_load -w 15,10,5 -c 30,25,20
```

```
command[check_hda1]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% -p /dev/hda1
command[check_zombie_procs]=/usr/lib/nagios/plugins/check_procs -w 5 -c 10 -s Z
command[check_total_procs]=/usr/lib/nagios/plugins/check_procs -w 150 -c 200
```

Diskussion

Mit der Installation von NRPE erlauben Sie aus dem Netzwerk Zugriff auf Ihre Maschinen. Achten Sie deshalb im Besonderen auf eine sichere Nutzung. NRPE bietet Ihnen zwar eine per SSL verschlüsselte Datenübertragung. Allerdings unterstützt das Plugin `check_nrpe` (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*) von den NagiosPlugins nur ein fest einkompiliertes Zertifikat.

Sofern Sie Rechenleistung einsparen müssen (siehe Rezept 1.3, *Hardware-Anforderungen abschätzen*) oder die Checks über ein sicheres Outband-Netzwerk (Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen* betreiben, empfehlen wir Ihnen den Einsatz von check_by_ssh Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*. Achten Sie dann darauf, die Schlüssel ohne Passphrase für ein vordefiniertes Kommando zu konfigurieren, wie in dem entsprechenden Rezept beschrieben.

Bei der Nutzung von NRPE sollten Sie auf jeden Fall die folgenden Punkte beachten:

- Verzichten Sie auf die Schlüsselwörter `dont_blame_nrpe=1` und `--enable-bash-command-substitution`. Dadurch wird Ihnen ermöglicht, dem Plugin beliebige Parameter zu übergeben beziehungsweise beliebige Shell-Kommandos auf dem Zielsystem auszuführen. Wir empfehlen Ihnen, dass Sie Checks über Alias-Kommandos definieren, in denen Sie alle Parameter vorgeben.
- Für Kommandos, die Root-Rechte erfordern, können Sie `sudo` nutzen. Sie sollten dabei auch hier unbedingt die Nutzung von `sudo` auf den gewünschten Check und dessen Parameter beschränken (siehe dazu Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten*).
- Stellen Sie sicher, dass Sie SSL-verschlüsselte Verbindungen verwenden, wie dies bei NRPE die Vorgabe ist. Sie sollten diese maximal zu Testzwecken deaktivieren.
- Auch sollten Sie den Zugriff mit der Option `allowed_hosts` in der Konfigurationsdatei des NRPE-Dienstes auf wenige IP-Adressen beschränken. Bei der Nutzung von `xinetd` wäre die entsprechende Option `only_from` in der xinetd-Konfigurationsdatei. Nutzen Sie dabei nicht etwa den Hostnamen, sondern die IP-Adresse selbst. Andernfalls lässt sich dieser Schutz über sogenanntes DNS-Spoofing aushebeln.

DNS-Spoofing: Bei DNS-Spoofing handelt es sich um einen Angriff auf das Domain Name System, bei dem der Angreifer die Zuordnung von IP-Adresse und Domain-Namen verfälscht. Es gibt zahlreiche, zum Teil sehr unterschiedliche Methoden, mit denen sich ein solcher Angriff realisieren lässt.

Wir erläutern die eigentliche Nutzung der entsprechenden Plugins in jeweils eigenen Rezepten: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*. In Rezept 8.1, *Überwachung von Unix-/Linux-Servern* finden Sie in zusammengefasster Form eine Auswahl von Vorschlägen zur Überwachung von Linux-Servern.

Siehe auch

- NRPE auf Sourceforge: *http://sourceforge.net/projects/nagios/files/nrpe-2.x/*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*
- Rezept zur Überwachung von Linux-Servern: Rezept 8.1, *Überwachung von Unix-/Linux-Servern*
- Rezept zur Remote-Ausführung über SSH: Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*

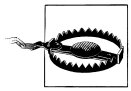
2.8 NRPE und NSClient auf Windows-Maschinen vorbereiten

Problem

Sie wollen auf Ihren Maschinen einen NRPE-Server unter Windows und/oder einen NSClient-Server unter Windows installieren und das System für die Überwachung durch Nagios/Icinga vorbereiten. Zu diesem Zweck wollen Sie den NSClient++ Monitoring Agent einsetzen.

Lösung

Für die Überwachung eines Windows-Servers mit NSClient++ über NRPE (Nagios Remote Plugin Executor) müssen Sie auf jedem zu überwachenden Windows-Server einen entsprechenden Dienst installieren. Wir zeigen Ihnen im Folgenden die Installation und Konfiguration von NSClient++ ab Version 0.4.



Vorversionen: Häufig unterscheiden sich Versionen nur wenig voneinander. Bei NSClient++ markiert der Versionssprung auf 0.4 jedoch im Vergleich zu vorherigen Versionen erhebliche Änderungen bezüglich der Möglichkeiten, der Installation und der Konfiguration. Dies ist bedingt durch eine komplette Neuimplementation von NRPE.Wenn Sie mit einer älteren Version arbeiten, treffen die Aussagen in diesem Rezept unter Umständen nicht zu. Sie können allerdings eine vorhandene Konfigurationsdatei der Version 0.3.x in Version 0.4.x weiterverwenden. Weitere Informationen zur Migration von Version 0.3.x auf Version 0.4.x finden Sie unter *http://www.nsclient.org/nscp/wiki/doc/migrate*.

Prinzipiell bietet NSClient++ unterschiedliche Varianten, um mit einem Monitoring-Server von Nagios/Icinga zu kommunizieren. Diese können Sie auch nach Ihren lokalen Anforderungen miteinander kombinieren. Wir beschreiben in diesem Rezept die Einrichtung für die Abfrage über NRPE als die vom Projekt empfohlene Methode. Dabei fungiert NSClient++ eben als NRPE-Agent und Sie können diesen über das Plugin `check_nrpe` abfragen. Im Folgenden werden wir die entsprechende Konfiguration von NSClient++ auf dem zu überwachenden Windows-Server erläutern. Die darauf aufbauende Verwendung des entsprechenden Plugins beschreiben wir in Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*.

Neben NRPE werden folgende weitere Methoden von NSClient++ unterstützt:

- Abruf über NSClient mit `check_nt`: NSClient (ohne ++) wird seit Ende 2009 nicht mehr weiterentwickelt, ist aber weit verbreitet. Sie können bei der Installation von NSClient++ das Modul NSClient-Server mitinstallieren. Von Vorteil ist die Nutzung über NSClient aber nur, wenn Sie sowieso schon Prüfungen mittels `check_nt` durchführen, oder die Konfiguration von NRPE scheuen. Die Nutzung des Plugins `check_nt` beschreiben wir für diesen Fall in Rezept 5.9, *Überwachung einer Windows-Maschine mit NSClient (check_nt)*. Beispiele einer Anwendung wurden in Rezept 8.2, *Überwachung von Windows-Maschinen* angeführt.
- Senden an NSCA-Server: Dies ist eine Variante, bei der auf der Monitoring-Maschine ein Nagios Service Check Acceptor (NSCA)-Server laufen muss. An diesen werden die Ergebnisse vom Client aus gesendet. Für die Ausführung der Prüfungen ist dann NSClient++ zuständig, und nicht der Monitoring-Server. Das Programm agiert hier als eigene Monitoring-Instanz, die die Prüfungen zeitlich plant und durchführt. Die Ergebnisse zur Verarbeitung werden dann als passive Ergebnisse an den eigentlichen Monitoring-Server versendet. Auf verteilte Konfigurationen werden wir in diesem Buch nicht weiter eingehen.

Im Folgenden werden wir die Installation und Konfiguration von NSClient in der Version 0.4 für den Einsatz mit NRPE beschreiben.

Die Installation von NSClient++

Laden Sie sich zunächst das Programm von den Seiten des Projekts unter *http://nsclient.org/nscp/downloads* herunter. Dabei stehen Ihnen ZIP- und MSI-Dateien in Versionen für 32-Bit- und 64-Bit-Systeme zur Verfügung.

Betriebssystem-Version: Achten Sie auf die ihrem System entsprechende, richtige Version. Die 32-Bit-Version würde auf einem 64-Bit-System laufen, so dass Sie einen solchen Fehler nicht unbedingt sofort bemerken. Sie hätten dann aber beispielsweise keinen Zugriff auf die Ereignisprotokolle. Prüfen Sie deshalb noch einmal, welche Version von Windows auf der Maschine installiert ist (unabhängig von der Hardware) und wählen Sie die entsprechend passende Version.

Wir empfehlen Ihnen an dieser Stelle die Verwendung der MSI-Datei, da Sie sich hier im Gegensatz zur ZIP-Datei ein paar Arbeitsschritte ersparen.

Deinstallation von Vorversionen: Sie müssen eine eventuell vorhandene ältere Version von NSClient++ in jedem Fall deinstallieren, bevor Sie eine neue Version installieren.

Wählen Sie nach dem Ausführen der MSI-Datei zunächst die Installationsvariante `Typical`. Anschließend akzeptieren Sie den Vorschlag, eine Beispielkonfiguration zu installieren. Das angegebene `ini://${exe-path}/nsclient.ini` verweist dabei normalerweise auf das Verzeichnis *C:\Program Files\NSClient++*.

Tragen Sie im nächsten Dialog wie in Abbildung 2-6 zunächst die Adresse Ihres Monitoring-Servers ein und geben Sie das Passwort an, das Sie für die Zugriffe verwenden möchten. Die Option `Enable common check plugins` sollten Sie setzen, um typische Abfragen wie CPU-Last, Festplattenauslastung und Ähnliches zu aktivieren. Bei den Abfrage-Methoden gehen wir hier wie dargestellt von `Enable NRPE server (check_nrpe)` aus. Sofern Sie außerdem `check_nt` ausprobieren möchten, können Sie ebenfalls `Enable nsclient server (check_nt)` aktivieren.

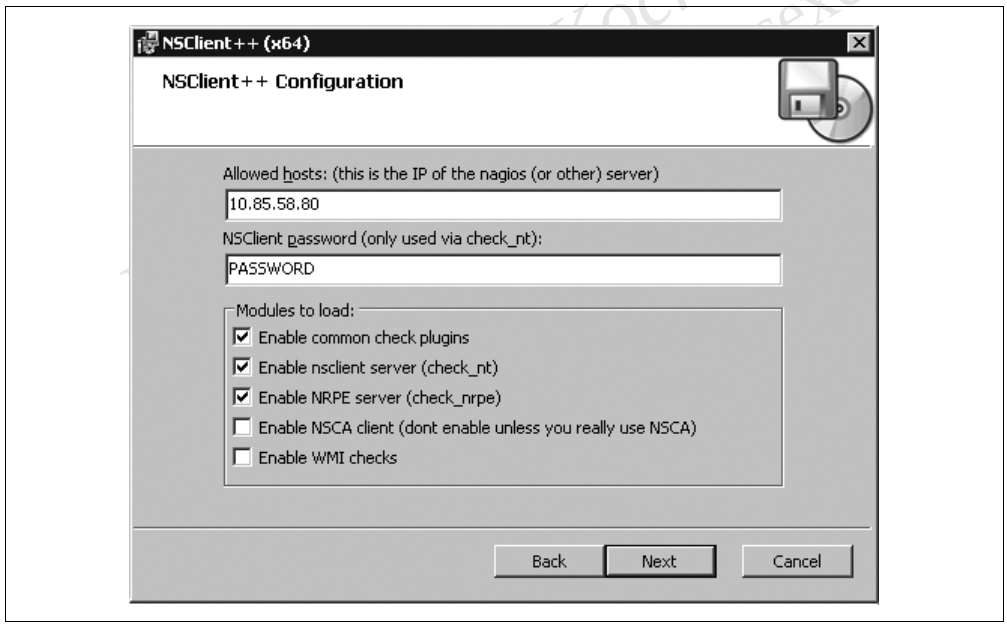


Abbildung 2-6: Dialog von NSClient++ zur Einrichtung

NSClient++ wird nach der Installation automatisch gestartet. Bei der Installation wird des Weiteren automatisch die Konfiguration der Windows-Firewall entsprechend angepasst.

Die Konfiguration von NSClient++

Bereits ohne weitere Konfiguration sollten Sie nun von Ihrer Monitoring-Maschine aus auf die ersten Daten zugreifen können. Testen Sie wie folgt den prinzipiellen Zugriff von der Monitoring-Maschine durch einen Aufruf der Plugins `check_nrpe` (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*):

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nrpe -H 10.85.58.113 -t 60 -p 5666
I (0,4,1,63 2012-10-24) seem to be doing fine...
```

Wenn Sie den NSClient-Server ebenfalls installiert haben, testen Sie diesen durch Aufruf des Plugins `check_nt` (siehe Rezept 5.9, *Überwachung einer Windows-Maschine mit NSClient (check_nt)*):

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nt -H 10.85.58.113 -p 12489 -s secretpass
-v CLIENTVERSION
NSClient++ 0,4,1,63 2012-10-24
```

An der Antwort können Sie erkennen, ob eine erfolgreiche Kommunikation mit dem installierten NRPE-Dienst beziehungsweise NSClient-Dienst stattgefunden hat. Wenn Sie hier eine Fehlermeldung erhalten, prüfen Sie die Verbindung. Eine Firewall muss sowohl ping (icmp echo request) als auch die Kommunikation über den NRPE port tcp/5666 (und optional den NSClient port tcp/12489) vom Monitoring-Server aus zulassen. Prüfen Sie gegebenenfalls auch, ob die bei der Installation von NSClient++ angegebene IP-Adresse des Monitoring-Servers und das definierte Passwort korrekt in der Konfigurationsdatei hinterlegt sind. Die Konfigurationsdatei werden wir im Folgenden beschreiben.

Die weitere Konfigurationen nehmen Sie dann in der Datei *nsclient.ini* vor. Diese Datei befindet sich im Installationsverzeichnis, in diesem Fall unter *C:\Program Files\NSClient++*. Im Ordner *modules* finden Sie dort auch die für die Prüfungen zur Verfügung stehenden Module. Eine Auswahl dieser haben wir Ihnen in Tabelle 2-1 zusammengestellt und erläutert.

Tabelle 2-1: Durch NSClient++ installierte Prüfmodule

Datei	Beschreibung
CheckDisk.dll	Testet auf Festplattennutzung und Dateigrößen (CheckFileSize, CheckFiles, CheckDriveSize)
	Testet nach Warnungen und Fehler in der Windows- Ereignisanzeige (nur NRPE)
CheckExternalScripts.dll	Wrapper um Aliases und externe Scripte auszuführen (unterstützt werden Visual Basic, PowerShell und Batch)
CheckHelpers.dll	Hilfefunktionen zur Erweiterung anderer Tests (nur NRPE)
CheckNSCP.dll	Zustandsüberprüfung des NSCP Agenten
CheckSystem.dll	Prüft auf Grundlegende Systemdaten wie Betriebszeit, CPU Auslastung, Hauptspeichernutzung, Leistungsindikatoren (Counter), Dienste- und Prozesszustände (checkUpTime, checkCPU, checkMem, checkCounter, checkProcState, checkServiceState)
NRPEServer.dll	Implementiert NRPE (check_nrpe)
NSClientServer.dll	Implementiert NSClient (check_nt)

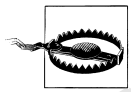
In der Konfigurationsdatei wurden diese Module bereits für Sie aktiviert. Der entsprechende Abschnitt in der Konfigurationsdatei `nsclient.ini` sah nach unserer Installation wie folgt aus:

```
[/modules]
CheckDisk = 1
CheckEventLog = 1
CheckExternalScripts = 1
CheckHelpers = 1
CheckNSCP = 1
CheckSystem = 1
NRPEServer = 1
NSClientServer = 1
```

Im nächsten Abschnitt `[/settings/default]` finden Sie Ihre Einstellungen zu den Maschinen mit den entsprechenden Berechtigungen wieder. Nur die hier aufgeführten IP-Adressen dürfen auf die installierten Dienste zugreifen.

```
[/settings/default]
allowed hosts = 10.85.58.80
password = PASSWORD
use ssl = true
```

Hier muss auf jeden Fall Ihr Nagios/Icinga-Monitoring-Server eingetragen sein. Sie können aber bei Bedarf auch weitere Geräte hinzufügen. Das NSClient++-Passwort, dass Sie bei der Installation angegeben haben, sollte bereits eingetragen sein. Dieses wird allerdings nur für `check_nt` benötigt. Die Option für SSL-Verschlüsselung ist nur relevant für `check_nrpe`. Die Verschlüsselung ist bei NSClient++ standardmäßig aktiviert.



»**Nasty Characters**«: NSClient++ verbietet standardmäßig die Übergabe von Parametern über das Netzwerk. Diese Sperre und das Filtern von Sonderzeichen, hier bezeichnet als *nasty characters*,lässt sich zu Testzwecken mit den folgenden Optionen außer Kraft setzten. Allerdings raten wir Ihnen aus Sicherheitsgründen dringend dazu, die Sperre beizubehalten, indem Sie beide Parameter auf den Wert *false* belassen. Besser ist es, Sie setzen diesen Abschnitt höchstens temporär ein, um neue Kommandos bequemer testen zu können:

```
[/settings/NRPE/server]
allow arguments = true
allow nasty characters = true
```

Wir empfehlen Ihnen die Nutzung von Alias-Kommandos. In diesen hinterlegen Sie das auszuführende Kommando, alle benötigten Parameter und die zu übergebenden Argumente auf dem zu überwachenden System.

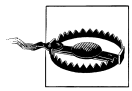
Folgend haben wir eine Auswahl von bei NSClient++ mitgelieferten Alias-Kommandos mit häufig genutzten Tests zusammengestellt. Auf diese gehen wir in Rezept 8.2, *Überwachung von Windows-Maschinen* genauer ein. Dort erläutern wir auch, wie Sie Alias-Kommandos selber definieren:

```
[/settings/external scripts/alias]
alias_cpu = checkCPU warn=80 crit=90 time=5m time=1m time=30s
alias_disk = CheckDriveSize MinWarn=10% MinCrit=5% CheckAll FilterType=FIXED
alias_event_log = CheckEventLog file=application file=system MaxWarn=1 MaxCrit=1
"filter=generated gt -2d AND severity NOT IN ('success', 'informational') AND source !=
'SideBySide'" truncate=800 unique descriptions "syntax=%severity%: %source%: %message%
(%count%)"alias_mem = checkMem MaxWarn=80% MaxCrit=90% ShowAll=long type=physical
type=virtual type=paged type=page
alias_process = checkProcState "$ARG1$=started"
alias_service = checkServiceState CheckAll
alias_up = checkUpTime MinWarn=1d MinWarn=1h
alias_volumes = CheckDriveSize MinWarn=10% MinCrit=5% CheckAll=volumes FilterType=FIXED
```

Nutzung externer Scripte

Um externe Scripte verarbeiten zu können, muss das Modul `CheckExternalScripts` geladen sein. Bei einer Standardinstallation ist dies der Fall. Beginnen Sie damit, den Abschnitt `[/settings/external scripts]` mit der Option `timeout` einzurichten. Der Standardwert ist 60 Sekunden. Sollten externe Scripte diesen vordefinierten Wert überschreiten, dann können Sie diesen nun bei Bedarf erhöhen. Dies kann beispielsweise bei der Überwachung von Windows-Updates notwendig sein.

```
[/settings/external scripts]
timeout = 60
```



Timeout: Beachten Sie, dass sich der Timeout ebenfalls auf Seite Ihres Monitoring-Systems für alle Checks definieren lässt. Dieser Wert ist oft auf 10 Sekunden gesetzt, lässt sich aber meist mit dem Parameter `-t` anpassen. Wenn dieser globale Timeout kleiner ist als der Wert für den Timeout auf Seite des NRPE-Clients, dann werden die Checks immer abgebrochen.

Des Weiteren benötigen Sie den Abschnitt `[/settings/external scripts/wrappings]`, in dem Sie definieren, wie NSClient++ externe Scripte verarbeiten soll. Dazu weisen Sie den Dateiendungen einem Interpreter zu. Unterstützt werden zu Zeit Batch-Dateien, Powerscript und Visual Basic.

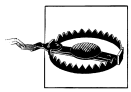
```
[/settings/external scripts/wrappings]
bat = scripts\\%SCRIPT% %ARGS%
ps1 = cmd /c echo scripts\\%SCRIPT% %ARGS%; exit($lastexitcode) |
powershell.exe -command -
vbs = cscript.exe //T:30 //NoLogo scripts\\lib\\wrapper.vbs %SCRIPT% %ARGS%
```

Als Nächstes benötigen Sie die Abschnitte `[/settings/external scripts/wrapped scripts]` und `[/settings/external scripts/alias]`. Im Abschnitt `wrapped scripts` definieren Sie zunächst ein Kommando, das auf das aufzurufende Script verweist, und erstellen anschließend im Abschnitt `alias` ein entsprechendes Alias-Kommando, dem Sie die gewünschten Parameter übergeben.

```
[/settings/external scripts/wrapped scripts]
check_updates = scripts\\check_updates.vbs
[/settings/external scripts/alias]
alias_updates = check_updates -warning 0 -critical 0
```

Beispielkonfiguration

Als Startpunkt für eine eigene Konfiguration können Sie mit folgendem Kommando eine Beispielkonfiguration mit allen möglichen Abschnitten und Schlüsselworten erstellen:



Backups: Die vorhandene Konfiguration wird mit diesen Kommandos überschrieben. Erstellen Sie unbedingt Backups, bevor Sie Ihre Konfigurationsdatei neu schreiben oder automatisiert modifizieren.

```
C:\Program Files\NSClient++>nscp settings --generate --add-defaults --load-all
```

Dabei werden Sie feststellen, dass diese Datei zwar sehr umfangreich, aber auch ungemein hilfreich ist. Unter Umständen müssen Sie allerdings dafür noch andere Pakete installieren (beispielsweise Python, siehe *http://www.python.org/ftp/python/2.7.3/*).

Sollten Sie bereits eine Konfigurationsdatei erstellt haben, dann können Sie mit folgendem Kommando Ihre Datei um die oben generierten Abschnitte und Schlüsselworte ergänzen. Typische Konfigurationsfehler, wie zum Beispiel ein Schlüsselwort in einem falschen Abschnitt, können Sie dann leichter identifizieren. In der so entstandenen Datei können Sie dann Anpassungen nach Ihren Vorstellungen vornehmen:

```
C:\Program Files\NSClient++>nscp settings --generate --add-defaults
```

Wenn Sie die Konfiguration abgeschlossen haben, können Sie die generierten Teile der Datei mit folgendem Kommando wieder entfernen.

```
C:\Program Files\NSClient++>nscp settings --generate --remove-defaults
```

Testen der Konfiguration

Nachdem die Konfiguration abgeschlossen wurde, sollten Sie die Konfigurationsänderungen mit folgendem Kommando überprüfen:

```
C:\Program Files\NSClient++>nscp settings --validate
```

Nach Änderungen in der Konfiguration müssen Sie NSClient++ neu starten, bevor diese wirksam werden. Den Neustart können Sie entweder über den Windows-Dienstemanager `Dienste` beziehungsweise `Services` oder wie folgt über die Kommandozeile bewerkstelligen:

```
C:\Program Files\NSClient++>net stop nscp
NSClient++ (x64) wird beendet..
NSClient++ (x64) wurde erfolgreich beendet.

C:\Program Files\NSClient++>net start nscp
NSClient++ (x64) wird gestartet.
NSClient++ (x64) wurde erfolgreich gestartet.
```

Optional können Sie NSClient++ mit nachfolgendem Kommando im Testmodus starten. Es werden nun alle Meldungen auf der Konsole ausgegeben. Das Kommando `find` reduziert die Ausgabe auf Meldungen mit dem Schlüsselwort `Error`:

```
C:\Program Files\NSClient++>nscp test | find "ERROR"
```

Sie werden feststellen, dass dies hilfreich bei der Fehlersuche ist.

Diskussion

Sie lassen mit der Installation von NSClient++ Zugriffe auf Ihre Windows-Maschinen aus dem Netzwerk zu. Deshalb sollten Sie auf eine sichere Nutzung achten. Wir fassen im Folgenden deshalb noch einmal für Sie zusammen, was Sie dabei beachten sollten.

Zunächst bedeutet dies für Sie, falls irgend möglich nicht das veraltete NSClient (ohne ++), beziehungsweise das Plugin check_nt, für Zugriffe zu nutzen, da die Kommunikation hier unverschlüsselt stattfindet und somit auch der mögliche Passwortschutz kaum von Wert ist, weil dieses im Klartext übertragen wird. Wenn Sie NSClient dennoch weiterhin im Einsatz haben, überlegen Sie, ob Sie die fraglichen Maschinen nicht langfristig auf NRPE umstellen. Aktualisieren Sie dann gegebenenfalls NSClient++ auf den Windows-Maschinen und passen Sie die Konfigurationsdateien wie schon vorher beschrieben an.

NRPE bietet Ihnen eine per SSL verschlüsselte Datenübertragung. Das Plugin `check_nrpe` von den NagiosPlugins unterstützt allerdings nur ein fest einkompiliertes Zertifikat. Um ein eigenes Zertifikat verwenden zu können, müssten Sie eine von NSClient++ gelieferte Variante dieses Plugins auf Ihrem Monitoring-Server verwenden. Konsultieren Sie dazu die Dokumentation von NSClient++.

Bei der Nutzung von NRPE sollten Sie auf jeden Fall die folgenden Punkte beachten:

- Verzichten Sie auf die Optionen `allow arguments = true` und `allow nasty characters = true`. Führen Sie Checks stattdessen über Alias-Kommandos durch.
- Verzichten Sie auf alle Module und Optionen, die Sie nicht unbedingt benötigen.
- Nutzen Sie SSL, um verschlüsselte Verbindungen zu verwenden (Option `ssl=1`).
- Beschränken Sie den Zugriff auf wenige IP-Adressen (Option `allowed_hosts`). Nutzen Sie hierzu die IP-Adresse selbst und nicht etwa den Hostnamen, da sich anderenfalls dieser Schutz über sogenanntes DNS-Spoofing aushebeln lässt.

DNS-Spoofing: Bei DNS-Spoofing handelt es sich um einen Angriff auf das Domain Name System, bei dem der Angreifer die Zuordnung von IP-Adresse und Domain-Namen verfälscht. Es gibt zahlreiche, zum Teil sehr unterschiedliche Methoden, mit denen sich ein solcher Angriff realisieren lässt.

Die eigentliche Nutzung der entsprechenden Plugins erläutern wir in jeweils eigenen Rezepten: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)* und Rezept 5.9, *Überwachung einer Windows-Maschine mit NSClient (check_nt)*. In Rezept 8.2, *Überwachung von Windows-Maschinen* finden Sie in zusammengefasster Form Vorschläge zur Überwachung von Windows-Maschinen.

Siehe auch

- Webseiten des Projekts NSClient++: *http://www.nsclient.org/*
- Informationen zur Migration von Version 0.3.x auf Version 0.4: *http://www.nsclient.org/nscp/wiki/doc/migrate*
- NSClient++ auf Sourceforge: *http://sourceforge.net/projects/nscplus/files/nscplus/*
- Rezept zur Überwachung einer Maschine mit NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*
- Rezept zur Überwachung einer Maschine mit NSClient: Rezept 5.9, *Überwachung einer Windows-Maschine mit NSClient (check_nt)*
- Rezept zur Überwachung von Windows-Maschinen: Rezept 8.2, *Überwachung von Windows-Maschinen*

2.9 SSH auf Linux-Maschinen vorbereiten

Problem

Sie möchten auf einer Linux-Maschine SSH installieren.

Lösung

Die Anwendung SSH (Secure SHell) ermöglicht Ihnen, die Kommunikation zwischen zwei Systemen zu verschlüsseln. Wir werden Ihnen im Folgenden zeigen, wie Sie SSH einrichten und nutzen können.

Installation

SSH ist in Form von OpenSSH auf Linux-Servern häufig schon von Hause aus installiert. Ansonsten können Sie die Pakete für den Client und/oder den Server manuell installieren (hier unter Debian/Ubuntu mit `aptitude`):

```
root@server:~# aptitude install openssh-server
[...]
root@moni:~# aptitude install openssh-client
[...]
```

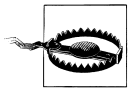
Von einer Maschine mit dem SSH-Client können Sie sich dann auf einer Maschine mit dem SSH-Server anmelden. SSH wird Sie bei der ersten Verbindung auffordern, die

Authentizität des Servers anhand eines Fingerabdrucks des genutzten Schlüssels zu bestätigen. Damit Sie dies tun können, lassen Sie sich zunächst den Fingerabdruck des bei der Installation automatisch erzeugten Schlüssels anzeigen:

```
root@server:~# ssh-keygen -lf /etc/ssh/ssh_host_rsa_key
2048 f3:91:df:7f:57:ee:fb:cd:0d:4c:51:a3:f4:56:1c:52 /etc/ssh/ssh_host_rsa_key.pub (RSA)
```

Über den Befehl `ssh` können Sie sich jetzt von einer anderen Maschine verbinden. Sofern Sie dabei nur die Zielmaschine als Parameter angeben, wird SSH davon ausgehen, dass Sie den gleichen Benutzernamen wie aktuell auf der Client-Maschine nutzen möchten. Setzen Sie andernfalls den gewünschten Nutzernamen (hier *root*) gefolgt von einem `@` direkt vor den Servernamen (hier *server*).

```
icinga@moni:~$ ssh root@server
The authenticity of host 'server (10.85.58.1)' can't be established.
RSA key fingerprint is f3:91:df:7f:57:ee:fb:cd:0d:4c:51:a3:f4:56:1c:52.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'server,10.85.58.1' (RSA) to the list of known hosts.
root@server's password:
Linux server 2.6.32-5-amd64 #1 SMP Thu Nov 3 03:41:26 UTC 2011 x86_64
[...]
```



SSH Fingerabdrücke: Wir empfehlen Ihnen, die Fingerabdrücke der verwendeten Schlüssel zu vergleichen. Anderenfalls ist ein potentieller Angreifer durch einem Man-in-the-middle-Angriff in der Lage, Ihre Kommunikation unverschlüsselt zu verfolgen. Wir sehen häufig, dass dieser Schritt übersprungen wird.

Der SSH-Client weist Sie mit einer Warnung entsprechend darauf hin, dass Sie dem entsprechenden Schlüssel für den fraglichen Server in Zukunft immer vertrauen werden. Der Schlüssel des Servers wird dabei auf dem Client im Heimatverzeichnis des jeweiligen Benutzers in die Datei *./.ssh/known_hosts* geschrieben. Sie werden feststellen, dass die Abfrage dazu bei der nächsten Verwendung nicht mehr erfolgt:

```
icinga@moni:~$ ssh root@server
root@server's password:
Linux server 2.6.32-5-amd64 #1 SMP Thu Nov 3 03:41:26 UTC 2011 x86_64
[...]
```

Falls Sie die Abfrage zu dem Fingerabdruck für diesen Server zu einem späteren Zeitpunkt erneut erhalten, sollten Sie sehr sorgfältig überprüfen, woran das liegt. Beachten Sie dabei, dass diese Überprüfung an die IP-Adresse des Servers gebunden ist. Aus diesem Grund führt eine Änderung der Serveradresse ebenfalls jedes Mal erneut zu dieser Abfrage.

Um SSH für das Monitoring einsetzen zu können, müssen Sie außerdem die interaktive Passwort-Abfrage umgehen. SSH sieht dies im Sinne einer Authentifizierung durch Schlüssel vor. Dabei muss auf dem Server der berechtigte Schlüssel für den Zugriff hinterlegt werden. SSH setzt kryptografisch auf einem Verfahren auf, das jeweils Schlüsselpaare bestehend aus einem öffentlichen (public key) und einem privaten (private key oder auch secret key) Teil vorsieht. Um also einem Benutzer einen Zugriff ohne Passwortabfrage zu

ermöglichen, muss sich dieser Benutzer ein solches Schlüsselpaar erstellen. Der öffentliche Teil muss dann auf dem zu überwachenden Gerät hinterlegt werden. Die einzelnen Schritte werden wir Ihnen im Folgenden zeigen.

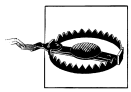
Die Erstellung von SSH-Schlüsseln

Erstellen Sie zunächst das Schlüsselpaar mit Hilfe des Befehls `ssh-keygen` und bestätigen Sie dabei alle Abfragen mit der Eingabetaste. Sie übernehmen damit den Vorgabewert für den Speicherort und geben eine leere Passphrase (keine Eingabe) an.

Eigener Schlüssel: Für menschliche Benutzer ist hier vorgesehen, dass diese bei jeder Verwendung des Schlüssels eine Passphrase eingeben. Eine Passphrase sollte länger und komplexer aufgebaut sein als ein Passwort. Treffen Sie entsprechende Maßnahmen und legen Sie auch für sich einen entsprechenden Schlüssel an. Schützen Sie diesen mit einer guten Passphrase und hinterlegen Sie ihn auf allen Maschinen, die Sie nutzen. Dies kann Ihre Arbeit sehr vereinfachen.

Da wir für das Monitoring aber eben einen Schlüssel ohne Passphrase benötigen, ist es hier notwendig, die Eingabe einer Passphrase auszusparen. Mit folgendem Kommando erstellen Sie ein RSA-Schlüsselpaar mit eine Schlüssellänge von 2048 Bits und einer leeren Passphrase. Dabei wird das erzeugte Schlüsselpaar im Verzeichnis *~/.ssh* abgelegt. Der private Schlüssel heißt hier *id_rsa* und der öffentliche Schlüssel heißt *id_rsa.pub*. Ein eventuell vorhandener Schlüssel mit der gleichen Bezeichnung wird dabei überschrieben.

```
icinga@moni:~$ ssh-keygen -b 2048 -t rsa -N "" -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /home/icinga/.ssh/id_rsa
Your public key has been saved in /home/icinga/.ssh/id_rsa.pub.
The key fingerprint is:
ba:95:b4:9c:b1:d2:99:6b:21:7f:c7:ee:6b:12:fe:68 icinga@moni
The key's randomart image is:
+--[ RSA 2048]----+
| |
| |
| |
| |
| S |
| .=.O. |
| oo@o o |
| +o.E.+ |
| ...o.B=. |
+-----------------+
```



SSH-Sicherheit: Bei der Authentifizierung ohne Passwort ist der Schlüssel selbst nicht verschlüsselt, da ja genau zu diesem Zweck die Passphrase benötigt wird. Dies bedeutet, dass jeder, der im Besitz des privaten Schlüssels ist, alles tun darf, was für diesen Schlüssel erlaubt wird. Nutzen Sie selbst keine Schlüssel ohne Passphrase und schränken Sie Schlüssel ohne Passphrase so stark wie möglich in Ihrer Verwendung ein. Lesen Sie dazu bitte die weiteren Absätze.

Die Hinterlegung von SSH-Schlüsseln

Wie Sie der Ausgabe bei der Erstellung entnehmen können, wurden die beiden Schlüsselteile jeweils in einer Datei abgelegt, die Sie sich auch anschauen können. Es handelt sich dabei um Textdateien mit jeweils einer langen Zeile:

```
icinga@moni:~$ cat .ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC44KZFzYGGBWnw2+wUKZYUqad1B4gllMksLBk3mWOmEFeKSGD8c+ACDcp5R
2pgUluFXBcwllZoPOez2neEHZM/kkftaZOeBTFY/E1UdaaWW8wXSeZTL3Y9quyvf6MSAIH9o+yqCEzYtjzj691vY1
lwsOaICc6tmK4JG6xQfQJ3XZaUF/X1dXEcRMa4yIsdkoLociCZvwXWxypOJoWfMZLBhu3rOn2dLvfrZDfhhROMWjU
LGm5Ll3oEgYzay7zfSXaL8o+qzDhH2I6jGj+5vXFZmQrnEXODqyU2tr/r/ipMzXCuorVkuFiYHpkycoqmhHguBhXr
EUSfPyY/atvOIKU3 icinga@moni
```

Den erzeugten öffentlichen Schlüssel mit der Endung *.pub* müssen Sie jetzt auf dem Server in einer speziellen Datei hinterlegen, um ihn für den Zugang zu autorisieren. In der Voreinstellung ist dies eine Datei mit dem Namen *authorized_keys* im Heimatverzeichnis des jeweiligen Benutzers im Ordner *.ssh*. Stellen Sie zunächst sicher, dass der Ordner vorhanden ist (Sie können ihn anderenfalls mit dem Befehl "`mkdir -p .ssh`" erstellen) und nur Sie Zugriffsrechte haben (mit dem Befehl "`chmod 0700 .ssh`" lassen sich diese erforderlichenfalls korrigieren). Dann kopieren Sie den öffentlichen Schlüssel in die Datei `authorized_keys`.

Mit folgendem Kommando kopieren Sie beispielsweise den öffentlichen RSA-Schlüssel `id_rsa.pub` des Nutzers Icinga auf dem Rechner `moni` in die `authorized_keys`-Datei des Nutzers `root` auf dem Rechner `server`:

```
icinga@moni:~# cat ~/.ssh/id_rsa.pub | ssh root@server "cat >> ~/.ssh/authorized_keys"
```

ssh-copy-id: Alternativ zu der Variante mit cat und ssh steht Ihnen für diese Operation auf vielen Distributionen das Kommando ssh-copy-id zur Verfügung:

```
icinga@moni:~# ssh-copy-id -i ~/.ssh/id_rsa.pub root@server
```

Der Nutzer `icinga` hat durch den Austausch des öffentlichen Schlüssels auf diese Weise zunächst unbegrenzten Zugriff auf den root-Zugang der Rechners `server` – und das ohne Passphrase.

Beschränkung des Zugangs

Um die Verwendung einzuschränken, können Sie am Anfang der Zeile eine oder mehrere durch Kommata separierte Optionen hinzufügen (siehe `man authorized_keys`):

- `command=""`

  Befehl, der für diesen Benutzer ausgeführt wird. Damit wird der Benutzer effektiv auf genau diesen einen Befehl eingeschränkt. Sämtliche vom Benutzer gesendete Kommandos werden ignoriert.

- `from=""`

  Durch Kommata separierte Liste mit Namen/Adressen, von denen aus sich dieser Schlüssel verbinden darf. Geben Sie hier am besten die IP-Adresse des Clients ein.
- `no-agent-forwarding`

  Unterbindet die Weiterleitung der Authentifizierung.
- `no-port-forwarding`

  Unterbindet die Port-Weiterleitung.
- `no-pty`

  Unterbindet die Zuweisung eines virtuellen Terminals – um daraus resultierende Fehlermeldungen beim Verbindungsaufbau zu verhindern, sollten Sie SSH dann über den Parameter `-T` mitteilen, dass es kein virtuelles Terminal anfordern soll.
- `no-user-rc`

  Unterbindet die Ausführung der Datei *~/.ssh/rc*.
- `no-X11-forwarding`

  Unterbindet die Weiterleitung von graphischen Ausgaben.

Setzen Sie für Schlüssel ohne Passphrase so viele dieser Optionen wie möglich, damit die betreffende Funktionalität lediglich auf das nötigste beschränkt wird. Als Beispiel führen wir nachfolgend ein paar Optionen an, die eine Verbindung von nur einer IP-Adresse zulassen, bei einer erfolgreichen Anmeldung den Befehl `echo SSH-Test` ausführen und die Verbindung anschließend sofort wieder beenden:

```
command="echo SSH-Test",from="moni",no-agent-forwarding,no-port-forwarding,no-user-rc,no-
X11-forwarding ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC44KZFzYGGBWnw2+wUKZYUqad1B4gllMksLBk3mWOmEFeKSGD8c+ACDcp5R
2pgUluFXBcwllZoPOez2neEHZM/kkftaZOeBTFY/E1UdaaWW8wXSeZTL3Y9quyvf6MSAIH9o+yqCEzYtjzj691vY1
lwsOaICc6tmK4JG6xQfQJ3XZaUF/X1dXEcRMa4yIsdkoLociCZvwXWxypOJoWfMZLBhu3rOn2dLvfrZDfhhROMWjU
LGm5Ll3oEgYzay7zfSXaL8o+qzDhH2I6jGj+5vXFZmQrnEXODqyU2tr/r/ipMzXCuorVkuFiYHpkycoqmhHguBhXr
EUSfPyY/atvOIKU3 icinga@moni
```

Die Ausführung gibt dann die Ausgabe des Befehls "echo SSH-Test" auf der Remote-Maschine zurück:

```
icinga@moni:~$ ssh -T -i ~/.ssh/id_rsa root@server
SSH-Test
```

Wenn Sie anstelle des Befehls `echo SSH-Test` den Aufruf eines Plugins einfügen, haben Sie einen Schlüssel für einen entsprechenden Service-Check erstellt. Um eine Überprüfung auf verfügbare, aber noch nicht installierte Aktualisierungen durchzuführen, haben wir im folgenden Beispiel das Plugin `check-apt` (siehe Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*) über SSH eingebunden:

```
command="/root/libexec/check_apt",from="moni",no-agent-forwarding,no-port-forwarding,no-
pty,no-user-rc,no-X11-forwarding ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC44KZFzYGGBWnw2+wUKZYUqad1B4gllMksLBk3mWOmEFeKSGD8c+ACDcp5R
2pgUluFXBcwllZoPOez2neEHZM/kkftaZOeBTFY/E1UdaaWW8wXSeZTL3Y9quyvf6MSAIH9o+yqCEzYtjzj691vY1
lwsOaICc6tmK4JG6xQfQJ3XZaUF/X1dXEcRMa4yIsdkoLociCZvwXWxypOJoWfMZLBhu3rOn2dLvfrZDfhhROMWjU
LGm5Ll3oEgYzay7zfSXaL8o+qzDhH2I6jGj+5vXFZmQrnEXODqyU2tr/r/ipMzXCuorVkuFiYHpkycoqmhHguBhXr
EUSfPyY/atvOIKU3 icinga@moni
```

Entsprechend wird bei einem Aufruf von SSH jetzt die Ausgabe des Plugins zurückgeliefert. Diesen Aufruf können Sie nun als Befehl für Nagios/Icinga einrichten (siehe Rezept 4.3, *Befehle hinzufügen (command)* zur Definition von Befehlen). Auf diese Weise können Sie gezielt einzelne Befehle sicher für die Remote-Ausführung konfigurieren und in Nagios/Icinga nutzen.

Diskussion

Sofern Sie, wie von uns empfohlen, die Verwendung von Schlüsseln ohne Passphrase auf jeweils einen Befehl begrenzen, werden Sie unter Umständen mehrere solcher Schlüssel einsetzen wollen. Sie müssen für die unterschiedlichen Schlüssel dann jeweils eigene Dateien angeben. Wir benennen den Schlüssel aus dem vorherigen Beispiel hier in *id_rsa-check_apt* um:

```
icinga@moni:~$ mv .ssh/id_rsa .ssh/id_rsa-check_apt
icinga@moni:~$ mv .ssh/id_rsa.pub .ssh/id_rsa-check_apt.pub
```

Damit wird SSH den Schlüssel aber zunächst nicht mehr finden, weshalb Sie beim Aufruf entsprechend angeben müssen, welchen Schlüssel Sie verwenden möchten. Nutzen Sie hierzu dann zusätzlich die Option `-i`:

```
icinga@moni:~$ ssh -T -i ~/.ssh/id_rsa-check_apt root@server
[...]
```

Sofern Sie mit mehreren Maschinen über SSH arbeiten, werden alle bestätigten Schlüssel jeweils in der Datei *~/.ssh/known_hosts* gesammelt. Sie können hier jedoch auch eine andere zu verwendende Datei festlegen und sich auf diese Weise einzelne Dateien für die jeweiligen Maschinen anlegen. Verwenden Sie dazu die Option `-o UserKnownHostsFile=datei`, hier beispielsweise, um als Datei *known_hosts.server* anzugeben:

```
icinga@moni:~$ ssh -T -i ~/.ssh/id_rsa-check_apt -o
UserKnownHostsFile="~/.ssh/known_hosts.server" root@server
[...]
```

Auf diese Weise können Sie die Ablage der für SSH relevanten Daten systematisch ordnen.

Weitere für die Automatisierung von SSH nützliche Optionen sind folgende:

- `-q`

  Mit dieser Option, die für QuietMode steht, unterdrücken Sie die Ausgabe von Warnungen und Diagnosemeldungen.

- `-o 'BatchMode yes'`

  Mit dieser Option können Sie die Optimierung für eine automatisierte Ausführung in Scripten aktivieren. Dabei wird die Passwortabfrage dann deaktiviert und alle fünf Minuten automatisch eine Keepalive-Nachricht versendet.

Siehe auch

- Hilfeseiten für das `ssh`-Kommando: `man ssh`
- Hilfeseiten für das `ssh-keygen`-Kommando: `man ssh-keygen`
- Hilfeseiten für die Konfigurationsdatei `authorized_keys`: `man authorized_keys`
- Rezept zur Überwachung von Aktualisierungen: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*

KAPITEL 3

Die Arbeit an und mit Nagios/Icinga

Bevor wir Ihnen im nächsten Kapitel die eigentlichen Konfigurationsobjekte und Dateien von Nagios/Icinga vorstellen werden, haben wir hier wichtige Grundlagen zusammengefasst. Da Ihr Monitoring-System nach der Installation aus Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten* bereits laufen sollte, können Sie sich in Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*, zunächst mit der Oberfläche vertraut machen. Sobald Sie mit den Grundzügen vertraut sind, empfehlen wir Ihnen allerdings unbedingt auch die folgenden Rezepte:

- **Vorlagen** (siehe Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*)

  Schon die automatisch erstellten Dateien verwenden das Vorlagen-System. Wie wichtig dieses im Weiteren ist, hängt sehr von der Komplexität Ihrer Umgebung ab. Sie sollten aber zumindest wissen, dass es diesen Mechanismus gibt, und ihn in den Grundzügen verstehen. Wir empfehlen Ihnen dieses Rezept zumindest zu überfliegen.
- **Ablage und Struktur** (Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*)

  Hier zeigen wir Ihnen Tipps zum Aufbau und zur Ablage der Konfigurationsdateien. Sofern Sie mehr als ein paar wenige Geräte überwachen möchten, sollten Sie sich mit diesen vertraut machen. Sie können dann entsprechende Strukturen frühzeitig aufbauen und ersparen sich unter Umständen sehr mühseliges Nachpflegen und Umrüsten.

Wir werden Sie im Folgen an entsprechenden Stellen aber auch jeweils noch einmal auf die Rezepte in diesem Kapitel verweisen.

3.1 Benutzerverwaltung für Zugriffe auf die Weboberfläche

Problem

Sie möchten verschiedenen Benutzern differenzierte Zugriffsrechte für die Weboberfläche erteilen.

Lösung

Für den Zugriff auf von Nagios/Icinga vorgehaltene Daten dient vornehmlich eine Weboberfläche, deren Benutzung wir Ihnen in den Rezepten Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web* erläutern. Zur Authentifizierung werden dabei über `http-auth` angelegte Benutzerkonten verwendet. Benutzer, die Sie dafür anlegen, können Sie parallel dazu genau so in der Definition von Kontakten verwenden (Konfigurationsobjekt contact, siehe Rezept 4.4, *Kontakte definieren (contact)*), und als Kontakte dann spezifisch Maschinen (Konfigurationsobjekt host, siehe Rezept 4.1, *Maschinen einbinden (host)*) und Diensten (Konfigurationsobjekt service, siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) zuweisen. Sofern ein Dienst keine Kontakte zugewiesen bekommen hat, werden diese implizit von der Maschine vererbt. Auf diesem Wege erhalten die betreffenden Benutzerkonten dann nur Zugriff auf Geräte beziehungsweise Dienste, in denen sie als Kontakt referenziert sind.

Icinga Classic: Bei der Weboberfläche von Icinga-Classic ist per Voreinstellung jeder Host-Kontakt berechtigt, auf alle mit der Maschine assoziierten Dienste zuzugreifen. Daher steht in Icingas CGI-Konfigurationsdatei `cgi.cfg` die Option `show_all_services_host_is_authorized_for=1` zur Verfügung. Wenn Sie also unter Icinga-Classic Kontakte auf Dienste-Ebene definieren möchten, dann sollten Sie diese Option deaktivieren.

Wir zeigen Ihnen im Folgenden, wie Sie entsprechende Benutzer anlegen, löschen und mit einem neuen Passwort versehen können. Sie müssen für die Durchführung wissen, wo die entsprechende Passwortdatei `htpasswd.users` (oder auch nur `htpasswd`) in Ihrem System zu finden ist. Sofern Sie die Installation nach einem Rezept in diesem Buch vorgenommen haben, können Sie diesen Ort der Tabelle 3-1 entnehmen.

Tabelle 3-1: Konfigurationsordner bei den unterschiedlichen Installationen

Installation	Datei
Installation Nagios aus Quellen	/usr/local/nagios/etc/
Installation Icinga aus Quellen	/usr/local/icinga/etc/
Debian undUbuntu paketbasiert Nagios	/etc/nagios3/
sonstige paketbasiert Nagios	/etc/nagios/
paketbasiert Icinga	/etc/icinga/

Webbenutzer anlegen und ändern

Um einen Webbenutzer anzulegen oder für einen vorhandenen Benutzer das Passwort zu ändern, rufen Sie das Programm `htpasswd` für Ihre Passwortdatei auf und übergeben den Benutzernamen als Parameter. Eventuell müssen Sie die Datei bei der ersten Benutzung mit Hilfe des Parameters `-c` anlegen lassen. Wenn der Benutzer noch nicht vorhanden ist, wird er dabei automatisch erstellt und das Passwort hinterlegt:

```
root@moni:~# htpasswd /usr/local/icinga/etc/htpasswd.users testbenutzer
New password:
Re-type new password:
Adding password for user testbenutzer
```

Wenn es den Benutzer hingegen schon gibt, wird das vorhandene Passwort mit dem neuen überschrieben:

```
root@moni:~# htpasswd /usr/local/icinga/etc/htpasswd.users testbenutzer
New password:
Re-type new password:
Updating password for user testbenutzer
```

Webbenutzer löschen

Um einen Webbenutzer zu löschen, rufen Sie `htpasswd` mit dem Parameter `-D` und dem entsprechenden Benutzernamen auf:

```
root@moni:~# htpasswd -D /usr/local/icinga/etc/htpasswd.users testbenutzer
Deleting password for user testbenutzer
```

Bei diesem Vorgang wird die gesamte den Benutzer betreffende Zeile aus der Datei entfernt.

Diskussion

Mit den hier gezeigten Schritten verwalten Sie die Benutzerkonten und die dazugehörigen Passwörter für den Webserver. Bevor Sie die angelegten Benutzer in der Nagios/Icinga-Konfiguration mit Maschinen (siehe Rezept 4.1, *Maschinen einbinden (host)*) und/oder Services (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) verknüpfen, haben diese dann zwar schon Zugriff auf die Oberfläche, aber eben nicht auf Daten der Maschinen oder Dienste. Dazu müssen Sie diese erst über allgemeine Konfigurationsanweisungen mit entsprechenden Rechten versehen (siehe unten).

Sie profitieren nicht nur bei der Betreuung einer sehr großen Installation von der Nutzung unterschiedlicher Benutzerkonten. Verwenden Sie diese, um Verantwortlichkeiten in Ihrer Organisation abzubilden und um dem jeweiligen Ansprechpartner selektiv Zugriff auf die Weboberfläche zu gewähren. Über die angesprochene Definition von gleichnamigen Kontakten (siehe Rezept 4.4, *Kontakte definieren (contact)*) und deren Referenzierung in der Definition von Maschinen und Services können Sie dann weiter die Benachrichtigungen entsprechend dieser Verantwortlichkeiten aufteilen. Dieser Aspekt wird für Sie mit einer wachsenden Installation schnell an Bedeutung gewinnen, denn Sie möchten sicherlich nicht mit einer Flut von Benachrichtigungen eingedeckt werden, die Sie eigentlich gar nicht direkt betreffen.

Freischaltungen in den CGI-Optionen

Über die Referenzierung in den Konfigurationsdateien zu den Maschinen und Services hinaus haben Sie noch die Möglichkeit, bestimmten Benutzern besondere Rechte einzu-

räumen. Diese Optionen zu den Rechten finden Sie in der Datei *cgi.cfg* im bereits vorher genannten Konfigurationsordner von Nagios/Icinga. Bei unserer quellenbasierten Referenz-Installation ist dies also */usr/local/icinga/etc/cgi.cfg*. Die folgenden Optionen stehen Ihnen beispielsweise bei Icinga zur Verfügung (siehe *http://docs.icinga.org/latest/de/configcgi.html*):

```
#default_user_name=guest
authorized_for_system_information=icingaadmin
authorized_for_configuration_information=icingaadmin
authorized_for_full_command_resolution=icingaadmin ### nur Icinga ###
authorized_for_system_commands=icingaadmin
authorized_for_all_services=icingaadmin
authorized_for_all_hosts=icingaadmin
authorized_for_all_service_commands=icingaadmin
authorized_for_all_host_commands=icingaadmin
#authorized_for_read_only=user1,user2
#authorized_contactgroup_for_read_only=
show_all_services_host_is_authorized_for=1 ### nur Icinga ###
show_partial_hostgroups=0 ### nur Icinga ###
```

Die meisten dieser Optionen sind selbsterklärend. Als Parameter geben Sie die durch Kommata separierte Benutzernamen an. Einige, wichtige Optionen werden wir Ihnen im Folgenden vorstellen, für die übrigen konsultieren Sie im Zweifel bitte die jeweilige Dokumenation Ihrer Installation:

- `authorized_for_all_hosts` beziehungsweise `authorized_for_all_services`

  Mit dieser Option können Sie für Benutzernamen den lesenden Zugriff auf alle Maschinen beziehungsweise alle Dienste freischalten.

- `authorized_for_all_host_commands` beziehungsweise `authorized_for_all_service_commands`

  Mit dieser Option können Sie Benutzernamen angeben, deren Benutzer generellen Zugriff auf alle über die Weboberfläche ausführbaren Befehle für alle Maschinen beziehungsweise Dienste haben.

- `authorized_for_read_only`

  Mit dieser Option können Sie für Benutzer die Anzeige und Nutzung der Kommandos der Weboberfläche unterbinden. Diese Benutzer können dann entsprechend Ihrer sonstigen Berechtigungen Maschinen und Dienste einsehen, aber keine Befehle an diese absetzen.

- `authorized_for_system_information`

  Mit dieser Option schalten Sie Benutzer für den Zugriff auf die über die Weboberfläche verfügbaren Prozessinformationen und Protokolldaten zu Nagios/Icinga frei. Diese Rechten sollten Administratoren vorbehalten sein.

- `authorized_for_configuration_information`

  Mit dieser Option erlauben Sie den referenzierten Benutzerkonten die in Nagios/Icinga hinterlegten Konfigurationsobjekte über die Weboberfläche einzusehen. Diese Rechte sollten Administratoren vorbehalten sein.

Gruppen-Freischaltung: Unter Icinga stehen Ihnen jeweils zusätzliche Konfigurationsoptionen für Kontaktgruppen (zu Kontaktgruppen siehe Rezept 4.8, *Kontakte in Gruppen zusammenfassen (contact-group)*) zur Verfügung, also etwa zu `authorized_for_all_service_commands` eine Option `authorized_contactgroup_for_all_service_commands` und so weiter. Mit diesen können Sie einfach ein Rollenkonzept umsetzen. Vergeben Sie die Rechte ausschließlich an Kontaktgruppen (für jede Rolle eine Gruppe) und weisen Sie Benutzer dann nach Bedarf den entsprechenden Gruppen zu.

Diese allgemeinen Optionen werden Sie häufig gar nicht benötigen, aber im Zweifel hilft es, wenn Sie von deren Existenz zumindest wissen. Sie können sie dann bei Bedarf aktivieren und verwenden.

Nur-Lese-Zugriff: Sie können einen Benutzer mit nur lesendem Zugriff auf alle Maschinen und/oder Dienste ausstatten, indem Sie ihn sowohl als `authorized_for_all_hosts` und `authorized_for_all_services`, aber eben auch als `authorized_for_read_only` in den CGI-Optionen eintragen. Sie müssen den Benutzer dafür natürlich wie beschrieben mit dem Befehl `htpasswd` anlegen und mit einem Passwort versehen.

Siehe auch

- CGI-Konfigurationsoptionen in der Nagios-Dokumentation: *http://nagios.sourceforge.net/docs/nagioscore/3/en/configcgi.html*
- CGI-Konfigurationsoptionen in der Icinga-Dokumenation: *http://docs.icinga.org/latest/de/configcgi.html*
- Rezepte zur Nutzung der Oberflächen: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*
- Rezepte zur Einbindung von Kontakten und Kontaktgruppen: Rezept 4.4, *Kontakte definieren (contact)* und Rezept 4.8, *Kontakte in Gruppen zusammenfassen (contact-group)*
- Rezept zur Einbindung einer Maschine: Rezept 4.1, *Maschinen einbinden (host)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*

3.2 Das Konzept von aktiven und passiven Prüfungen

Problem

Sie möchten den Unterschied zwischen aktiven und passiven Prüfungen verstehen, um nach Gegebenheit jeweils entscheiden zu können, welche Form Sie anwenden möchten.

Lösung

Nagios/Icinga sieht grundsätzlich zwei unterschiedliche Arten von Prüfungen vor:

- **Aktive Prüfung**: Bei dieser Durchführungsart plant Nagios/Icinga die zeitliche Durchführung und ruft entsprechend ein Kommando auf, das dann ein Ergebnis zurückliefert.
- **Passive Prüfung**: Bei dieser Durchführungsart kennt Nagios/Icinga das Gerät beziehungsweise den Dienst, plant aber keine Prüfungen und führt solche auch nicht durch. Stattdessen nimmt das Monitoring-System Ergebnisse von extern geplanten und durchgeführten Prüfungen entgegen, aus dem es den jeweiligen Status ermittelt.

Aktive Prüfungen sind dabei der Standard, auf den wir uns bei den meisten Rezepten in diesem Buch beziehen. Wir beschreiben also in unseren Rezepten meist wie Sie ein Plugin als aktive Prüfung einbinden. Hier erläutern wir Ihnen, in zusammenfassender Art und Weise, wie Sie eine passive Einbindung bewerkstelligen können.

Aktivierung in den Konfigurationsdateien

Damit Nagios/Icinga passive Prüfungen überhaupt beachtet, müssen Sie diese in der Hauptkonfigurationsdatei aktivieren. Bei unserer Referenzinstallation ist dies */usr/local/icinga/etc/icinga.cfg*. Wenn Sie Ihre Installation anhand einer oder analog zu einer der hier beschriebenen Installationsweisen erstellt haben, können Sie den Speicherpfad den Tabellen im Diskussionsteil des jeweiligen Rezeptes in Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten* entnehmen.

Um passive Prüfungen für Maschinen beziehungsweise Dienste zu aktivieren, müssen Sie in dieser Datei nun zunächst die nachfolgend angeführte Optionen auf den Wert 1 setzen:

```
accept_passive_service_checks=1
accept_passive_host_checks=1
```

Bei den Maschinenprüfungen analysiert Nagios/Icinga bei einem DOWN-Status normalerweise über die bekannten Abhängigkeiten anhand einer Erreichbarkeitslogik einen Zustand für untergeordnete Maschinen. Damit vermeidet Nagios/Icinga Prüfungen von Maschinen, die aufgrund von Ausfällen gar nicht erreichbar sind (diese bekommen dann automatisch den Status UNREACHABLE). Dieser Mechanismus wird bei passiven Maschinenprüfungen standardmäßig außer Kraft gesetzt und stattdessen fest der übermittelte Zustand angenommen. Sie können dieses Verhalten ändern, wenn Sie die nachfolgende Option auf den Wert `1` setzen:

```
translate_passive_host_checks=0
```

Zu den Auswirkungen dieser Option und den entsprechenden Anwendungsszenarien enthält die Dokumentation eigene Seiten (siehe *http://docs.icinga.org/latest/de/passivestatetranslation.html*). Konsultieren Sie diese bitte, bevor Sie den entsprechenden Mechanismus einsetzen.

Unterschiede bei den Maschinen- und/oder Dienstdefinitionen

Bei einer passiven Prüfung ist im Gegensatz zu einer aktiven Prüfung keine Kommandodefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) erforderlich, da die Prüfung selbst ja nicht von Nagios/Icinga ausgeführt werden wird. Ausnahme sind hier die Frische-Prüfungen (siehe unten).

Dennoch muss das Gerät oder der Dienst in Nagios bekannt sein, so dass Sie (wie bei aktiven Prüfungen) eine entsprechende Maschinendefinition (siehe Rezept 4.1, *Maschinen einbinden (host)*) und Dienstdefinition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) benötigen. Dabei müssen Sie in den entsprechenden Definitionen die Option

```
passive_checks_enabled 1
```

setzen. Sie können diese Option auch über die Einbindung einer entsprechenden Vorlage (siehe Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*) setzen. Sofern Sie mehrere Maschinen und/oder Dienste passiv überwachen, sollten Sie sich eine entsprechende Vorlage erstellen.

Übermittlung der Ergebnisse

Die verwendete Programmierschnittstelle (Application Programming Interface, kurz API) ist für passive Prüfungen eine andere als bei aktiven (zu letzteren siehe Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*). Die Übermittlung der passiven Ergebnisse erfolgt über das sogenannte *external command file*, das Sie in der Hauptkonfigurationsdatei (mit der Option `command_file`) festlegen. Bei unserer Referenzinstallation ist dies als Vorgabewert die Datei */usr/local/icinga/var/rw/icinga.cmd*.

Es handelt sich hier faktisch um eine Datei, in der Nagios/Icinga regelmäßig nach neuen Ergebnissen schaut, um diese dann auszuwerten. Auch von Ihnen in der Weboberfläche abgesetzte Befehle werden beispielsweise über diese Datei an Nagios/Icinga übermittelt. Das dabei für passive Ergebnisse genutzt Format erläutern wir Ihnen im Folgenden.

Bei Maschinen benötigt Nagios/Icinga für die Verarbeitung den Zeitpunkt, den ermittelten Status, den Namen der Maschine und die Rückgabe des Plugins. Sie übergeben diese zusammen mit dem Befehl `PROCESS_HOST_CHECK_RESULT` im folgenden Format:

```
[unix-time] PROCESS_HOST_CHECK_RESULT;Maschinen-Name;Maschinen-Status;Textausgabe
```

Als Wert für `unix-time` übergeben Sie dabei den Zeitpunkt der Messung oder den des Eingangs des Ergebnisses als sogenannten Unix-Timestamp (Sekunden seit 1970-01-01 0:00). Diesen können Sie für den aktuellen Zeitpunkt beispielsweise mit dem Befehl `date` und für das entsprechende Format mit der Option `+%s` ermitteln:

```
icinga@moni:~$ date +%s
1361125380
```

Bei Ergebnissen für Dienste verwenden Sie abweichend den Befehl `PROCESS SERVICE CHECK RESULT` und geben zusätzlich den Namen des Services an:

```
[unix-time] PROCESS_SERVICE_CHECK_RESULT;Maschinen-Name;Service-Name;Service-Status;
Textausgabe
```

Bevor Sie entsprechende Tests durchführen können, muss die Maschine (und gegebenenfalls der Dienst) in Nagios/Icinga definiert sein. Ansonsten wird das Prüfungsergebnis verworfen und ein Logeintrag generiert.

Frische-Prüfung – Mindesthaltbarkeit passiver Prüfungen (freshness)

Da Nagios/Icinga die Prüfungen bei passiver Durchführung nicht selbst vornimmt, kann es auch nicht wissen, wann die nächste Prüfung durchgeführt werden soll. Als Folge bleibt beim Ausbleiben von Prüfungen immer der zuletzt bekannte Status erhalten. Sofern bei Ihrer Prüfung ein Ausbleiben von Prüfungsergebnissen an sich als Problem erkannt werden soll, können Sie die sogenannte Frische-Prüfung (siehe auch *http://docs.icinga.org/latest/de/freshness.html*) verwenden.

Dabei wird Nagios/Icinga dann in regelmäßigen Abständen überprüfen, wann das letzte Prüfungsergebnis eingegangen ist und bei zu alten Ergebnissen einen aktiven Check ausführen. Deshalb müssen Sie, um die Frische-Prüfung verwenden zu können, sicherstellen, dass die fragliche Prüfung auch aktiv ausgeführt werden kann. Den Befehl dazu hinterlegen Sie wie bei aktiven Prüfungen über die Option `check_command`.

Die Aktivierung der Frische-Prüfung und die dafür zu verwenden Intervalle geben Sie für Maschinen in deren Definition (siehe Rezept 4.1, *Maschinen einbinden (host)*) mit

```
check_host_freshness=1
host_freshness_check_interval=60
```

und für Dienste in deren Definition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) mit

```
check_service_freshness=1
service_freshness_check_interval=60
```

an. Sie können den Wert für das Interval auch auslassen, dann wird Nagios/Icinga ihn über die Werte von `check_interval` und `retry_interval` bestimmen (und dabei zusätzlich den in der Hauptkonfigurationsdatei hinterlegten Wert von `additional freshness latency` berücksichtigen!). Wir empfehlen Ihnen aber, diese Intervalle (gegebenenfalls über Vorlagen, siehe Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*) über die eben angeführten Optionen jeweils anzugeben, da Sie so bei späteren Untersuchungen leichter feststellen können, welches Intervall verwendet wird. Des Weiteren müssen Sie wie erwähnt sowohl bei Maschinen als auch bei Diensten zusätzlich das Kommando angeben, welches bei passiven Prüfungen sonst nicht nötig ist.

Dummy-Kommando für passive Prüfungen: Sofern Sie die Prüfung gerade deshalb passiv ausführen, weil eine aktive Prüfung nicht möglich ist, können Sie anstelle eines entsprechenden Plugins ein sogenanntes Dummy-Kommando angeben. In der Icinga-Dokumentation wird dieses Vorgehen im Artikel zu den Frische-Prüfungen (siehe *http://docs.icinga.org/latest/de/freshness.html*) unter Verwendung des Plugins `check_dummy` beschrieben.

Diskussion

Passive Prüfungen sind eine wertvolle Ergänzung zu den üblichen aktiven Prüfungen. Neben dem Einsatz bei verteilten Umgebungen, auf die wir in diesem Buch nicht weiter eingehen werden, sind sie unter anderem in den folgenden Szenarien wichtig:

- Sie können die Prüfung nicht aktiv ausführen, weil Sie den Dienst, die Maschine oder das Netzwerk von der Monitoring-Maschine aus nicht erreichen können. Sie können die Prüfung dann in dem Remote-Netz durchführen und das Ergebnis passiv übermitteln, sofern umgekehrt von dort ein Zugriff auf die Monitoring-Maschine möglich ist.
- Die fragliche Prüfung dauert sehr lange, so dass Sie Nagios/Icinga nicht während der Ausführung warten lassen möchten. Wir haben in der Vergangenheit beispielsweise WLAN-Controller über ein Script (Programm) abgefragt und daraus hunderte von Ergebnissen für einzelne WLAN-Stationen ermittelt. Diese hat das Script dann jeweils passiv an ein Monitoring-System übergeben. Bei aktiver Abfrage wären in diesem Fall sonst hunderte Abfragen bei den WLAN-Controllern eingegangen, was wir vermeiden wollten.
- Sie überwachen Werte, die sich nur sehr unregelmäßig ändern, so dass eine regelmäßige Abfrage nicht nötig und womöglich mit zusätzlichen Kosten verbunden ist. Wenn Sie beispielsweise Messwerte über einen kostenintensiven Kommunikationskanal übertragen, beispielsweise über Satellit, und wöchentlich nur eine Änderung erwarten, können Sie mit der passiven Übertragung bei der Änderung unter Umständen erhebliche Einsparungen erzielen.

Die passiven Prüfungen sind in der Umsetzung nicht trivial. Sie müssen insbesondere Folgendes beachten:

- Bei den Voreinstellungen werden aktive Prüfungen für die jeweilige Maschine beziehungsweise den jeweiligen Dienst nicht mehr ausgeführt .
- Bei Maschinen-Prüfungen wird die Erreichbarkeitslogik für diese Maschine außer Kraft gesetzt.
- Wenn Sie die Frische-Prüfung einsetzen wollen, sollten Sie zusätzlich eine aktive Prüfung ermöglichen.

Addons für passive Prüfungen: Das Addon Nagios Service Check Acceptor (NSCA) dient der Verarbeitung extern übermittelter (also passiver) Ergebnisse. Die Icinga-Dokumentation verfügt über eine eigene Dokumentation zu NSCA (siehe *http://docs.icinga.org/latest/de/nsca.html*), das wir in diesem Buch nicht weiter behandeln werden.

Siehe auch

- Kapitel mit den Installationsrezepten: Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*
- Rezepte zu Maschinen, Dienst- und Befehlsdefinitionen: Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* und Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zur Einführung in die von Nagios/Icinga verwendete API: Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*
- Passive Checks in der Icinga-Dokumentation: *http://docs.icinga.org/latest/de/passivechecks.html*
- Erreichbarkeitslogik in der Icinga-Dokumentation: *http://docs.icinga.org/latest/de/networkreachability.html*
- Rezept zur Vereinfachung der Konfiguration mit Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Frische-Prüfung in der Icinga-Dokumentation: *http://docs.icinga.org/latest/de/freshness.html*
- Icinga-Dokumentation zum Addon NSCA: *http://docs.icinga.org/latest/de/nsca.html*

3.3 Nutzung der Weboberflächen von Nagios und Icinga-Classic

Problem

Sie möchten die Weboberfläche Ihres Nagios/Icinga-Monitoring-Systems kennenlernen und effektiv nutzen.

Lösung

Wir zeigen Ihnen im Folgenden wichtige Funktionen der Weboberfläche auf, über die die Bedienung des Monitoring-Systems im Alltag stattfindet. Sie können dabei unterschiedliche Benutzer mit differenzierten Rechten wie in Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche* beschrieben einrichten. Benutzer haben also unter Umständen nicht auf alle der im Folgenden beschriebenen Funktionen und Inhalte

Zugriff. Wir werden Ihnen nachfolgend Abbildungen der Standard-Icinga-Oberfläche, genannt Icinga-Classic, zeigen. Auf Unterschiede zur Nagios-Oberfläche werden wir Sie dabei entsprechend hinweisen.

Bevor wir jedoch auf die Elemente und Ansichten der GUI eingehen, sollten Sie die von GUI genutzten Status kennen. Neben den von den Plugins zurückgegebenen Werten (siehe Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*) verwendet das Monitoring-System weitere Status:

- PENDING

  Dieser Zustand sollte nur auftreten, bevor der Test zum ersten Mal durchgeführt wird. Er besagt, dass für den geplanten Test noch kein Ergebnis vorliegt.
- UNREACHABLE

  Sofern Sie Eltern-Kind-Beziehungen für die Maschinen eingepflegt haben, wird Nagios/Icinga beim Ausfall einer Maschine anhand dieser Beziehung überprüfen, ob die betreffende Maschine anderen Maschinen übergeordnet ist. Wenn dies der Fall ist, wird es die untergeordneten Maschinen mit dem Status UNREACHABLE versehen und gar nicht erst versuchen, sie weiter zu prüfen. Dies ist die sogenannte Erreichbarkeitslogik.
- ACKNOWLEDGED

  Sie können über die Oberfläche sowohl bei Diensten als auch bei Maschinen eine Markierung setzen, um weitere Benachrichtigungen zu unterdrücken und somit anderen Benutzern anzuzeigen, dass das Problem bekannt ist. Dieser Status heißt `ACKNOWLEDGED` (aus dem englischen *zur Kenntnis genommen*). Wie Sie den Status setzen, erläutern wir in Rezept 3.13, *Kommentare, Acknowledgments und Downtimes nutzen* .
- HANDLED

  Wenn sich bei Nagios/Icinga eine Maschine im Status `DOWN` befindet, geht es davon aus, dass die Dienste auf dieser Maschine nicht erreichbar sind. Um Ihre Aufmerksamkeit auf dieses eigentliche Problem zu lenken, werden solche Dienste eben in einem ganz eigenen Status angezeigt.
- FLAPPING

  Sofern Sie die Erkennung von häufigen Zustandswechseln verwenden, erkennt Nagios/Icinga ein Hin-und-her-Wechseln des Zustands und markiert diese Prüfungen als `FLAPPING` (siehe hierzu Rezept 3.12, *Den Flattern-Status verstehen und nutzen*).

Die Darstellung und Anordnung dieser Status in der Oberfläche können Sie über den Aufruf vorgegebener Ansichten steuern, die wir Ihnen jetzt vorstellen werden.

Die obere Statuszeile

Auf allen Seiten zeigt Icinga Ihnen am oberen Rand eine Statuszeile, die vertikal in zwei Hälften getrennt ist. Die linke Hälfte zeigt Ihnen eine Zusammenfassung der Status für die von Ihnen überwachten Maschinen und Dienste.

Nagios-Statuszeile: Beachten Sie, dass Nagios die Statuszeile im Gegensatz zu Icinga nicht auf allen Seiten in dieser Form anzeigt. Wählen Sie im Zweifel die taktische Übersicht (englisch `Tactical Overview`) im Menü, um diese Statuszeile zu sehen.

Dabei wird wie in Abbildung 3-1 gezeigt oben für Maschinen und unten für Dienste die Anzahl je Status sowie die Summe angegeben. Die Angabe für die Status ist dabei dreigeteilt, so dass ersichtlich ist, welche Maschinen oder Dienste dabei keinen Status beziehungsweise welchen der beiden oben erläuterten Status `ACKNOWLEDGED` und `HANDLED` sie haben.

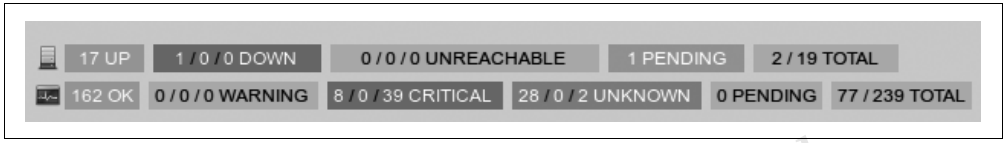


Abbildung 3-1: Linker Teil der oberen Statuszeile der Icinga-GUI

Die dargestellten Zahlen und Status sind dabei verlinkt. Sie können also auf eine Zahl oder einen Status klicken und es werden dann entsprechend die Maschinen- oder Dienst-Details genau der betroffenen Maschinen beziehungsweise Dienste angezeigt. Dies ist ein sehr guter Weg, um schnell Details zu Problemen zu erhalten, die von Ihrem Monitoring-System festgestellt wurden. Allerdings werden die Status nach Maschinen (obere Zeile) beziehungsweise Diensten (untere Zeile) getrennt.

In der rechten Hälfte der Statuszeile sind bei Icinga (im Gegensatz zu Nagios) noch einmal die Summen der aktiven Maschinen und Dienste angeführt, und zusätzlich die Anzahl der passiven und deaktivierten. Sowohl für die Maschinen- als auch die Dienstprüfungen werden Ihnen hier weiter die minimalen, maximalen und durchschnittlichen Durchführungszeiten angezeigt. Abbildung 3-2 zeigt Ihnen diesen Teil beispielhaft.

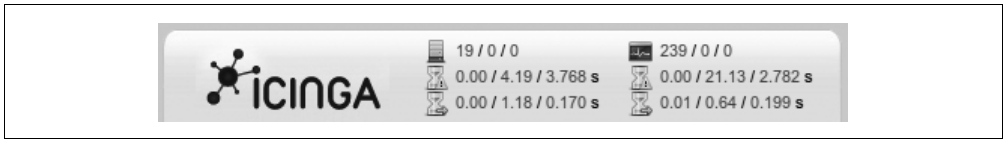


Abbildung 3-2: Rechte Seite der oberen Statuszeile der Icinga-GUI

Performance-Indikator: Nutzen Sie die Angabe der Durchführungszeiten, um nach Änderungen an der Konfiguration mögliche Probleme zu identifizieren.

Lange Prüfzeiten wie die hier gezeigten 21 Sekunden können beispielsweise bei der Durchführung von Prüfungen über SSH (siehe Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*) entstehen.

Das Menü

An der linken Seite finden Sie das Menü, das Ihnen Abbildung 3-3 zeigt. Hier können Sie nach Maschinen und Diensten suchen sowie die verschiedenen Ansichten aufrufen.

Nagios-Menü: Das Menü ist bei Nagios nicht so strukturiert wie bei Icinga. Die hier im Icinga-Menü angeführten Kategorien `Status` und `Problems` sowie die Suche sind bei Nagios alle in einer Rubrik `Monitoring` zusammengefasst. Funktional gesehen stehen dahinter aber vergleichbare Seiten, so dass Sie den nachfolgenden Ausführungen auch mit einem Nagios-System folgen können.

Abbildung 3-3: Menü der Icinga-GUI

Die Ihnen hier angebotene Suche ist eine sehr hilfreiche Funktion, da nicht nur mittels des vollständigen Namens sondern auch mittels entsprechender Bestandteile nach Maschinen und Diensten gesucht werden kann,.

Suchen = Filtern: Nutzen Sie die Suche auch als eine schnelle Filter-Funktion. Geben Sie hier einen Teil eines Maschinennamens ein, um alle entsprechenden Geräte anzeigen zu lassen. Bei der Icinga-Suche werden Ihnen auch Dienste und Gruppen angezeigt. Außerdem können Sie hier auch reguläre Ausdrücke verwenden.

Die unterschiedlichen Ansichten und Hinweise zur Nutzung werden wir in den nachfolgenden Unterabschnitten erläutern.

Die taktische Übersicht (Tactical Overview)

Die taktische Übersicht wie in Abbildung 3-4 bietet Ihnen eine detailliertere Ansicht der in der Statuszeile zusammengefassten Zähler. Hier sehen Sie zusätzlich, welche Prüfungen als FLAPPING erkannt wurden und welche Sie temporär deaktiviert haben.

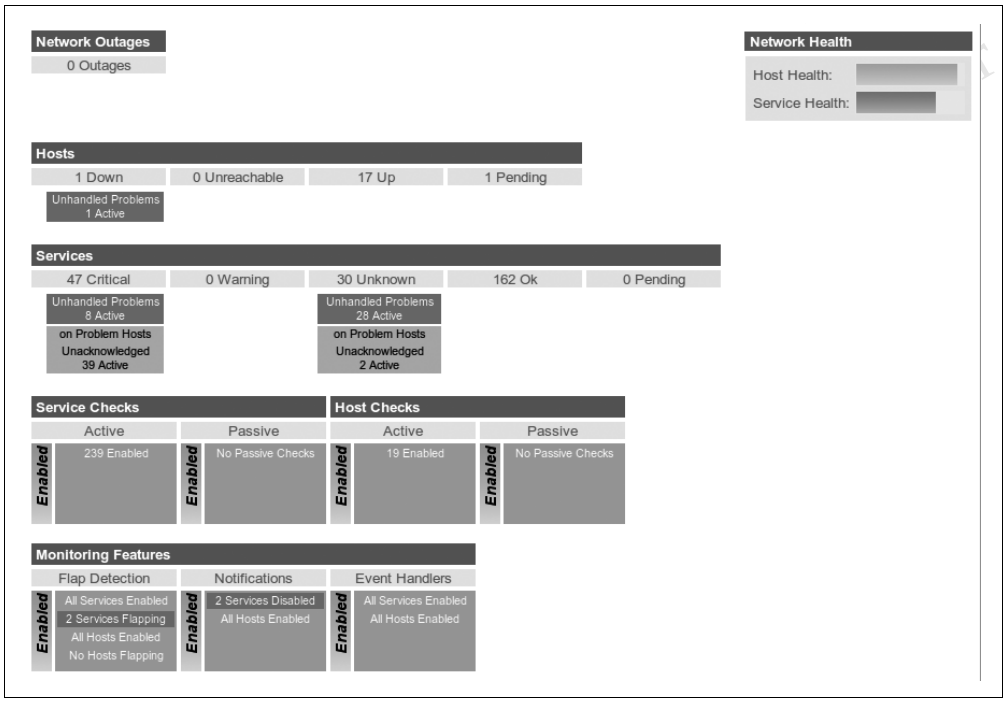


Abbildung 3-4: Die taktische Übersicht der Icinga-GUI

Die taktische Übersicht ist damit eine gute Einstiegsseite. Ansonsten haben Sie mit der oberen Statuszeile wichtige Änderungen dieser Daten immer Blick. Auch in dieser Ansicht sind die dargestellten Zähler und Status verlinkt, so dass Sie schnell zu den entsprechenden Detail-Übersichten (siehe folgender Teil) springen können.

Passive Checks: In Icinga werden passive Checks auch als solche dargestellt. In Nagios werden diese als `active check disabled` aufgeführt.

Detail-Übersichten zu den Maschinen (Host) und Diensten (Service)

Über die Menüpunkte `Host-Detail` und `Service-Detail` gelangen Sie auf Übersichtsseiten aller Maschinen beziehungsweise aller Maschinen und ihrer Dienste. Abbildung 3-5 zeigt Ihnen einen Ausschnitt dieser Ansicht für die ersten fünf Maschinen. Hier sehen Sie neben dem Status auch die Kurzausgabe des jeweiligen Plugins beziehungsweise ersatzweise eine Kurzinformation Ihres Monitoring-Systems wie hier bei der Demonstrations-Maschine.

Über einen Klick auf den Namen einer Maschine in der Detail-Übersicht gelangen Sie von hier aus zu der Detail-Ansicht der Maschine, aber ohne die entsprechenden Dienste. Das rechte der beiden in Abbildung 3-5 gezeigten Icons ist ein Link auf die Dienst-Detail-Seite aller Dienste dieser Maschine, die wir Ihnen direkt im Anschluss beschreiben. Beim linken Icon handelt es sich um den im Rahmen der Installation (siehe Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*) integrierten Link auf die entsprechende Seite von PNP4Nagios mit den entsprechenden Graphen, sofern diese vorhanden sind.

Host Status Details For All Host Groups

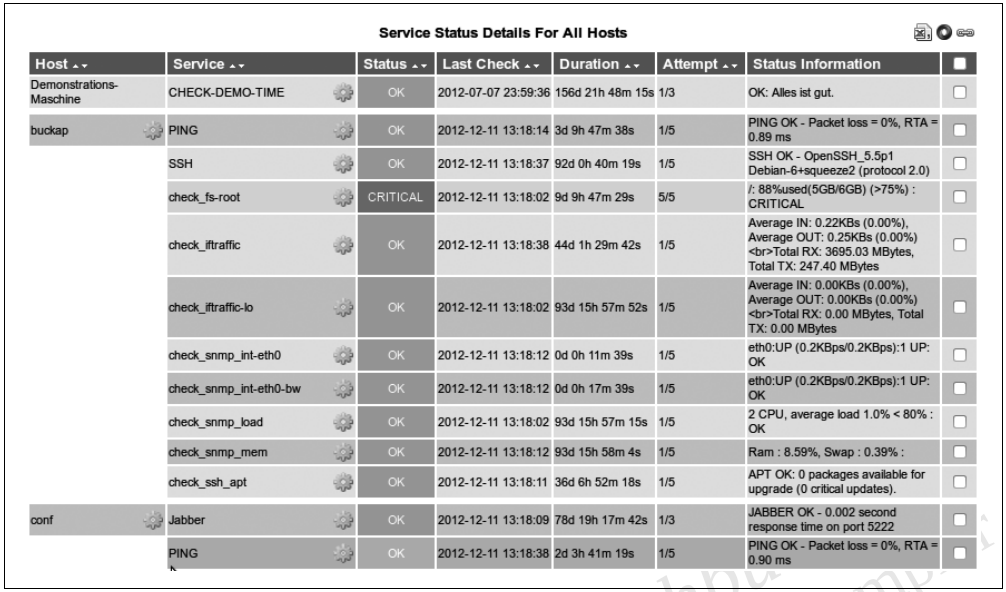
Host	Status	Last Check	Duration	Attempt	Status Information	
Demonstrations-Maschine	PENDING	N/A	156d 21h 45m 41s	1/1	Host check scheduled for Tue Dec 11 13:19:48 CET 2012	☐
buckap	UP	2012-12-11 13:11:17	270d 16h 51m 33s	1/2	PING OK - Packet loss = 0%, RTA = 0.32 ms	☐
conf	UP	2012-12-11 13:14:07	95d 2h 10m 38s	1/2	PING OK - Packet loss = 0%, RTA = 0.26 ms	☐
db	UP	2012-12-11 13:13:17	95d 2h 10m 38s	1/2	PING OK - Packet loss = 0%, RTA = 10.36 ms	☐
docu	UP	2012-12-11 13:13:27	270d 16h 51m 30s	1/2	PING OK - Packet loss = 0%, RTA = 0.25 ms	☐

Abbildung 3-5: Die Übersicht Host-Detail der Icinga-GUI

Die Dienst-Detail-Seite ermöglicht Ihnen den Überblick über Maschinen und jeweils alle dazugehörigen Dienste inklusive der Plugin-Kurzausgaben. Abbildung 3-6 zeigt Ihnen ein Beispiel für einen solchen Ausschnitt. Analog zu den Verlinkungen der Maschinennamen führen Sie die Dienstnamen zu den Detailseiten des jeweiligen Dienstes, auf der Sie dann gegebenenfalls zusätzlich die lange Ausgabe des Plugins finden (siehe im Nachfolgenden).

Übersichtsseiten bei Nagios: Bei Nagios erhalten Sie die Übersichtsseiten für die Einträge `Hosts` beziehungsweise `Services`.

Beide Übersichtsseiten ermöglichen Ihnen, Befehle zu einer Auswahl von Maschinen beziehungsweise Diensten abzusetzen. Hierzu setzen Sie rechts einen Haken an alle entsprechenden Maschinen/Dienste und wählen im Dropdown-Feld (beschriftet mit `Select`



Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
Demonstrations-Maschine	CHECK-DEMO-TIME	OK	2012-07-07 23:59:36	156d 21h 48m 15s	1/3	OK: Alles ist gut.
buckap	PING	OK	2012-12-11 13:18:14	3d 9h 47m 38s	1/5	PING OK - Packet loss = 0%, RTA = 0.89 ms
	SSH	OK	2012-12-11 13:18:37	92d 0h 40m 19s	1/5	SSH OK - OpenSSH_5.5p1 Debian-6+squeeze2 (protocol 2.0)
	check_fs-root	CRITICAL	2012-12-11 13:18:02	9d 9h 47m 29s	5/5	/: 88%used(5GB/6GB) (>75%) : CRITICAL
	check_iftraffic	OK	2012-12-11 13:18:38	44d 1h 29m 42s	1/5	Average IN: 0.22KBs (0.00%), Average OUT: 0.25KBs (0.00%) Total RX: 3695.03 MBytes, Total TX: 247.40 MBytes
	check_iftraffic-lo	OK	2012-12-11 13:18:02	93d 15h 57m 52s	1/5	Average IN: 0.00KBs (0.00%), Average OUT: 0.00KBs (0.00%) Total RX: 0.00 MBytes, Total TX: 0.00 MBytes
	check_snmp_int-eth0	OK	2012-12-11 13:18:12	0d 0h 11m 39s	1/5	eth0:UP (0.2KBps/0.2KBps):1 UP: OK
	check_snmp_int-eth0-bw	OK	2012-12-11 13:18:12	0d 0h 17m 39s	1/5	eth0:UP (0.2KBps/0.2KBps):1 UP: OK
	check_snmp_load	OK	2012-12-11 13:18:02	93d 15h 57m 15s	1/5	2 CPU, average load 1.0% < 80% : OK
	check_snmp_mem	OK	2012-12-11 13:18:12	93d 15h 58m 4s	1/5	Ram : 8.59%, Swap : 0.39% :
	check_ssh_apt	OK	2012-12-11 13:18:11	36d 6h 52m 18s	1/5	APT OK: 0 packages available for upgrade (0 critical updates).
conf	Jabber	OK	2012-12-11 13:18:09	78d 19h 17m 42s	1/3	JABBER OK - 0.002 second response time on port 5222
	PING	OK	2012-12-11 13:18:38	2d 3h 41m 19s	1/5	PING OK - Packet loss = 0%, RTA = 0.90 ms

Abbildung 3-6: Die Übersicht Service-Detail der Icinga-GUI

`Command`) über der Tabelle (in den Screenshots hier nicht zu sehen) das gewünschten Kommando. Auf die unterschiedlichen Befehle werden wir in der Diskussion später in diesem Abschnitt noch eingehen.

Command Dropdown: Die Dropdown-Felder m*ultiple commands* und *command dropdown* stehen Ihnen nur bei Icinga zur Verfügung. Nagios besitzt diese Features nicht.

Details zu Maschinen (Host) und Diensten (Service)

Wenn Sie wirklich alle verfügbaren Daten zu einer Maschine oder einem Dienst einsehen möchten, benötigen Sie, im Gegensatz zu den bisherigen Detail-Übersichten, die entsprechende Detail-Ansicht. Zu dieser gelangen Sie, wenn Sie auf den Namen einer Maschine oder eines Dienstes klicken.

Abbildung 3-7 zeigt Ihnen zunächst die Detail-Ansicht einer Maschine. Oben sehen Sie links zunächst ein Menü mit relevanten Sprungzielen, unter anderem zu der Dienst-Detail-Übersicht aller Dienste dieser Maschine. Auch mögliche Abhängigkeiten (Eltern-Beziehungen, siehe Rezept 4.1, *Maschinen einbinden (host)* und Maschinen-Abhängigkeiten siehe Rezept 4.9, *Abbilden von Abhängigkeiten (host- und service-dependencies)*) sowie Zugehörigkeiten zu Maschinengruppen (siehe Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*) werden hier angezeigt.

Abhängigkeiten: Beachten Sie, dass Ihnen die Abhängigkeiten von Maschinen (Host Dependencies) und Diensten (Service Dependencies) an dieser Stelle nur unter Icinga angezeigt werden.

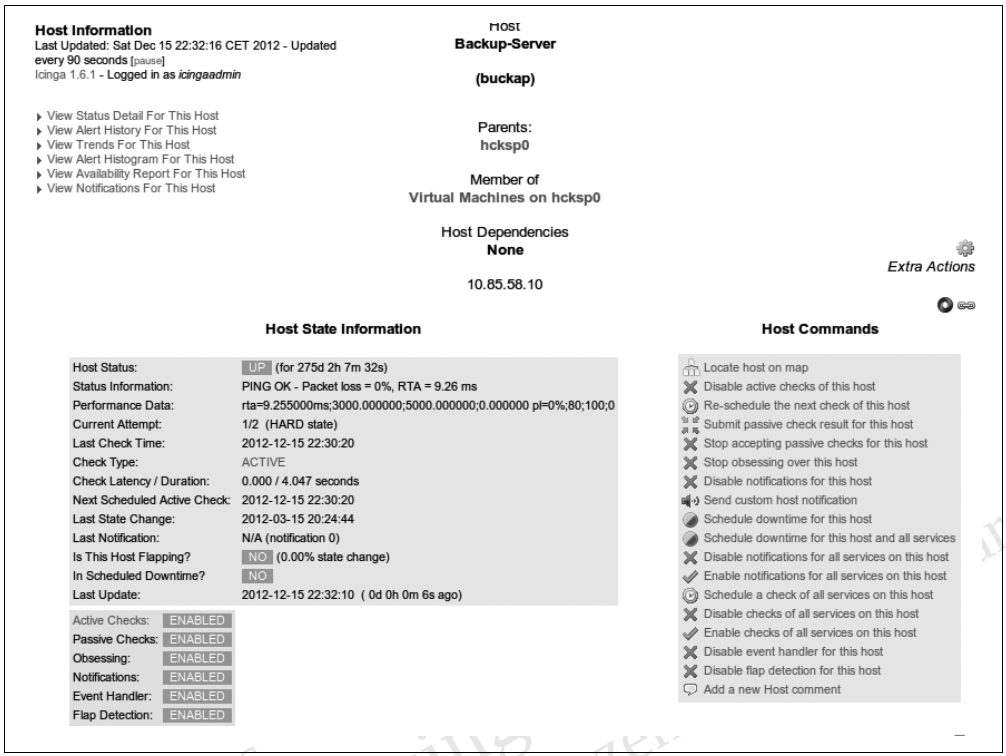


Abbildung 3-7: Detail-Ansicht einer Maschine in der Icinga-GUI

In dem Kasten `Host State Information` zeigt Ihnen Nagios/Icinga nun alle Details zum Zustand der Maschine und zu der zu dessen Feststellung durchgeführten Prüfung an. Neben den Ergebnissen gehören hierzu auch Details wie Uhrzeit der letzten und nächsten Durchführung, der letzten Benachrichtigung, Anzahl der Prüfungen im aktuellen Status und verschiedene andere mehr. Am rechten Rand haben Sie dann noch eine Liste aller Befehle, die Sie über die Weboberfläche für diese Maschine absetzen können.

Vergleichbar zu der Detail-Ansicht für Maschinen ist die Detail-Ansicht für Dienste. Abbildung 3-8 zeigt Ihnen ein Beispiel für einen Dienst in dieser Ansicht. Der Aufbau ist dem der Detail-Ansicht für Maschinen von eben sehr ähnlich, nur dass die Informationen und Befehle sich jetzt auf den Dienst beziehen. Eine Eltern-Beziehung gibt es bei Diensten nicht (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*), aber von Ihnen eingepflegte Dienst-Gruppen (siehe Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*) und -Abhängigkeiten (siehe Rezept 4.9, *Abbilden von Abhängigkeiten (host- und service-dependencies)*) werden hier angezeigt.

Die Detail-Ansicht ist daher typischerweise nur bei bestehenden Problemen relevant. Von hier können Sie die Maschine beziehungsweise den Dienst dann auch in den Sonderzustand `ACKNOWLEDGED` setzen (mit dem Befehle `acknowledge host/service problem`, siehe auch Diskussion), so dass keine Meldungen mehr zu dem Problem versendet werden.

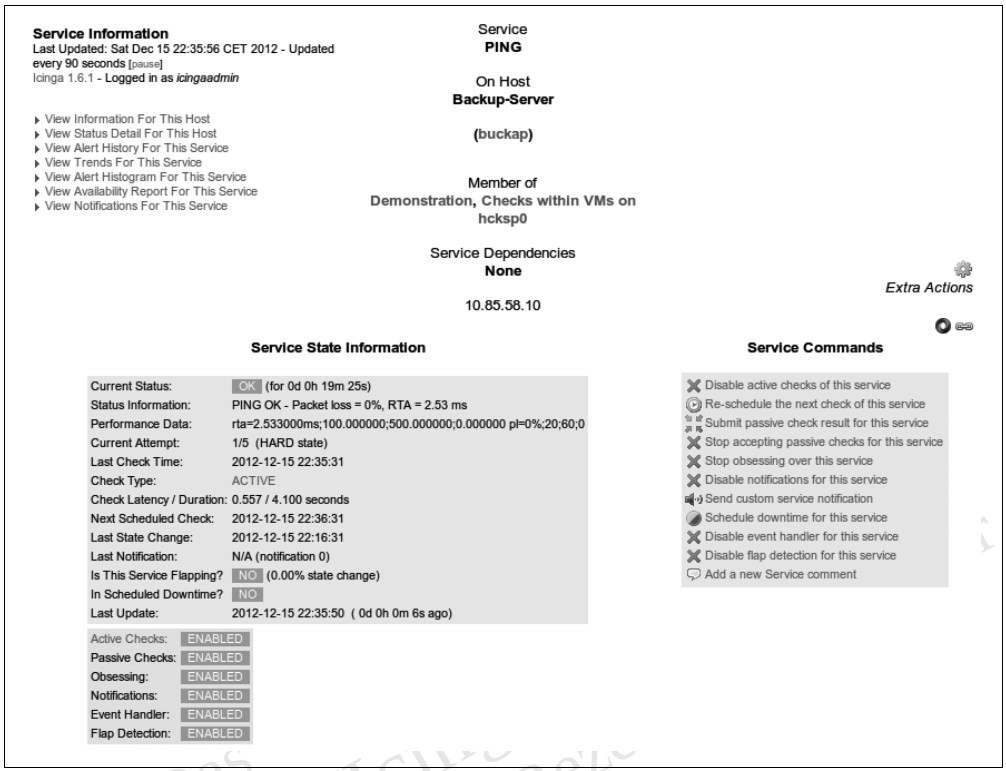


Abbildung 3-8: Detail-Ansicht eines Dienstes in der Icinga-GUI

Übersichten zu Maschinengruppen (hostgroup)

Sofern Sie Ihre Maschinen wie in Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)* beschrieben gruppieren, stehen Ihnen weitere Übersichten für diese Gruppen zur Verfügung. Zunächst die Maschinengruppen-Übersicht genannte Ansicht, in der Sie für jede Maschine der Gruppe eine Zusammenfassung ihrer Dienste nach Status erhalten. Abbildung 3-9 zeigt Ihnen eine Gruppe in dieser Ansicht. Die Icons an jeder Maschine erlauben Ihnen Zugriffe auf

- die Detail-Ansicht der Maschine,
- den Graphen der Prüfung der Maschine (sofern vorhanden),
- die Status-Übersicht aller Dienste der Maschine, sowie
- die Status-Karte mit Markierung der Maschine (zur Status-Karte siehe den entsprechenden Abschnitt weiter unten).

Wie bei den anderen Ansichten auch sind Namen und Zähler mit den entsprechenden Ansichten verknüpft, so dass Sie über diese über einen Klick bequem zu detaillierteren Ansichten gelangen.

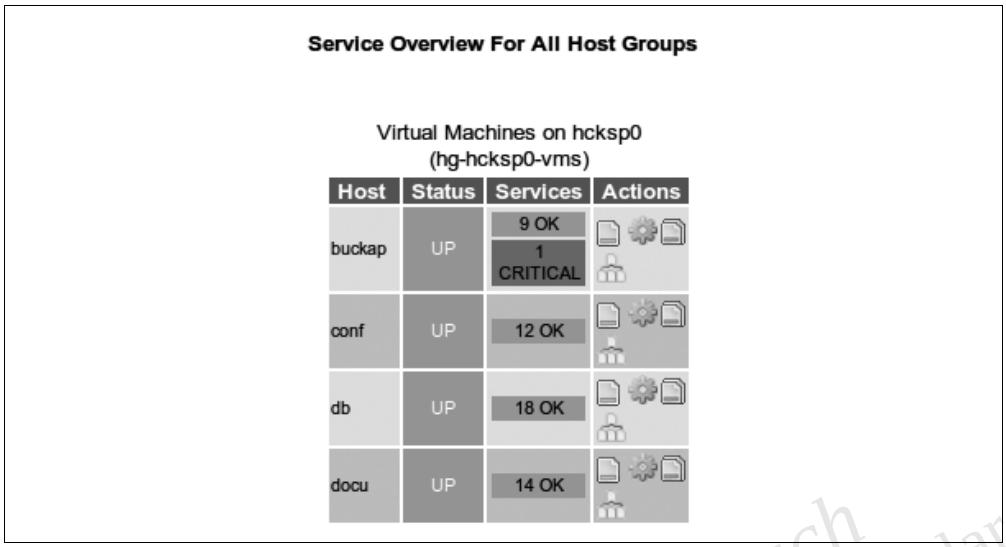


Abbildung 3-9: Die Übersicht über Maschinengruppen in der Icinga-GUI

Daneben gibt es eine Status-Zusammenfassung (`Status Summary`) genannte Ansicht für Maschinengruppen, in denen die zu jeder Gruppe nicht mehr die einzelnen Maschinen, sondern nur noch die Summen der Maschinen und Dienste dieser Gruppe angezeigt werden. Abbildung 3-10 zeigt Ihnen ein Beispiel für zwei Gruppen in dieser Ansicht. Auch hier sind die Zähler wieder verlinkt, so dass Sie zu den Detail-Übersichten der entsprechenden Maschinen oder Dienste gelangen können.

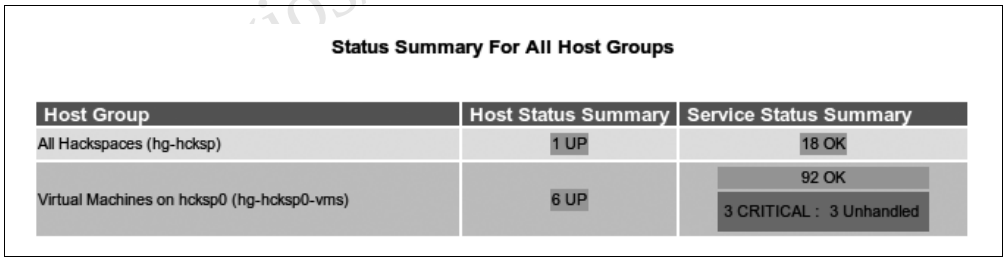


Abbildung 3-10: Ansicht Maschinengruppen-Zusammenfassung in der Icinga-GUI

Neben der Möglichkeit, Dienste anstatt einzelnen Maschinen gleich ganzen Maschinengruppen zuzuweisen (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*), können Sie Maschinengruppen auch zur Strukturierung der Oberfläche verwenden. Allerdings kann dies für Sie auch problematisch werden, falls Sie beides parallel tun möchten. Bei sehr vielen Gruppen verlieren die Übersichtsseiten dann nämlich schnell an Übersichtlichkeit.

Grid-Ansichten: Zusätzlich gibt es die sogenannte Grid-Ansicht für die Gruppen, die anstelle der summierten Anzahl an Diensten die Namen der jeweiligen Dienste aufführt. Die Status der Dienste werden Ihnen dabei durch farbige Hinterlegung angezeigt. Abbildung 3-11 zeigt Ihnen diese Ansicht an einem Beispiel für Maschinengruppen unter Nagios. Dort erreichen Sie diese direkt über das Hauptmenü. Dagegen ist die Grid-Ansicht unter Icinga über das Mini-Menü links oben in den Gruppenansichten erreichbar.

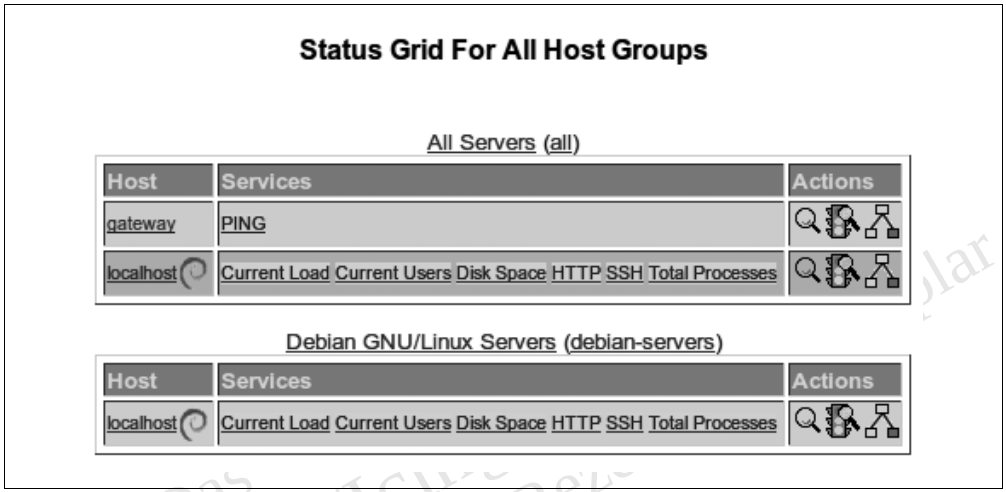


Abbildung 3-11: Beispiel der Grid-Ansicht aus Nagios für Maschinengruppen

Übersichten zu Dienstgruppen (servicegroup)

Vergleichbar mit den Übersichten zu Maschinengruppen gibt es Ansichten für die Dienstgruppen (siehe Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*). Die Dienstgruppen-Übersicht zeigt Ihnen also die Ansicht der Maschinengruppen-Übersicht, aber eben nur mit den Diensten der Dienstgruppe und den entsprechenden Maschinen. Abbildung 3-12 zeigt Ihnen ein Beispiel für zwei Dienstgruppen in dieser Ansicht.

Analog zur Maschinengruppen-Zusammenfassung ist auch die Darstellung der Dienstgruppen-Zusammenfassung, aber eben wieder mit der Beschränkung auf Dienste der Dienstgruppe und die entsprechenden Maschinen. Abbildung 3-13 zeigt Ihnen ein Beispiel für zwei Dienstgruppen in dieser Ansicht.

Grid-Ansichten: Analog zu der für Maschinen verfügbaren Grid-Ansicht gibt es auch bei den Diensten eine entsprechende Ansicht (siehe oben).

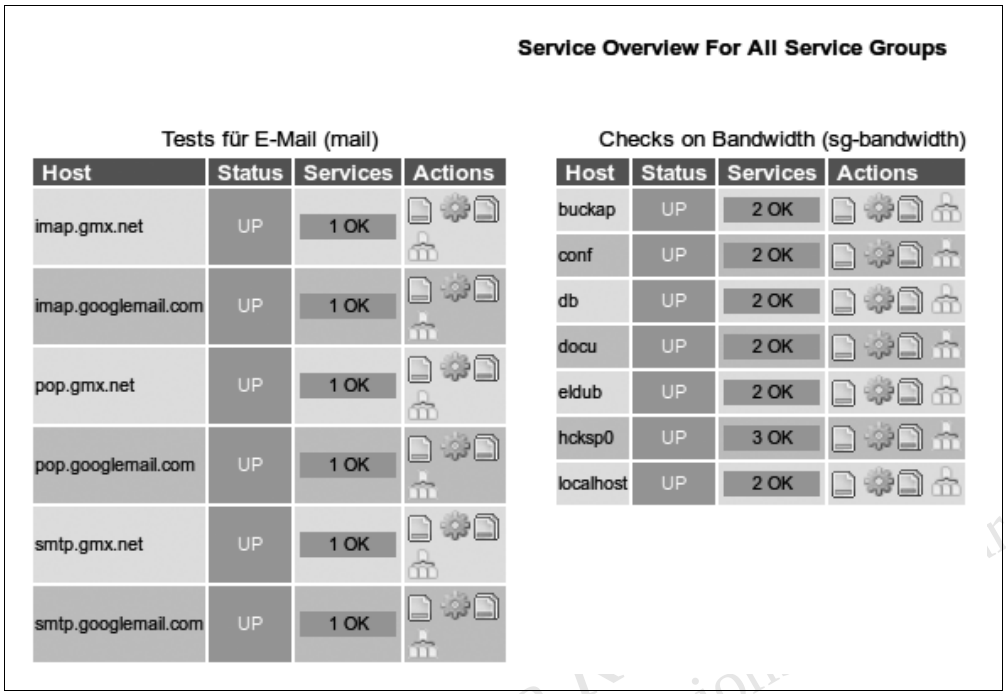


Abbildung 3-12: Ansicht Dienstgruppen-Übersicht in der Icinga-GUI

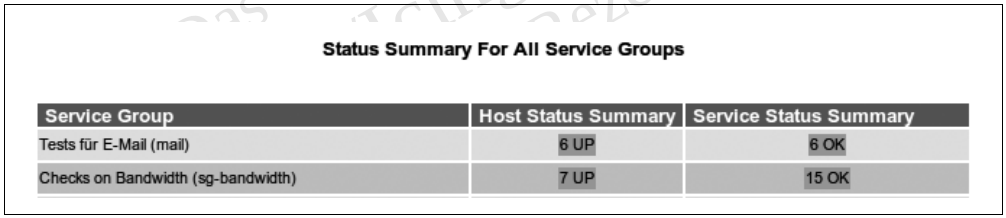


Abbildung 3-13: Ansicht Dienstgruppen-Zusammenfassung in der Icinga-GUI

Dienstgruppen können Sie damit sehr gut nutzen, um sich Ansichten mit einer Auswahl an Diensten zusammenzustellen. In den Beispielen hier haben wir nach technischer Gemeinsamkeit gruppiert, aber in der Praxis möchten Sie hier vielleicht Gruppen nach Zuständigkeit, Support-Verträgen oder ganz anderen Aspekten gruppieren.

Problem-Ansichten

Da Sie bei den Status-Ansichten immer nach einem Status filtern, bietet Ihnen Nagios/Icinga in der Rubrik `Problems` zusammenfassende Listen aller Maschinen beziehungsweise Dienste mit einem Problem-Status oder wahlweise eine zusammengefasste Liste mit allen Problemen. Die gezeigten Informationen entsprechen dabei den eben bereits beschriebenen Detail-Übersichten, aber eben beispielsweise bei Diensten mit den

Status UNKNOWN, WARNING oder CRITICAL. Bei den jeweils zusätzlich vorhandenen Ansichten mit dem Zusatz Unhandeled werden Ihnen dabei nur die Maschinen beziehungsweise Dienste angezeigt, die Sie nicht über die Oberfläche mit dem zusätzlichen Status ACKNOWLEDGED versehen haben (siehe hierzu Rezept 3.13, *Kommentare, Acknowledgments und Downtimes nutzen*). Die Ansicht Network Outages zeigt Ihnen Probleme, die Nagios/Icinga aufgrund der Erreichbarkeitslogik festgestellt hat (siehe oben).

Diese Problem-Ansichten bieten Ihnen entsprechend eine gute Möglichkeit, um sich schnell eine Übersicht über alle bestehenden Probleme zu verschaffen. Dies wird typischerweise häufig Ihr primäres Anliegen sein, wenn Sie nicht gerade nur spezifische Informationen einholen möchten.

Status-Karte (Status Map)

Die Status-Karte ordnet Maschinen anhand ihrer Eltern-Beziehung (siehe Rezept 4.1, *Maschinen einbinden (host)*) graphisch an. Abbildung 3-14 zeigt Ihnen die Ansicht bei Aufruf des entsprechenden Menüs, wobei Sie die unter hcksp0 angeordneten Maschinen buckap, conf, db und so weiter bemerken werden. Bei umfangreichen, verschachtelten Netzwerken offenbart sich die Stärke dieser Ansicht, da Sie hier den Pfad des Ausfalls in der Netzwerktopologie nachvollziehen können.

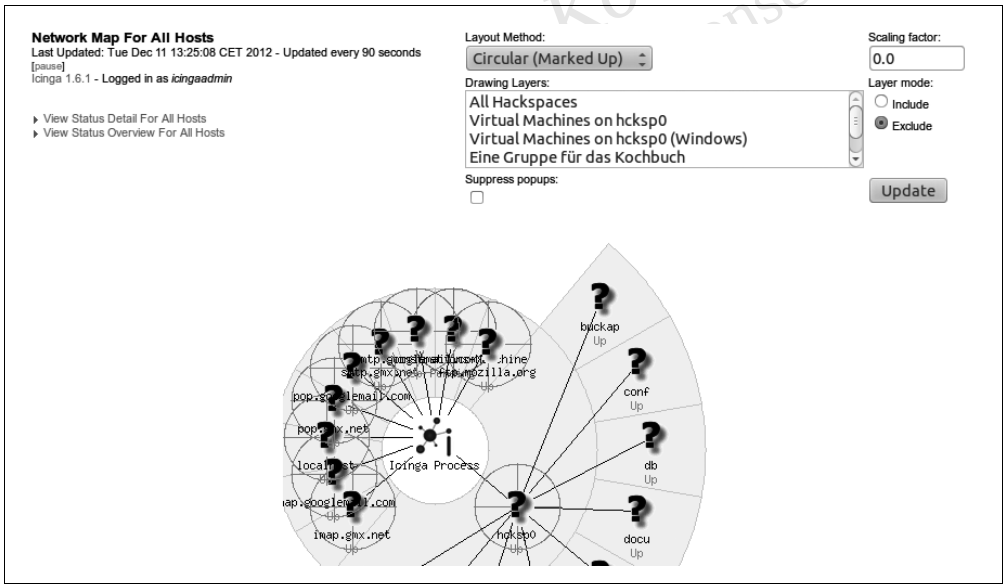


Abbildung 3-14: Status-Karten-Ansicht in der Icinga-GUI

Nähere Informationen zu den Maschinen können Sie dabei über den sogenannten Tooltip abrufen, indem Sie mit der Maus über der entsprechenden Maschine verweilen. Links haben Sie ein Menü, über das Sie zu der ursprünglichen Ansicht zurückgelangen oder

Detail-Ansichten der ausgewählten Maschine aufrufen können. Über die Optionen rechts oben können Sie außerdem

- die Darstellung vergrößern oder verkleinern,
- auf Gerätegruppen beschränken oder diese ausschließen und
- andere Darstellungsarten wählen.

Änderungen werden jeweils erst nach dem Klick auf den Button `Update` wirksam.

Ansichten: Wenn Sie ein sehr großes Netzwerk verwalten, schauen Sie sich die Ansicht `Depth Layers` und `Balanced Tree` an. Diese können Ihnen durchaus dabei helfen, einen besseren Überblick zu erlangen.

Berichte (Reporting)

Nagios/Icinga beinhalten ein Berichtswesen, mit dem Sie sich Daten über Zeiträume und gegebenenfalls aggregiert, eben als Berichte, anzeigen lassen können. Wir beschreiben Ihnen im Folgenden kurz die Ihnen hier zur Verfügung stehenden Optionen:

- Trends

  Hiermit lassen Sie sich eine einfache Übersicht über den Gesamtstatus von Maschinen oder Diensten anzeigen. Die betreffenden Berichte beziehen sich auf einzelne Maschinen oder Dienste.

- Availability (Verfügbarkeit)

  Mit dieser Option können Sie sich Statusinformationen zu Maschinen, Diensten oder entsprechenden Gruppen anzeigen lassen. Sofern mehrere Maschinen oder Dienste ausgewählt sind, erhalten Sie dabei eine kurze, tabellarische Übersicht. Wenn Sie eine einzelne Maschine oder einen einzelnen Dienst auswählen, erhalten Sie hingegen einen detaillierten Bericht, der bei Maschinen auch all Ihre Dienste umfasst.

- Alert Histogram

  Mit dieser Option können Sie sich zu einer Maschine oder einem Dienst eine graphische Darstellung der Status-Entwicklung über einen von Ihnen ausgewählten Zeitrahmen erstellen lassen.

- Alert History

  Hier werden alle Warnmeldungen angezeigt, die Nagios/Icinga aufgrund von Statusänderungen generiert hat (inklusive der dazugehörigen Ausgabe des Plugins). Sie können diese nach Typ und Status filtern.

- Alert Summary

  Mit dieser Berichtsoption können Sie detaillierte Berichte über Statusänderungen erstellen oder zwischen entsprechenden vorgefertigten Berichten wählen. Dabei stehen Ihnen beispielsweise Berichte zu den häufigsten Alarm-Verursachern und chronologisch sortierte Alarmmeldungen eines bestimmten Zeitraums zur Verfügung, die Sie nach Bedarf auf entsprechende Maschinen, Dienste oder Gruppen einschränken können.

- Notifications

  Hier können Sie alle versandten Benachrichtigungen einsehen. Die zunächst angezeigte Liste aller Benachrichtigungen können Sie mit Hilfe der Auswahl rechts oben nach Typen filtern.

- Event Log

  Hier werden alle Aktivitäten angezeigt, also sowohl die auslösenden Alarme als auch die daraufhin versandten Benachrichtigungen. Auch hier können Sie über die Optionen rechts oben wieder Filter setzen. Diese Ansicht ist besonders praktisch, weil Sie über das rechts oben vorhandene Suchfeld nach Namen suchen und so gegebenenfalls schnell filtern können.

Suche und Filter: Die Suche und Filter wurden exklusiv in Icinga implementiert. Nagios besitzt diese Funktionalität nicht.

Die Berichtsoptionen bieten Ihnen Abrufoptionen hinsichtlich der verschiedenen von Nagios/Icinga verwalteten Informationen. Im Rahmen Ihrer täglichen Arbeit werden diese bei Fragestellungen wie »Welche Benachrichtigungen wurden letzte Woche an Person xyz versandt?« hilfreich sein. Ansonsten bieten Sie eine vorzügliche Grundlage, um Berichte für Kunden beziehungsweise Abteilungen und Ähnliches zu erstellen.

Jasper-Server: Als Erweiterung lässt sich ein Jasper-Server an Nagios/Icinga anbinden, mit dem Sie dann Berichte erstellen können. Jasper nutzt das NDO/IDO-Backend als Datenquelle.

Systeminformationen

Unter der Rubrik *System* bietet Ihnen Nagios/Icinga diverse Ansichten zu den internen Abläufen des Monitoring-Systems. Die Ihnen hier zur Verfügung stehenden Optionen sind folgende:

- Kommentare (`Comments`)

  Eine Ansicht, die Ihnen alle aktiven Kommentare (siehe hierzu Rezept 3.13, *Kommentare, Acknowledgments und Downtimes nutzen*) aufgeteilt nach Maschinen und Diensten anzeigt. Eine sehr nützliche Ansicht, die Ihnen kommentierte »Baustellen« auf einen Blick auflistet.

Massenlöschung: Unter Icinga können Sie auch mehrere Kommentare oder Ausfallzeiten markieren und dann alle gleichzeitig löschen.

- Geplante Ausfallzeiten (`Downtime`)

  In dieser Ansicht werden Ihnen alle geplanten Ausfallzeiten angezeigt, wieder aufgeteilt nach Maschinen und Diensten.

- Prozess-Informationen (`Process Info`)

  Diese Ansicht zeigt Ihnen detaillierte Informationen zu Version und Laufzeit des Prozesses inklusive der Werte wichtiger Optionen.
- Performance-Informationen (`Performance Info`)

  Einige Informationen zur Performance werden Ihnen wie oben dargestellt in der Statusleiste angezeigt. In dieser Ansicht können Sie hierzu detaillierte Daten abrufen.
- Geplante Prüfungen (Scheduling Queue)

  Hier können Sie sich anschauen, welche Prüfungen als Nächstes durchgeführt werden. Sie haben zusätzlich die Möglichkeit, geplante Prüfungen abzubrechen oder für einen früheren oder späteren Zeitpunkt zu planen.

Diese Ansichten sind bei der täglichen Arbeit normalerweise zunächst nur von begrenztem Nutzen. Aber bei vielen Mitarbeitern, vielen Kommentaren und häufigen Ausfallzeiten einzelner Komponenten werden Sie sie schnell zu schätzen lernen. Bei Performance-Problemen geben Ihnen beispielsweise die Performance-Informationen Hinweise darauf, wo das eigentliche Problem zu suchen ist.

Tooltip: Kommentare und deren Autor werden unter Icinga in der Statusübersicht als Tooltip angezeigt.

Konfigurationsobjekte (Configuration)

Die Kategorie zur Einsicht in die Konfigurationsdateien hat nur eine Option, hinter der sich Auswahldialoge zur Eingrenzung der Ansicht befinden. Es wird Ihnen hier nicht etwa einfach die Konfigurationsdateien selbst angezeigt, sondern die Optionen und Werte, die Nagios/Icinga in verschiedenen Stufen zusammensetzt, also etwa zu einer Maschine nicht nur die direkt in ihrer Definition hinterlegten Optionen, sondern zusätzlich auch alle über Vorlagen vererbte Optionen sowie über diese Maschine referenzierte Dienste. Und bei Diensten können Sie hier einsehen, wie die Makros mit Werten gefüllt werden und den auszuführenden Befehl ergeben.

Diskussion

Typischerweise werden die Übersichten und Problem-Ansichten zunächst die von Ihnen am meisten genutzten Ansichten sein, um die gerade wichtigen Details in Erfahrung zu bringen. Nutzen Sie bei Problemen mit den Konfigurationsdateien die Konfigurationsansicht, um zu sehen, wie Nagios/Icinga die Ersetzungen vornimmt und ob vielleicht dabei irgendwo etwas nicht wie gewünscht funktioniert hat.

In Icinga können Sie mehrere Maschinen oder Dienste auswählen und Kommandos an diese Auswahl absetzen, was einen erheblichen Vorteil gegenüber der Nagios-Oberfläche darstellen kann, die Ihnen diese Möglichkeit nicht bietet. Hier müssen Sie gegebenenfalls

jede Maschine beziehungsweise jeden Dienst nacheinander auswählen und das entsprechende Kommando absetzen.

Wenn Sie eine Möglichkeit wünschen, um eigene Ansichten zusammenstellen zu können (über Filter oder durch die Kombination verschiedener Ansichten), sollten Sie sich die neuere Oberfläche Icinga-Web anschauen (siehe Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*). Diese bietet Ihnen die entsprechenden Möglichkeiten inklusive einer eigenen Benutzer- und Rechteverwaltung und weiteren »Nettigkeiten«. Wir wollen allerdings nicht unerwähnt lassen, dass eine Filterung auch mit den klassischen Oberflächen möglich ist – allerdings müssen Sie die Bedingungen hierzu in den URL kodieren (oder über eine Kommandozeile aufrufen). Dies ist vergleichsweise kompliziert und bedingt eine Speicherung außerhalb der Oberfläche (etwa durch Lesezeichen oder in Scripten). Wir werden auf diese Möglichkeiten deshalb hier nicht weiter eingehen. Wenn Sie auf diese Alternative zurückgreifen müssen, lesen Sie bitte die entsprechenden Hilfe-Seiten (für Icinga *http://docs.icinga.org/latest/de/cgiparams.html*).

Siehe auch

- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zur Einführung in die von Nagios/Icinga verwendete API: Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*
- Rezept zu Kommentaren, Acknowledgments und Downtimes: Rezept 3.13, *Kommentare, Acknowledgments und Downtimes nutzen*
- Rezept zum Flattern-Status: Rezept 3.12, *Den Flattern-Status verstehen und nutzen*
- Rezept zur Remote-Ausführung von lokalen Plugins über SSH: Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*
- Kapitel mit den Installationsrezepten: Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*
- Rezepte zu Maschinen und Dienstdefinitionsn: Rezept 4.1, *Maschinen einbinden (host)* und Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Abbildung von Abhängigkeiten: Rezept 4.9, *Abbilden von Abhängigkeiten (host- und service-dependencies)*
- Rezept zur Zusammenfassung von Maschinen in Maschinengruppen und von Diensten in Dienstegruppen: Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)* und Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*
- Rezept zur Nutzung der Weboberfläche Icinga-Web: Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*
- Hilfeseiten des Icinga-Projektes zu den CGI-Parametern: *http://docs.icinga.org/latest/de/cgiparams.html*

3.4 Nutzung der Weboberfläche Icinga-Web

Problem

Sie möchten die vom Icinga-Projekt neu entwickelte Oberfläche, sogenannt Icinga-Web, verstehen und verwenden.

Lösung

Icinga-Web ist eine von Grund auf neu programmierte Oberfläche für den Zugriff auf das darunter liegende Icinga-Core. Entsprechend stehen hier zunächst die gleichen Daten zur Verfügung wie unter der auf Nagios basierenden Oberfläche Icinga-Classic auch. Allerdings verwendet Icinga-Web eine andere Art des Datenzugriffs: Anstatt auf die herkömmliche Weise auf die entsprechenden Status-Dateien von Icinga-Core zurückzugreifen, benötigt Icinga-Web eine Datenbank-Anbindung über IDOUtils und außerdem auch eine eigene Datenbank, in der es seine Einstellungen speichert. Sie müssen sich dabei nicht zwischen Icinga-Classic und Icinga-Web entscheiden, sondern können beide problemlos auch parallel betreiben. Dabei könnten Sie Icinga-Web auch auf einer anderen Maschine betreiben, um die Last der Oberfläche von der der eigentlichen Monitoring-Maschine zu trennen.

Die Ansichten von Icinga-Web sind konfigurier- und erweiterbar (siehe unten im Diskussionsteil) und wurden mit moderneren Techniken wie AJAX realisiert. Die einzelnen Anzeigekomponenten werden dabei Cronks genannt. Aus anderen Bereichen mag Ihnen der Begriff Widget vertrauter sein. Im Gegensatz zur klassischen Oberfläche haben Sie insbesondere die Möglichkeit, Ansichten in der Oberfläche umzusortieren oder zu filtern (je nach Ansicht). Wir werden in diesem Buch nicht auf die Programmierung eigener Cronks eingehen, aber Sie werden sehen, dass die bereits enthaltenen Möglichkeiten bereits sehr umfangreich und komfortabel sind. So können Sie sich beispielsweise aus den vorhandenen Ansichten eigene Übersichten zusammenstellen.

Um an dieser Stelle Dopplungen mit dem vorhergehenden Rezept (siehe Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*) zu vermeiden, machen Sie sich bitte dort mit

- den von den Oberflächen verwendeten zusätzlichen Status (`acknowledged`, `handled` und `flapping`), wie sie im einleitenden Teil des vorherigen Rezeptes erläutert wurden, und
- der oberen Statuszeile, nebst den rechts dargestellten Performance-Indikatoren

vertraut, sofern Sie ihnen nicht bereits bekannt sind.

Menüleiste

- Ganz oben finden Sie bei Icinga-Web eine Menüleiste für den Zugriff auf übergreifende Funktionen, wie die Benutzer- und Gruppenverwaltung. Diese werden wir im Folgenden kurz beschreiben:
- Über den Menüpunkt Monitoring gelangen Sie aus anderen Ansichten in die Datenanzeige zurück.
- Unter dem Menüpunkt *Admin* können Sie, sofern Ihr Benutzerkonto über die entsprechenden Rechte verfügt, auf die folgenden administrative Optionen zugreifen:
  - Users ermöglicht Ihnen die Verwaltung der Benutzer. Im Gegensatz zur klassischen Oberfläche können Sie bei Icinga-Web direkt über die Oberfläche neue Benutzer hinzufügen und vorhandene Benutzer anpassen. Neben allgemeinen Benutzerdaten wie Name und E-Mail-Adresse können Sie hier auch die Rechte der Benutzer verwalten und insbesondere auch einschränken. In der klassischen Oberfläche ist dies nur über die Zuordnung von mit `http-auth` angelegten Benutzer-Konten als Kontakte in den Konfigurationsdateien möglich (siehe Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*).
  - Groups ermöglicht Ihnen die Administration von Gruppen. Neben den bereits für Sie angelegten Gruppen `appkit_admin`, `appkit_user`, `guest` und `icinga_user` können Sie hier auch eigene Gruppen anlegen und verwalten.
  - Logs ermöglicht Ihnen den Zugriff auf die Protokolldateien von Icinga-Web (`debug` und `icinga-web`).
  - Tasks bietet Ihnen zusätzliche administrative Funktionen, wobei Ihnen hier zunächst nur die Löschung des Zwischenspeichers (`cache`) zur Verfügung steht.
- Ein weiterer Menüpunkt *Help* erlaubt es Ihnen, über Icinga home schnell zu den Seiten des Projektes (*https://www.icinga.org/*) zu springen und über About die Versionsinformationen Ihres Systems abzufragen.

  Neben dem eigentlichen Menü haben sie in der gleichen Zeile auf der rechten Seite dann noch die folgenden Funktionen zur Verfügung:
- Sofern Sie sich in der Monitoring-Ansicht befinden (siehe oben bei Menüpunkt *Monitoring*), haben Sie rechts von der Menüleiste weiter das Eingabefeld für die Suchfunktion (gekennzeichnet durch eine Lupe). Dieses zeigt Ihnen bereits während der Eingabe eine entsprechend gefilterte Liste von Diensten und Maschinen und ist dadurch für einen schnellen, zielgerichteten Zugriff auf Informationen gut geeignet. Sie können dabei das Zeichen * als Platzhalter verwenden.
- Daneben wird Ihnen eine Schaltfläche mit den zu Ihrem Konto hinterlegten Namen angezeigt. Hier können Sie einige Einstellungen zu Ihrem eigenen Konto ändern, wie etwa eine andere Sprache wählen oder das Benutzerprofil zurücksetzen. Wir gehen hier allerdings nur auf die englische Version ein. Die Übersetzungen in der hier von uns betrachteten Version 1.8.1 betreffen auch nur Teile der Oberfläche und die Unterschiede sind demzufolge nicht sehr groß. Darüber hinaus haben Sie in diesem

Benutzermenü noch die Möglichkeit, sich abzumelden. Wählen Sie hierzu das Türsymbol aus und bestätigen Sie die entsprechende Sicherheitsabfrage.

Damit bietet Ihnen das Menü Zugriff auf wichtige Funktionen. Die meisten davon, ausgenommen die Suchfunktion und das Abmelden, dienen administrativen Zwecken, die Sie vermutlich nicht besonders häufig benötigen werden. Gerade aber die Möglichkeit, Benutzer und Gruppen über die Oberfläche selbst zu verwalten, stellen einen deutlichen Vorteil gegenüber der klassischen Oberfläche dar.

Tactical Overview – Ihr Überblick

Im Gegensatz zur klassischen Oberfläche, die nur eine taktische Übersicht bietet, können Sie sich in Icinga-Web verschiedene Ansichten zur taktischen Übersicht anzeigen lassen, die in der Gruppe `Tactical Overview` zusammengefasst sind. Wir werden die drei vorgegebenen Ansichten im Folgenden erläutern.

TO Charts – Die Übersicht: Die gröbste Übersicht bietet Ihnen die Ansicht `TO Charts`. Hier erhalten Sie zwei Tortendiagramme: eines für die Maschinen- und eines für die Dienst-Status. Abbildung 3-15 zeigt Ihnen ein Beispiel dafür, wie diese aussehen.

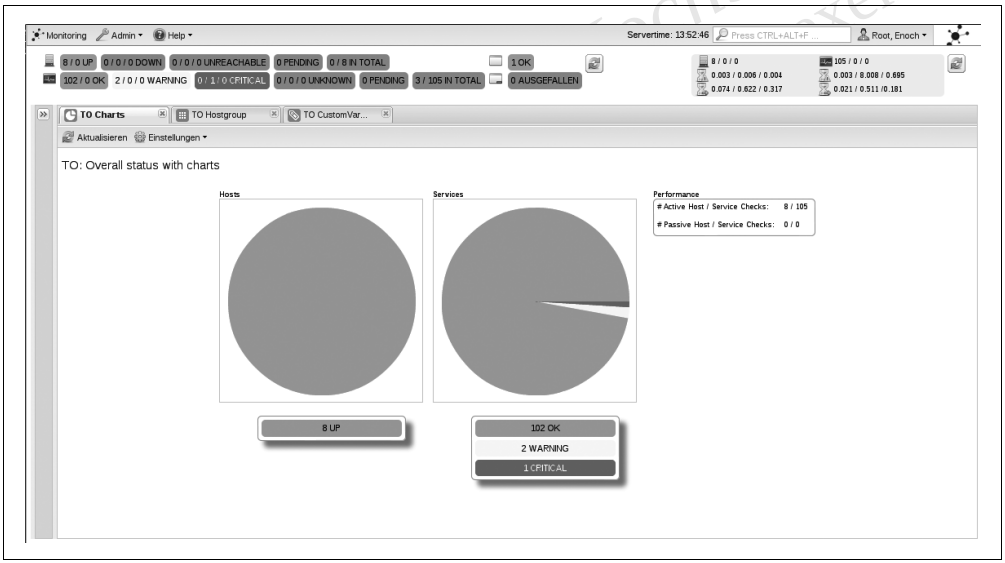


Abbildung 3-15: Screenshot TO Charts

Sie sehen in dieser Ansicht zunächst nicht, welche Maschinen und/oder Dienste sich in welchem Status befinden, sondern nur, wie viele sich in den jeweiligen Status befinden. Über die Schaltflächen unter den Diagrammen können Sie aber schnell zu einer nach dem entsprechenden Status gefilterten Liste der Maschinen beziehungsweise Dienste springen.

TO Hostgroups – Die Gruppenansicht: Der Name dieser Ansicht ist eigentlich etwas unglücklich gewählt, da Ihnen nicht nur Maschinen-, sondern auch Dienstgruppen angezeigt werden. Wir nennen diese Ansicht deshalb hier einfach Gruppenansicht. Für jede Gruppe sehen Sie auf einen Blick den Status und haben die Möglichkeit direkt zu einer entsprechenden Detail-Anzeige zu springen. In Abbildung 3-16 sehen Sie ein Beispiel für diese Ansicht.

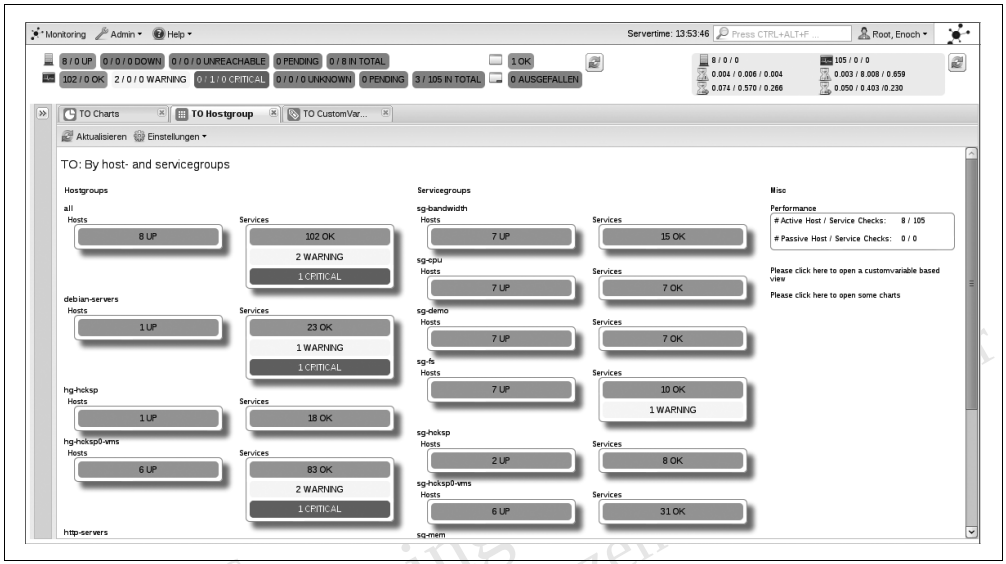


Abbildung 3-16: Screenshot TO Hostgroup

Da Ihnen diese Ansicht die Statusübersicht für jede Gruppe zeigt, ist sie ein sehr nützliches Werkzeug. Voraussetzung ist allerdings, dass Sie Gruppen entsprechend einsetzen und diese möglichst auch gleichzeitig angezeigt werden können. Bei einer sehr großen Anzahl an Gruppen können Sie allerdings über entsprechende Rechte immer noch dafür sorgen, dass einzelnen Rollen/Benutzern nur die für sie relevanten Gruppen angezeigt werden.

TO CustomVariables – Ihre eigenen Gruppen: Zusätzlich zu den in Icinga-Core vorgesehenen und bereits aus der klassischen Oberfläche bekannten Gruppen, analog zu den Maschinen- und Dienstegruppen in den Konfigurationsdateien, können Sie nach Belieben weitere Gruppen anhand benutzerdefinierter Variablen (siehe Rezept 4.10, *Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen*) bilden. Hierzu fügen Sie Ihren Konfigurationsobjekten entsprechende eigene Variablen hinzu, die Sie dann als Gruppierungskriterium verwenden können. Die Ansicht entspricht dabei der der `TO Hostgroups`. Abbildung 3-17 zeigt Ihnen ein Beispiel für eine entsprechende Gruppierung.

Sie können die Gruppierung zu Maschinengruppen zwar sehr gut verwenden, um die Konfiguration zu vereinfachen (siehe Rezept 4.6, *Maschinen zu Gruppen zusammenfassen*

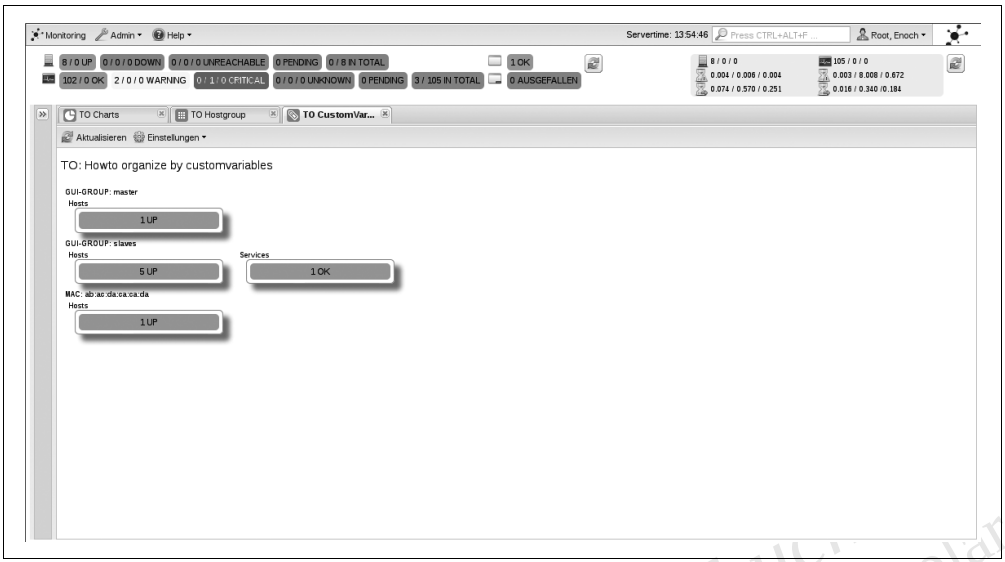


Abbildung 3-17: Screenshot TO Custom Variables

(host-group)), es entstehen dadurch aber auch leicht sehr viele Gruppen, die für eine Gruppenansicht nur schlecht geeignet sind. Entsprechend ist die Gruppierung nach benutzerdefinierten Variablen eine gute Möglichkeit, nach Bedarf Gruppen für die Anzeige zu bilden. Dabei werden Ihnen die benutzerdefinierten Variablen eben wie Gruppen angezeigt. Zu Beginn und bei kleinen Umgebungen werden Sie diese Möglichkeit aber nicht unbedingt benötigen.

Datenzugriff

Unter der Gruppe `Data` bietet Ihnen Icinga-Web dann außerdem Ansichten für den gezielten Zugriff auf bestimmte Teile der Konfiguration. In der hier betrachteten Version 1.8.1 stehen Ihnen dabei zunächst 15 vorgefertigte Ansichten zur Verfügung, die wir Ihnen im Folgenden erläutern werden.

Unhandeled – Die noch nicht bearbeiteten Probleme: Um die typischen Fragen wie »Was läuft nicht?« beziehungsweise »Wo gibt es Probleme?« zu beantworten, stehen Ihnen zunächst drei Ansichten zur Verfügung, die nur Problemmeldungen (Status `WARNING` und `CRITICAL`) anzeigen, aber nicht den zusätzlichen Status `ACKNOWLEDGED` aufweisen. Dabei können Sie entweder nur Maschinen (Unhandeled Host Problems), Dienste (Unhandeled Service Problems) oder alle zusammen (Unhandeled Problems beziehungsweise Open Problems) betrachten. Abbildung 3-18 zeigt Ihnen ein Beispiel für die zusammenfassende Ansicht für Maschinen und Dienste.

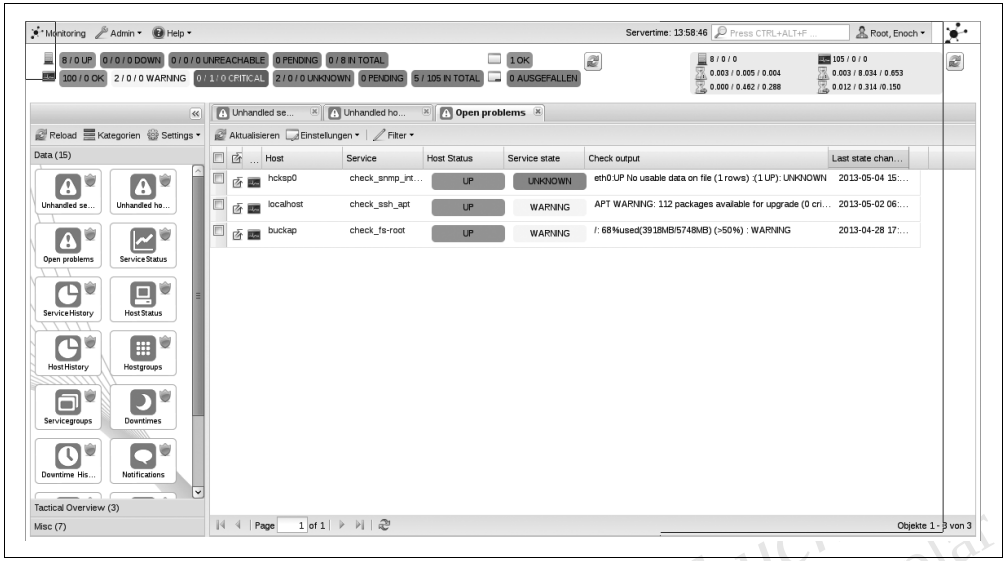


Abbildung 3-18: Screenshot Open Problems

Häufig werden Sie genau diese Seiten im Auge behalten, da Ihnen hier die neu aufgetretenen Probleme angezeigt werden. Das gilt allerdings nur unter der Voraussetzung, dass Sie erkannte und in Arbeit befindliche Probleme in den Status `ACKNOWLEDGED` versetzen.

Maschinen- und Dienst-Status und -Historie: Wenn Sie alle Maschinen oder Dienste übersehen möchten, verwenden Sie die Ansichten Host-Status beziehungsweise Service-Status. Sie haben in der angezeigten Liste dabei die Möglichkeit, diverse weitere Informationen abzurufen. Abbildung 3-19 zeigt Ihnen ein Beispiel dieser Status-Ansicht für Maschinen.

Neuerung ab Icinga Version 1.8.1: In den Versionen bis einschliesslich 1.6.2 waren diese Funktionen nur zum Teil vorhanden und sie wurden direkt in der jeweiligen Zeile dargestellt. Wenn Sie eine ältere Version kennen, werden Sie sich eventuell erst daran gewöhnen müssen, für den Abruf der Graphen einen Klick mehr zu benötigen. Seit Version 1.9 können Sie Symbole für Details allerdings auch zur direkten Anzeige auswählen. Klicken Sie in dem Detail-Menu mit der rechten Maustaste auf das Symbol und wählen Sie »Move to Grid«. Anschließend wird das entsprechende Symbol direkt in der Übersicht angezeigt.

Im Vergleich zu früheren Versionen befindet sich seit Version 1.8.1 ganz links ein Symbol zur Anzeige von weiteren Details. Wenn Sie auf dieses klicken, öffnet sich ein zusätzliches Menü, über das Sie dann auf die folgenden Funktionen zugreifen können:

- Details zeigt Ihnen die Konfigurationsoptionen, Plugin-Ausgaben und Beziehungen (wie etwa zugeordnete Kontakte) an.

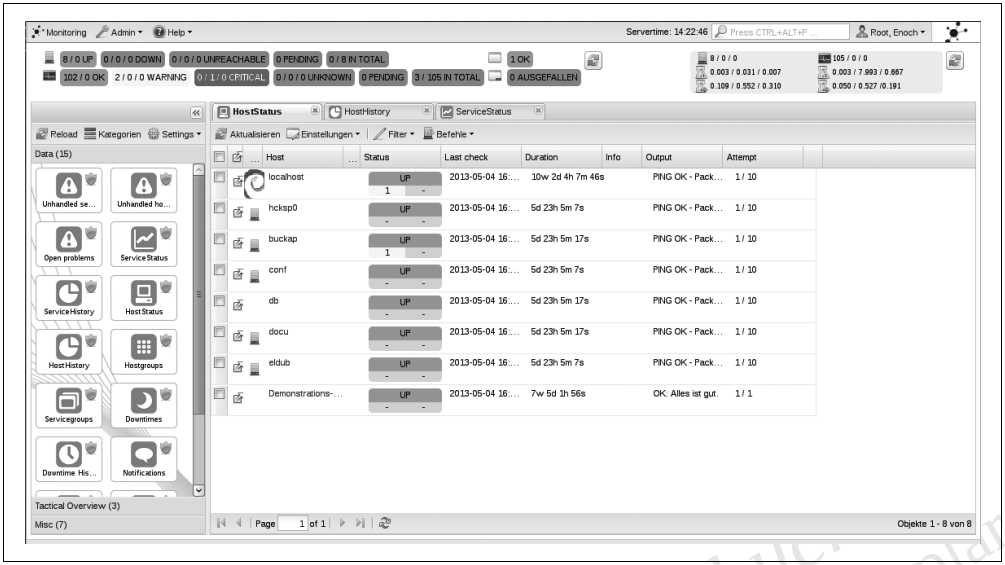


Abbildung 3-19: Screenshot Host Status

- Bei Maschinen haben Sie dann die Wahl zwischen Services, History und Statusmap, die Ihnen die jeweiligen Ansichten mit Fokus/Filter auf der jeweilige Maschine in einem neuen Tab öffnen.
- Bei Diensten stehen Ihnen im Gegensatz zu den Optionen bei Maschinen Details, Host Details, Host und History zur Verfügung.
- Sofern Sie PNP4Nagios verwenden und integriert haben, können Sie sich mit Graph den jeweils ersten Graphen in einem überlagerten Fenster anzeigen lassen oder mit Detail die PNP4Nagios-Seite zu der Maschine in einem neuen Tab öffnen.

Die Darstellungen der Ereignisse, die Sie über die Funktion History bei Maschinen und Diensten abrufen können, sind dabei so gefiltert, dass eben nur die jeweilige Maschine beziehungsweise der jeweilige Dienst dargestellt wird. Über die Ansichten HostHistory und ServiceHistory können Sie auf die ungefilterten Ereignisse aller Maschinen beziehungsweise Dienste zugreifen. Abbildung 3-20 zeigt Ihnen ein Beispiel für die Ansicht der Historie für Maschinen.

Bei Installationen mit mehreren Hundert Maschinen und einem entsprechenden Vielfachen an Diensten verlieren diese Übersichten allerdings an Bedeutung. Sie sind dann viele Seiten lang, weshalb ein zielgerichteter Zugriff besser über die Suche (siehe weiter vorne unter Menüleiste) oder Gruppendarstellungen (siehe folgender Abschnitt) erfolgen kann.

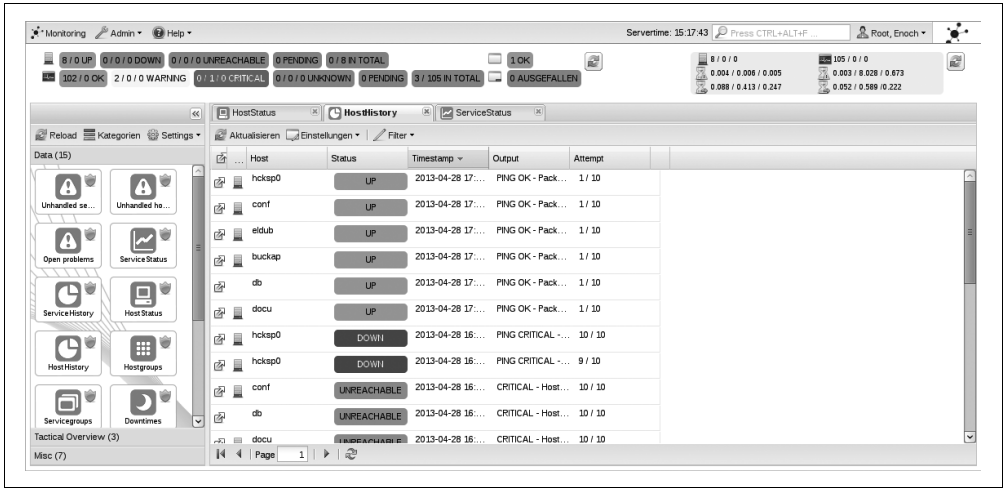


Abbildung 3-20: Screenshot Host History

Gruppendarstellungen: Wenn Sie den Zustand einer bestimmten Gruppe (zur Gruppenbildung siehe Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)* und Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*) überprüfen möchten, können die Gruppendarstellungen für Maschinen (Hostgroups) oder Dienste (Servicegroups) verwenden. Diese zeigen Ihnen alle bekannten Gruppen (auf die Ihre Benutzerrolle Zugriff hat) mit ihrem Status an. Zu jeder Gruppe erhalten Sie verlinkte Symbole, über die Sie zu den Maschinen (bei Maschinengruppen) und/oder Diensten der Gruppe springen können. Abbildung 3-21 zeigt Ihnen ein Beispiel für die Ansicht der Maschinengruppen.

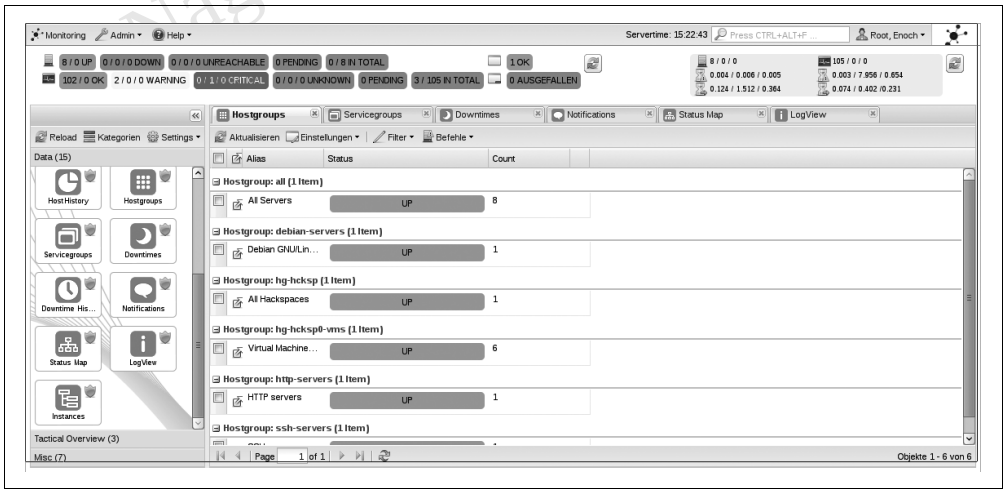


Abbildung 3-21: Gruppendarstellungen für Maschinen (Hostgroups)

Wenn Sie eine große Installation mit vielen Gruppen betreuen, können Sie über entsprechende Rollen dafür sorgen, dass diese Ansicht dennoch übersichtlich bleibt.

Geplante Ausfallzeiten und deren Historie: Wenn Sie geplante Ausfallzeiten (siehe Rezept 3.13, *Kommentare, Acknowledgments und Downtimes nutzen*) in Icinga einpflegen, werden Ihnen die Ansichten Downtimes und Downtime History nützlich sein. Hier können Sie sich alle hinterlegten Ausfallzeiten beziehungsweise die Ausfallzeiten in der Vergangenheit anzeigen lassen.

Wenn Sie mit mehreren Personen an einer Installation arbeiten, können Sie durch die Angabe von Ausfallzeiten vermeiden, dass Kollegen aufgrund von wartungsbedingten Fehlern aktiv werden. Über Downtimes und Downtime History können Sie dann auch sehen, welche Dienste für welche Zeiträume für Wartungen eingeplant sind.

Benachrichtigungshistorie: Über Notifications können Sie einsehen, welche Benachrichtigungen das System versendet hat. So können Sie sich innerhalb der Oberfläche darüber Informieren, was genau das System vermeldet hat. Abbildung 3-22 zeigt Ihnen ein Beispiel dieser Ansicht.

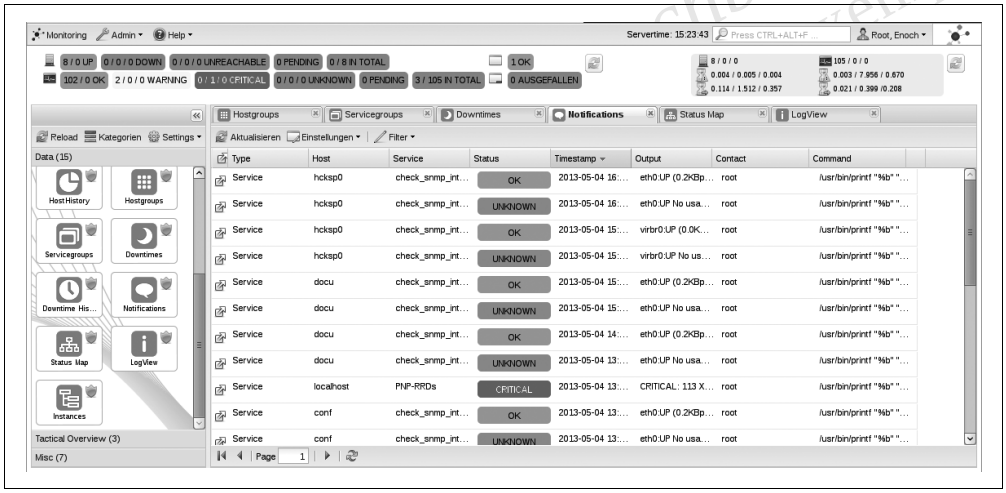


Abbildung 3-22: Cronk Notifications

Sofern Sie über die passenden Rechte verfügen, können Sie dabei auch Benachrichtigungen einsehen, die an andere Benutzer versendet wurden. Wenn Ihr System viele Benachrichtigungen versandt hat, können Sie oben im Fenster den Menüpunkt Filter verwenden, um die Anzeige auf bestimmte Maschinen oder Dienste zu beschränken.

Statuskarte: Wie auch die klassische Oberfläche bietet Ihnen Icinga-Web als Status Map eine auf den Eltern-Kind-Beziehungen basierende graphische Darstellung. Die Darstellung ist allerdings überarbeitet und verwendet beispielsweise Animationen bei der Auswahl der Maschine, auf die die Ansicht zentriert wird. Abbildung 3-23 zeigt Ihnen ein Beispiel für diese Statusansicht.

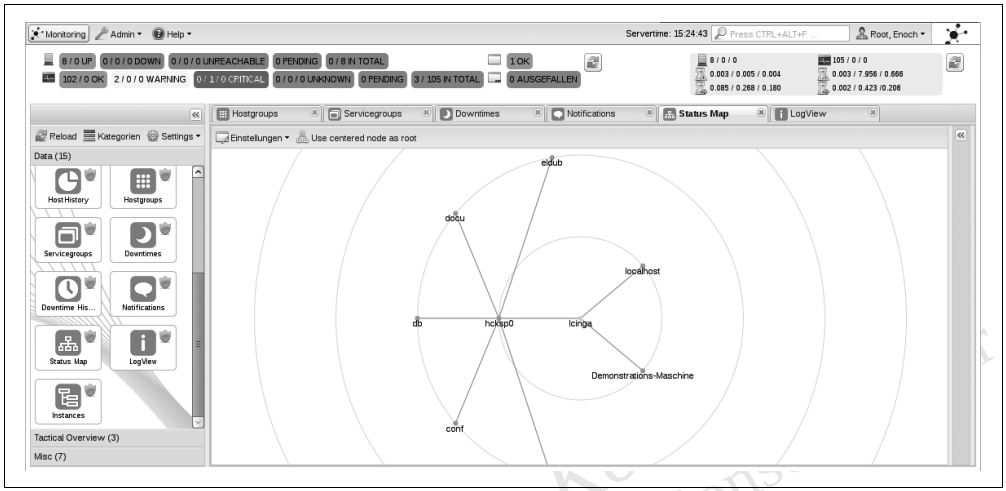


Abbildung 3-23: Status Map

Sofern Sie die Eltern-Kind-Beziehungen konsequent verwenden, kann diese Ansicht sehr hilfreich sein. Bei größeren Installationen mit vielen Teilnetzen können Sie hier leicht erkennen, welche weiteren Teile ebenfalls von einem Ausfall betroffen sind.

Protokollansicht (LogView): Über die Protokollansicht und dort über die Schaltfläche LogView haben Sie aus der Weboberfläche heraus Zugriff auf die Meldungen, die Icinga generiert hat. Dies sind dieselben Inhalte, die unter *nix-Systemen per Vorgabe auch in das Syslog geschrieben werden. Abbildung 3-24 zeigt Ihnen ein Beispiel für diese Ansicht.

Bei größeren Installationen werden Ihnen hier schnell viele Meldungen aufgelistet. Nutzen Sie dann gegebenenfalls die Filtermöglichkeit über die Schaltfläche Filter oben im Fenster.

Instances – steuern Sie Ihr(e) Icinga(s): Die Ansicht Instances ist insbesondere bei verteilten Installationen mit mehreren, parallel laufenden Instanzen von Icinga interessant. Sie können von hier Befehle an die Instanzen absetzen und so beispielsweise alle Prüfungen stoppen oder starten.

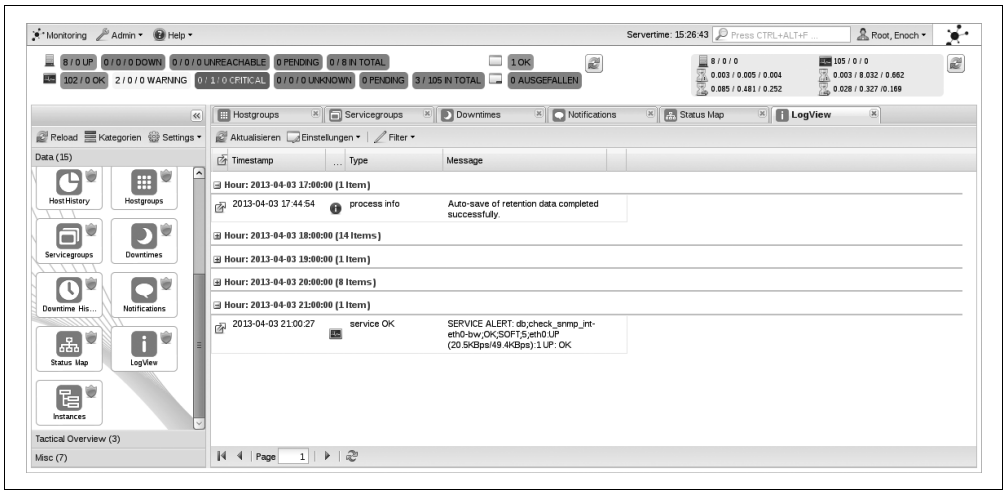


Abbildung 3-24: IcingaWeb – LogView

Reporting

Der Abschnitt Reporting bietet nur den gleichnamigen Cronk Reporting. Über diesen können Sie auf einen verbundenen Jasper-Server für die Erstellung von Berichten zugreifen. Da wir die Anbindung an einen Jasper-Server in diesem Buch jedoch nicht behandeln werden, gehen wir nicht weiter auf diese Ansicht ein.

Miscellaneous – Sonstige Ansichten

In der Kategorie sonstige Ansichten stehen Ihnen weitere Zusammenstellungen zur Verfügung, die nicht in eine der anderen Kategorien gepasst haben. Hier finden Sie Vorlagen zum Erstellen eigener Ansichten, können die Dokumentationen in deutsch oder englisch abrufen und können auf die Konfigurationsoptionen von Icinga-Web zugreifen. Wir werden Ihnen diese Ansichten im Folgenden vorstellen.

Mit den Cronks Portal 1Column, Portal 2Column und Portal 3Column stehen Ihnen Vorlagen für eigene Ansichten zur Verfügung. Entsprechend ihrem Namen haben die Vorlagen 1-3 Spalten. Erstellen Sie sich eine eigene Ansicht, indem Sie zunächst die gewünschte Vorlage wählen. Sie können dann mit der Maus vorhandene Cronks in diese Ansicht ziehen oder diese dort verschieben. Abbildung 3-25 zeigt Ihnen ein Beispiel für einen auf diese Weise erstellten, zwei-spaltigen Cronk, in den wir wir oben die Ansichten `Unhandeled HostProblems` (links) und `Unhandeled ServiceProblems` (rechts) und unten die Ansichten `Hostgroups` (links) und `Servicegroups` (rechts) eingefügt haben. Wenn Sie mit einer von Ihnen erstellen Ansicht zufrieden sind, können Sie die Ansicht über die Schaltfläche Save Cronk, die Ihnen nach einem Rechtsklick auf den Tabulator Ihres Cronks angezeigt wird, speichern. Die Abbildung zeigt Ihnen zusätzlich die Ihnen daraufhin angebotenen Optionen zur Speicherung.

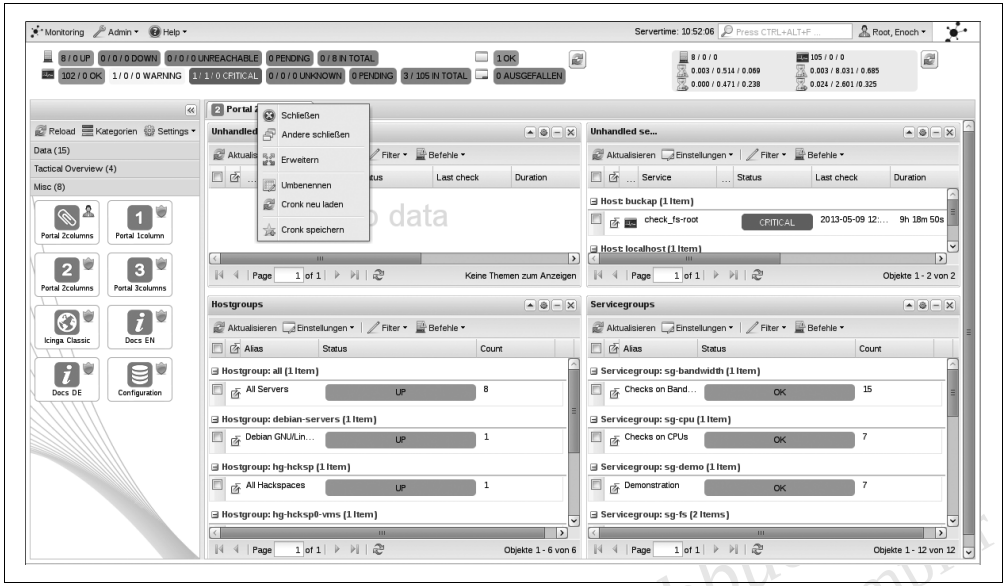


Abbildung 3-25: Custom Cronk

Geben Sie Ihrer Ansicht hier einen Namen und eine Beschreibung, wählen Sie aus, in welcher Kategorie sie angezeigt werden soll und wählen Sie ein entsprechendes Symbol aus. Optional können Sie den Cronk auch für andere Benutzer freigeben, ansonsten steht er nur Ihrem Benutzerkonto zur Verfügung.

Wenn Sie viel mit Icinga arbeiten, werden Sie diese Anpassungsfunktionen vermutlich sehr zu schätzen lernen. Hier können Sie sich aus den vorhandenen Ansichten und ohne jegliche Programmierkenntnisse eigene Ansichten nach Bedarf zusammenstellen. Beachten Sie hierzu bitte auch die weiteren Anpassungmöglichkeiten über Filter, die wir Ihnen später in der Diskussion näherbringen werden.

Über die Cronks Docs DE und Docs EN können Sie auf die deutsche beziehungsweise englische Dokumentation zugreifen. Diese wird Ihnen dann im Anzeigebereich rechts als neuer Tabulator angezeigt.

Mit dem Cronk Icinga Classic können Sie die klassische Oberfläche innerhalb des Anzeigebereiches von Icinga-Web aufrufen und verwenden. Die Beschreibung der dort enthaltenen Funktionen finden Sie im vorangegangenem Rezept (siehe Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*).

Configuration – Konfigurationsoptionen von Icinga-Web: Über den Cronk *Configuration* können Sie sich die Parameter von Icinga-Web und Ihre Werte ansehen. Dies ist für die reine Verwendung der Oberfläche allerdings nicht notwendig, weshalb wir nicht weiter auf diese Ansicht eingehen werden.

Diskussion

Neben den bereits dargestellten Möglichkeiten der Anpassung durch Zusammenstellen verschiedener Ansichtstypen können Sie die meisten Ansichten auch durch Filterung weiter Ihren Bedürfnissen entsprechend modifizieren. Hierzu wählen Sie in der fraglichen Ansicht die dort enthaltene Filter-Auswahl und setzen einen entsprechenden Filter. Welche Filter Ihnen dabei zur Verfügung stehen, hängt von der jeweiligen Ansicht ab. Für Maschinen und Dienste stehen Ihnen hier zunächst sowohl die Namen, als auch die Status zur Verfügung, wobei Sie auch Platzhalter verwenden können und damit bereits recht flexibel sind.

Eine besondere Bedeutung messen wir der Möglichkeit bei, nach benutzerdefinierten Variablen (siehe Rezept 4.10, *Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen*) zu filtern. Auf diese Weise können Sie sich nach Bedarf völlig eigene Ordnungskriterien schaffen – eine Möglichkeit, die es in Nagios und der klassischen Icinga-Oberfläche bislang nicht gab. Insbesondere bei umfangreichen Umgebungen werden Sie diese vermutlich nutzen wollen, gerade weil das übliche Gruppierungsmerkmal über die Ordnung zu Maschinen- beziehungsweise Dienstgruppen dann nicht mehr ausreicht und Sie die Maschinengruppen als Vereinfachung für die Konfiguration nutzen können und sollten (siehe hierzu Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*).

Zusätzlich bietet Ihnen Icinga-Web den Zugriff auf verschiedene Systeme (Icinga-Classic, PNP4Nagios, Hilfe-Seiten) aus einem einzelnem System heraus. Damit wird es zu einer Schaltzentrale, wie sie sich sicher viele Nagios-Benutzer lange gewünscht haben. Allerdings ist damit auch eine zusätzliche Belastung der Systeme verbunden: Icinga-Web benötigt im Vergleich zu den bisherigen Oberflächen deutlich mehr Ressourcen und fühlt sich zur Zeit noch teilweise etwas träge an, wenn es viele Daten für eine Anzeige abrufen muss. Zumindest bei umfangreichen Installationen macht es dies durch die vorhandene Flexibilität aber wieder wett, wenn Sie entsprechende Filter nicht über CGI-Optionen »programmieren« wollen. Falls die Belastung durch Icinga-Web Ihre Monitoring-Maschine stark beansprucht, können Sie erwägen die Oberfläche auf eine eigene Maschine auslagern – auf die Details einer derart verteilten Konfiguration werden wir in diesem Buch allerdings nicht weiter eingehen.

Siehe auch

- Projekt-Seiten des Icinga-Projektes: *https://www.icinga.org/*
- Projekt-Seiten zur Beschreibung von Icinga-Web: *https://www.icinga.org/about/icinga-web/*
- Rezept zur Benutzerverwaltung über http-auth: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zur Einführung in die klassische Oberfläche: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*

- Rezept zu den benutzerdefinierten Variablen: Rezept 4.10, *Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen*
- Rezepte zur Gruppierung von Maschinen Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)* und Diensten Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*
- Rezept zur Nutzung der Ausfallzeiten: Rezept 3.13, *Kommentare, Acknowledgments und Downtimes nutzen*

3.5 Vereinfachen der Konfiguration mit dem Vorlagen-System

Problem

Die verschiedenen Objekttypen der Konfigurationsdateien kennen viele unterschiedliche Parameter, die die Dateien schnell unübersichtlich werden lassen. Sie möchten bei einzelnen Definitionen aber nur die relevanten Optionen sehen und auf einfache Art und Weise Änderungen durchführen können.

Lösung

Durch konsequente Nutzung von Vorlagen können Sie Redundanzen vermeiden und für mehr Ordnung in den Konfigurationsdateien sorgen. Gleichzeitig treffen Sie dabei Vorkehrungen, um mit einzelnen Einträgen Änderungen an ganzen Gruppen durchführen zu können. Die Möglichkeit, gezielt einzelne Maschinen/Services zu ändern, bleibt Ihnen dabei erhalten.

Vererbung über Vorlagen

Bei den Konfigurationsdateien von Nagios/Icinga, die wir Ihnen im folgenden Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga* vorstellen, werden Sie Vorlagen insbesondere bei den Konfigurationsobjekten für

- Maschinen (`host`, siehe Rezept 4.1, *Maschinen einbinden (host)*),
- Dienste (`service`, siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) und
- Kontakte (`contact`, siehe Rezept 4.4, *Kontakte definieren (contact)*)

nutzen. Die Definition einer Vorlage unterscheidet sich dabei nur durch die zusätzliche Angabe der Option

```
register                        0
```

von einer zur direkten Nutzung bestimmten Definition. Intern bedeutet dies eben, dass das Objekt nicht registriert wird, sondern lediglich als Vorlage für Basisattribute verwendet werden soll. Die Optionen der einzelnen Objekte zeigen wir Ihnen in den jeweiligen Rezepten genauer.

Abbildung 3-26 zeigt, wie Teile der Konfiguration aufeinander aufbauen können. Die Vorlagen haben wir hier hellgrau dargestellt, tatsächliche Maschinen- und Servicedefinitionen hingegen sind mit dunkelgrauer Farbe gekennzeichnet. Als oberste Vorlage in diesem Baum können Sie die bei der Installation erstellte generische Vorlage verwenden. Für einzelne Teilbereiche erben Sie die Einstellungen aus dieser und überschreiben nur für den jeweiligen Bereich abweichende Optionen.

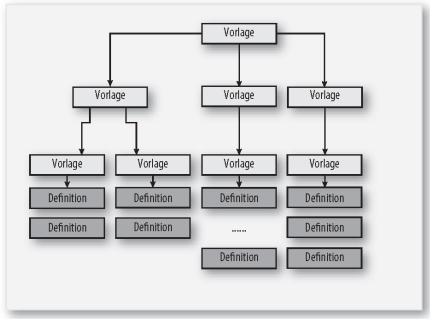


Abbildung 3-26: Darstellung einer Struktur

Wenn Sie die vorgeschlagene Struktur für die Ablage der Konfigurationsdateien nutzen, ergeben sich automatisch sinnvolle Gruppen, für die Sie entsprechende Vorlagen nutzen können. Nehmen wir die folgende Struktur als Beispiel:

```
config.d/
|-- infrastructure
|-- server
`-- printer
```

Sie können sich hier Vorlagen jeweils für die Maschinen und Services in den einzelnen Untergruppen anlegen. Abbildung 3-27 zeigt dies für die Gruppe `infrastructure`. Der linke Teilbaum von Abbildung 3-26 wird entsprechend detaillierter dargestellt.

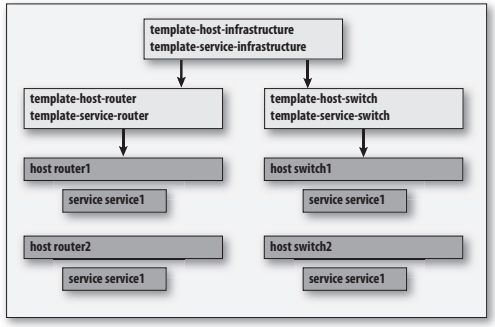


Abbildung 3-27: Strukturumsetzung mit Vorlagen

Von der systemweiten Vorlage leiten Sie eine Vorlage für alle Maschinen und Services des Bereiches Infrastruktur ab. Von dieser Infrastruktur-Vorlage abweichende Vorgaben für Router beziehungsweise Switches geben Sie dann wiederum in einer abgeleiteten Vorlage an.

Mehrfachvererbung durch mehrere Vorlagen

Nagios/Icinga bietet Ihnen über diesen einfachen Vererbungsmechanismus hinaus auch eine Mehrfachvererbung. Sie haben also die Möglichkeit, nicht nur eine Vorlage zu referenzieren, sondern beliebig viele. Dadurch lassen sich weitere Redundanzen vermeiden.

Wenn Sie eine Struktur wie aus dem eben angeführten Beispiel, jedoch in einer Umgebung mit mehreren Standorten haben, benötigen Sie entsprechende Teilbäume für jeden Standort. Möchten Sie jetzt für Router und Switches etwa jeweils andere Icons verwenden, dann müssten Sie in der vorherigen Definition die entsprechenden Optionen in allen Teilbäumen setzen – und bei einer Änderung an ebenso vielen Stellen ändern. Die Mehrfachvererbung ermöglicht Ihnen an dieser Stelle hingegen, Vorlagen für die verschiedenen Icons zu erstellen und dann in den Teilbäumen einzubinden.

In Abbildung 3-28 ist dies dargestellt, wobei die Vorlagen für die Icons weiß eingefärbt sind. Bei dieser Vorgehensweise müssen Sie nur noch in der Icon-Vorlage die Datei angeben, während alle anderen Definitionen dann eben eine Ihrer Icon-Vorlagen einbinden. Wenn Sie jetzt das für Switches genutzte Icon ändern möchten, müssen Sie nur noch die entsprechende Vorlage anpassen, um alle Switches gleichzeitig entsprechend zu modifizieren.

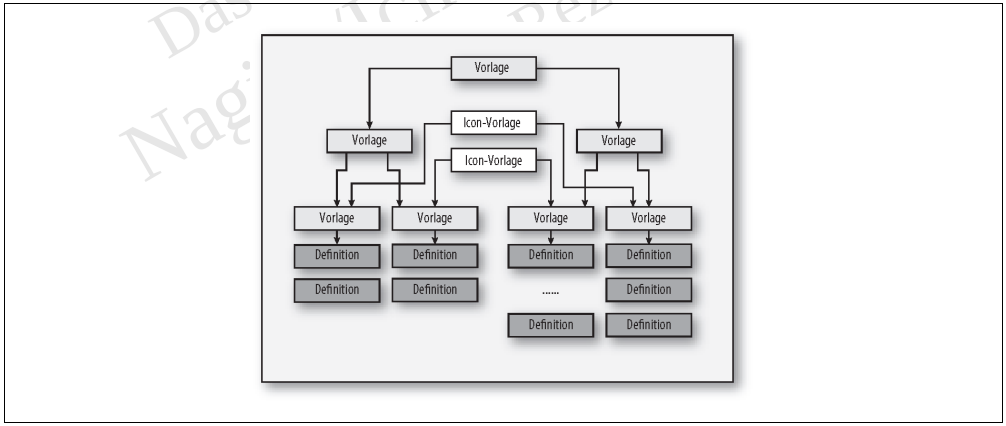


Abbildung 3-28: Darstellung der Mehrfachvererbung

Diskussion

Die gezeigten Mechanismen der Vererbung können Sie vielfältig einsetzen, um zusätzliche Strukturen in Ihren Konfigurationsdateien zu bilden. Dem Vorteil, Redundanzen zu vermeiden und Änderungen zentral durchführen zu können, steht dabei der Nachteil gegenüber, dass Sie zur Einbindung neuer Maschinen/Dienste Kenntnisse über die zu

nutzenden Vorlagen besitzen müssen. Typischerweise arbeitet aber nur ein eingeschränkter Personenkreis direkt mit den Konfigurationsdateien, innerhalb dessen sich entsprechende Absprachen/Dokumentationen relativ problemlos realisieren lassen. Entscheiden Sie für Ihre Umgebung, ob und wie Sie diese Mechanismen nutzen möchten.

Neben der hier als Beispiel dienenden Einbindung von Icons können Sie Vorlagen auch für ganz andere Inhalte verwenden, wie beispielsweise

- der Angabe von unterschiedlichen (internen) Domänen zur Verwendung bei der Adressauflösung über entsprechende Maschinen-Vorlagen,
- der Zuweisung von Kontakten über Service-Vorlagen für Abteilungen,
- der Trennung von unterschiedlichen Service Level Agreements (SLAs), also definierten Reaktionszeiten und entsprechenden Benachrichtigungsoptionen in Vorlagen für Maschinen und/oder Dienste, oder
- die Einbindung von Links zu externen Systemen (siehe Rezept 4.11, *Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo)* und die Teile zu PNP4Nagios in Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten* als Beispiel).

Sie können als Strukturierungsmerkmale anstelle rein technischer also auch organisatorische Gruppen abbilden. Es kann sich dabei beispielsweise um die Vorlagen für einzelne Länder oder Abteilungen handeln, über die dann etwa entsprechende Kontakte zugewiesen werden. Mit Hilfe der Mehrfach-Vererbung können Sie diverse Unterscheidungsmerkmale dann weiter individuell zusammenstellen, indem Sie über die gerade angesprochenen Länder-Vorlagen zwar die Kontakte zuweisen, für die technischen Details aber Vorlagen für unterschiedliche Gerätetypen verwenden. Dazu müssen Sie dann lediglich beide Vorlagen in den entsprechenden Geräten referenzieren. Stellen Sie dabei sicher, dass die Vorlagen unterschiedliche Optionen spezifizieren (sich also ausschließend ergänzen) – anderenfalls müssen Sie auf die Reihenfolge achten, da diese dann darüber entscheidet, welche Vorlage für ein bestimmtest Datum tatsächlich genutzt wird.

Siehe auch

- Dokumentation zu Objektvererbung bei Icinga: *http://docs.icinga.org/latest/de/objectinheritance.html* beziehungsweise für Nagios *http://nagios.sourceforge.net/docs/nagioscore/3/en/objectinheritance.html*
- Dokumentation zu zeitsparenden Tricks für Objektdefinitionen bei Icinga: *http://docs.icinga.org/latest/de/objecttricks.html* beziehungsweise für Nagios *http://nagios.sourceforge.net/docs/nagioscore/3/en/objecttricks.html*
- Kapitel zu den Konfigurationsobjekten von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*, insbesondere Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* und Rezept 4.4, *Kontakte definieren (contact)*

- Rezept zur Integration mit Fremd-Systemen: Rezept 4.11, *Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo)* und als Beispiel die Integration mit PNP4Nagios in Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*

3.6 Aufbau und strukturierte Ablage der Konfigurationsdateien

Problem

Sie möchten den Aufbau der Konfigurationsdateien verstehen und kennenlernen, wie Sie diese strukturiert ablegen können.

Lösung

Wir werden Ihnen zunächst beschreiben, wie die Konfigurationsdateien prinzipiell aufgebaut sind, und die Abhängigkeiten zwischen den Konfigurationsobjekten sowie Möglichkeiten zur Ablage erläutern. Darauf aufbauend geben wir Ihnen Hinweise, wie Sie Ihre Konfigurationsdateien strukturieren und auf diese Weise eine Ordnung in den Konfigurationsdateien herstellen können.

Prinzipieller Aufbau

Nagios/Icinga sehen von Hause aus eine Aufteilung der Anweisungen vor. Dabei werden Befehle, deren Ausführung später eine Prüfung ergeben soll, zunächst generisch definiert und alle variablen Teile der Befehlsanweisung mit Platzhaltern gekennzeichnet. Diese werden dann bei der Ausführung gegen die eigentlichen Inhalte ausgetauscht. Im Kontext von Nagios/Icinga wird bei diesen Platzhaltern von Makros gesprochen, die wir Ihnen in Rezept 4.3, *Befehle hinzufügen (command)* genauer erläutern werden.

Wir beschreiben zunächst ein generisches Beispiel hierzu. Die dabei angeführten Schritte werden wir detailliert in jeweils einzelnen Rezepten erläutern, auf die wir hier verweisen. Zunächst möchten wir Ihnen jedoch einen Überblick verschaffen.

Ein Befehl xyz soll als Prüfung eingerichtet und mit unterschiedlichen Optionen ausgeführt werden. Zunächst müssen dazu die Maschine als Konfigurationsobjekt host (siehe hierzu Rezept 4.1, *Maschinen einbinden (host)*) und der Befehl als Konfigurationsobjekt command (siehe Rezept 4.3, *Befehle hinzufügen (command)*) definiert werden. Die Befehlsdefinition wird hierbei mit Makros als Platzhalter versehen, typischerweise mindestens drei, nämlich

- für den Pfad zum Plugin,
- für die Netzwerkadresse des Gerätes und
- für weitere über die Dienstdefinition festzulegende Optionen für das Plugin.

Nun können Sie neue Dienste als Konfigurationsobjekt definieren (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*), die diesen Befehl xyz auf dem jeweiligen Zielgerät mit dem spezifizierten Parameter ausführen. Abbildung 3-29 zeigt Ihnen diesen grundsätzlichen Mechanismus, wobei die von Ihnen zu definierenden Teile links und die automatisch ablaufenden Ersetzungen rechts dargestellt sind.

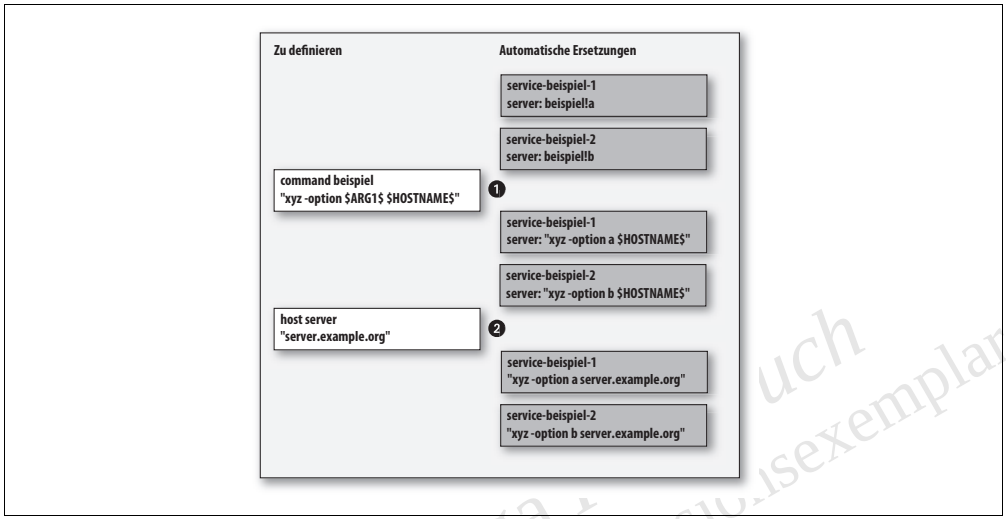


Abbildung 3-29: Ersetzungsmechanismus bei der Referenzkette Befehl – Dienst – Maschine

Dabei referenziert ein Dienst genau einen Befehl. Er kann aber auf beliebig vielen Maschinen ausgeführt werden. Wie genau Sie diese Zuordnungen vornehmen, erfahren Sie in den folgenden Rezepten noch.

Spezifikation der zu nutzenden Konfigurationsdateien

Welche Dateien Nagios/Icinga als Konfigurationsdateien einliest, legen Sie in der Hauptkonfigurationsdatei fest. Bei unserer Referenzinstallation ist dies die Datei */usr/local/icinga/etc/icinga.cfg*. Sie haben hier zwei Möglichkeiten:

- Die Referenzierung einzelner Dateien über die Option `cfg_file`
- oder ganzer Verzeichnisse inklusive ihrer Unterverzeichnisse über die Option `cfg_dir`.

Bei der Installation wurden einzelne Dateien referenziert, die für die lokale Maschine automatisch eingerichtet wurden. Bei unserer Referenzinstallation waren dies fünf Anweisungen:

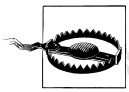
```
# You can specify individual object config files as shown below:
cfg_file=/usr/local/icinga/etc/objects/commands.cfg
cfg_file=/usr/local/icinga/etc/objects/contacts.cfg
cfg_file=/usr/local/icinga/etc/objects/timeperiods.cfg
cfg_file=/usr/local/icinga/etc/objects/templates.cfg

# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/icinga/etc/objects/localhost.cfg
```

Sie können diese Vorgehensweise übernehmen und weitere Ihrer Konfigurationsdateien dann in der Hauptkonfigurationsdatei durch aus- beziehungsweise entkommentieren ab- und anschalten.

Einzelnes Konfigurationsverzeichnis: Legen Sie sich genau ein Konfigurationsverzeichnis an, dann können Sie Dateien einfach hinzufügen, umbenennen oder innerhalb des Ordners verschieben, ohne diese Änderungen jedes Mal nachpflegen zu müssen. Um Dateien temporär zu deaktivieren, können Sie sie dann einfach mit einer anderen Endung versehen. Außerdem können Sie eine Kopie des Verzeichnisses für Änderungen machen und durch eine einzelne Änderung in der Hauptkonfigurationsdatei zwischen diesen Konfigurationen umschalten.

Wir nutzen in diesem Buch als Konvention ein einzelnes Konfigurationsverzeichnis unter */usr/local/icinga/etc/config.d*. Wenn Sie möchten, legen Sie sich analog ein entsprechendes Verzeichnis an. Bei einer paketbasierten Installation wäre dies denn der Ordner */etc/icinga/config.d*:

- Erstellen Sie das Verzeichnis (möglichst innerhalb des */usr/local/icinga/etc*-Ordners, um die spätere Sicherung der Konfigurationsdateien zu vereinfachen),
- verschieben Sie alle eigenen bislang in der Hauptkonfigurationsdatei aktivierten Konfigurationsdateien in das neue Verzeichnis,
- kommentieren Sie die bisher aktiven Konfigurationsanweisungen zu den einzelnen Dateien aus und
- aktivieren Sie Ihr neues Verzeichnis über eine entsprechende config_dir-Direktive.



Doppelte Einbindung: Achten Sie darauf die Orte für Konfigurationsverzeichnisse und Konfigurationsdateien nicht zu vermischen. Diese werden sonst mehrfach geladen.

Alternativ können Sie bei den weiteren Rezepten in diesem Buch natürlich auch ein anderes Verzeichnis verwenden oder die Konfigurationsdateien einzeln warten. Denken Sie in diesem Falle bitte an jeweils erforderliche Änderungen und/oder Anpassungen, da wir von der hier beschriebenen Konfiguration ausgehen.

Strukturierung durch Unterverzeichnisse

In diesem Konfigurationsverzeichnis und möglichen Unterverzeichnissen können Sie nun beliebig Konfigurationsdateien ablegen, diese müssen nur die Endung `.cfg` aufweisen. Sofern Sie eine größere Installation einpflegen wollen, sollten Sie dies nutzen, um eine passende Struktur abzubilden. Wenn Sie ein Netzwerk betreuen und zusätzlich Server und Drucker überwachen möchten, könnten Sie etwa eine Struktur wie in Abbildung 3-30 verwenden.

```
/usr/local/icinga/etc/
   config.d/
      infrastructure/
      server/
      printer/
```

Abbildung 3-30: Verzeichnisstruktur nach Techniktyp

Wenn Sie hingegen unterschiedliche Kunden, Abteilungen und/oder Standorte betreuen, möchten Sie eventuell eine Struktur wie in Abbildung 3-31 verwenden.

```
/usr/local/icinga/etc/
   config.d/
      Germany/
      Finland/
      South Africa/
```

Abbildung 3-31: Verzeichnisstruktur nach Land

Je nach Komplexität können Sie entsprechende Strukturen auch miteinander verschachteln. Abbildung 3-32 zeigt Ihnen ein Beispiel für eine Kombination der beiden Strukturen.

```
/usr/local/icinga/etc/
   config.d/
      Germany/
         infrastructure/
         server/
         printer/
      Finland/
         infrastructure/
         server/
         printer/
      South Africa/
         infrastructure/
         server/
         printer/
```

Abbildung 3-32: Verzeichnisstruktur nach Land und Techniktyp

Da das ganze Verzeichnis mit Unterverzeichnissen eingelesen wird, können Sie diese Struktur nach Bedarf bilden und modifizieren. Nutzen Sie dies, um Ihre Konfiguration dem sich ändernden Bedarf anzupassen.

Strukturierung durch Aufteilung in Dateien

Mit einer Installation wie unserer Referenzinstallation wurden automatisch die folgenden Dateien eingerichtet:

- *commands.cfg*
- *contacts.cfg*
- *services.cfg*
- *templates.cfg*
- *timeperiods.cfg*
- *localhost.cfg*

Diese sollen ihrer Bestimmung nach Konfigurationsanweisungen enthalten, die für die gesamte Installation gelten, und enthalten in *localhost.cfg* Anweisungen zur Überwachung der lokalen Maschine, also des Monitoring-Servers selbst. Hier ist Ihnen also zunächst eine Struktur nach Typ der Konfigurationsobjekte vorgegeben. Sie werden solche für die ganze Installation geltenden Anweisungen voraussichtlich benötigen, also können Sie diese zumindest zunächst beibehalten.

Die spezifischen Anweisungen für die einzelnen, durch Verzeichnisse abgebildeten Teile der Installation können Sie jetzt entsprechend auch weiter in Dateien aufteilen. Dabei können Sie eine Aufteilung nach dem verwendeten Anweisungstypen verwenden oder alternativ unabhängig vom Typ alle zu einer Gruppe von Geräten gehörenden Anweisungen in jeweils einer Datei ablegen. Zur Verdeutlichung, wie dies bei der oben vorgeschlagenen Struktur aussehen kann, hier die Ansätze für zwei entsprechende Umsetzungen. Abbildung 3-33 zeigt zunächst eine Aufteilung nach Anweisungstyp.

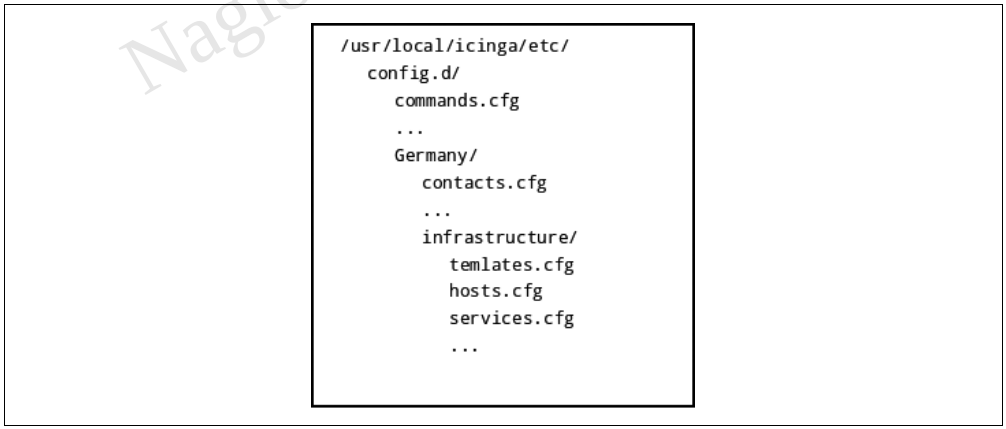


Abbildung 3-33: Beispiel einer Dateistruktur nach Objekttyp

Hierbei würden sich die Anweisungen für ein Gerät über mehrere Dateien verteilen. Zur Gegenüberstellung ist in Abbildung 3-34 eine Aufteilung nach Gerätegruppen dargestellt. Hierbei wären in den Dateien die jeweils für die Gruppe spezifischen Vorlagen, Maschi-

nen, Befehle und Services definiert, die Anweisungen für eine Maschine also nicht so stark über einzelne Dateien verstreut.

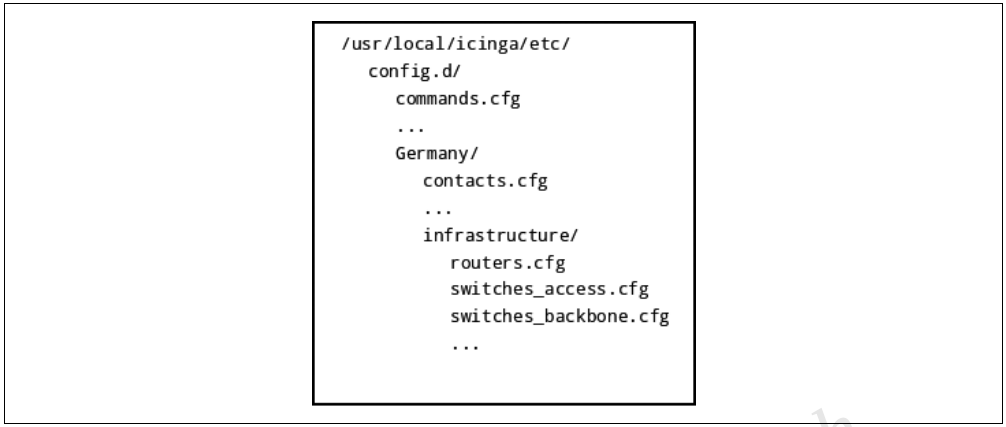


Abbildung 3-34: Beispiel einer Dateistruktur nach Gerätetyp

Diese Vorgehensweisen zur Strukturierung über Dateien schließen sich dabei nicht aus. Sie können also auch bei einer Struktur basierend auf Maschinengruppen etwa für die Dienste eine eigene Datei verwenden. Für umfangreiche Konfigurationen zeigen wir Ihnen in Rezept 3.7, *Konfigurationsdateien ändern und nach Abhängigkeiten durchsuchen*, wie Sie diese mit Hilfe der Kommandozeilen-Befehle `grep` und `find` durchsuchen können.

Diskussion

Mit zunehmender Anzahl zu überwachender Maschinen und Dienste werden die Konfigurationsdateien für die verschiedenen Konfigurationsobjekte umfangreicher. Gleichzeitig bestehen zwischen den verschiedenen Konfigurationsanweisungen vielfältige Abhängigkeiten. Um den Überblick zu behalten und auch bei umfangreichen Installationen zielgerichtet auf bestimmte Teile der Konfiguration zugreifen zu können, bietet es sich an, die Konfigurationsdateien einer geeigneten Struktur gemäß aufzuteilen.

Oft wächst die Anzahl überwachter Maschinen und Dienste allerdings erst mit der Zeit an, so dass es gar nicht immer möglich ist, eine entsprechende Struktur vorab festzulegen. Außerdem können sich Änderungen ergeben, die dann möglicherweise auch eine entsprechende Anpassung oder gar Neuplanung der Struktur bedingen. Wenn die Konfigurationsdateien tausende von Zeilen umfassen, werden Änderungen auch an sich zeitaufwendiger und bei nicht ausreichender Strukturierung kann es dann bereits unangemessen lange dauern, auch nur die richtige Stelle für eine gewünschte Änderung herauszufinden.

Eine Reorganisation (auch *Refactoring*) der Einstellungen ist gerade dann zwar auch aufwendig, kann aber eben dennoch zeitsparend wirken, wenn hierdurch regelmäßig anfallende Arbeitsschritte entsprechend stark vereinfacht werden können. Eine von unserem Vorschlag abweichende, den jeweiligen Ansprüchen angepasste Strukturierung ist natürlich ebenfalls möglich.

Siehe auch

- Rezepte zu Maschinen, Dienst- und Befehlsdefinitionen: Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* und Rezept 4.3, *Befehle hinzufügen (command)*
- Dokumentation zu Objektvererbung bei Icinga: *http://docs.icinga.org/latest/de/objectinheritance.html* beziehungsweise bei Nagios *http://nagios.sourceforge.net/docs/nagioscore/3/en/objectinheritance.html*
- Rezept zur Prüfung von Abhängigkeiten in Konfigurationsdateien: Rezept 3.7, *Konfigurationsdateien ändern und nach Abhängigkeiten durchsuchen*

3.7 Konfigurationsdateien ändern und nach Abhängigkeiten durchsuchen

Problem

Sie möchten an Ihrer Konfiguration Änderungen vornehmen und dabei sicherstellen, dass sie funktionsfähig bleibt und keine verwaisten Daten übrig bleiben.

Lösung

Abhängig von den eigentlichen Änderungen können die Abhängigkeiten zwischen den verschiedenen Konfigurationsobjekten, Plugins und Addons für zunächst nicht direkt absehbare Zusatzarbeiten sorgen. Die fälligen Anpassungen sind nicht nur von der Art der Änderung, sondern insbesondere von Ihrer individuellen Konfiguration abhängig. Deshalb können wir Ihnen hier nur allgemein aufzeigen, welche Abhängigkeiten prinzipiell bestehen können und wie Sie mit diesen umgehen.

Die prinzipielle Kette von Abhängigkeiten zwischen den Objekten ist folgende (siehe hierzu auch Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*):

- Maschinen (`host`) und Maschinengruppen (`hostgroup`) die Sie definiert haben,
- Dienste (`service`). die an diese angebunden sind und möglicherweise Dienstegruppen (`servicegroup`) bilden,
- Befehle (`command`), die von diesen Diensten genutzt werden und
- Plugins, die über die Befehle angebunden sind.

Wenn Sie also beispielsweise den Namen eines Dienstes ändern, müssen Sie gegebenenfalls seine Referenz bei Dienstegruppen, Maschinen und Maschinengruppen anpassen. Wenn Sie einen Dienst entfernen, sollten Sie überprüfen, ob der von ihm genutzte Befehl noch von anderen Diensten genutzt wird, damit Sie ihn gegebenenfalls ebenfalls entfernen können. Die gleiche Prüfung sollten Sie dann auch noch für möglicherweise vorhandene Dienstegruppen durchführen. Des Weiteren müssen Sie möglicherweise verwendete Vorlagen daraufhin kontrollieren, ob sie überflüssig geworden sind.

Auf diese Weise können Änderungen sehr kompliziert werden und gleichzeitig speziell von den von Ihnen verwendeten Schemata für Namen sowie der von Ihnen gewählten Aufteilung in Vorlagen und Gruppen abhängen. Stellen Sie fest, welche Referenzen bestehen, welche Vorlagen genutzt werden, welche Gruppen betroffen sind und welche anderen Objekte eine Abhängigkeit aufweisen.

Diskussion

Als Hilfe für den Umgang mit derartigen Abhängigkeiten möchten wir Ihnen das Kommandozeilenprogramm grep empfehlen. Mit diesem können Sie Textdateien wie die Konfigurationsdateien nach dem Auftreten bestimmter Zeichenketten durchsuchen lassen und so Referenzen finden. Das Programm gibt Ihnen dabei normalerweise die Zeilen aus, in denen die gesuchte Zeichenkette enthalten ist. Sie können sich über entsprechende Parameter auch eine Anzahl von Zeilen vor und/oder nach der Trefferzeile ausgeben lassen, ganze Verzeichnisbäume oder nur bestimmte Dateien durchsuchen oder reguläre Ausdrücke für erweiterte Suchen nutzen.

Dieses einfache Programm kann sich als mächtiges Werkzeug bei der Suche in den Konfigurationsdateien erweisen. Über entsprechende reguläre Ausdrücke können Sie auch nach bestimmten Optionen suchen oder sich die jeweils darüber befindliche Zeile(n) ausgeben lassen. Sie können Aufrufe von grep durch die sogenannten Pipes auch miteinander verketten, um Ergebnisse des einen Aufrufes durch einen weiteren Aufruf zu verfeinern. Wir zeigen Ihnen im Folgenden anhand von Beispielen, wie mächtig dieses Werkzeug bei der Arbeit mit textbasierten Konfigurationsdateien sein kann.

grep benötigt zunächst immer eine Zeichenkette, nach der gesucht werden soll, und eine Angabe dazu, in welchen Dateien es suchen soll. Als einfaches Beispiel hier ein Aufruf, der nach der Zeichenkette check_procs in der Datei *localhost.cfg* sucht:

```
icinga@moni:/usr/local/icinga/etc/moni-conf.d$ grep check_procs localhost.cfg
 check_command check_procs!-C npcd -w 1:1 -c 1:1
```

Wenn Sie alle Dateien durchsuchen möchten, können Sie für die Spezifikation der Dateien auch den Platzhalter * verwenden:

```
icinga@moni:/usr/local/icinga/etc/moni-conf.d$ grep check_procs *
commands.cfg:# check_procs
commands.cfg: command_name check_procs
commands.cfg: command_line $USER1$/check_procs $ARG1$
ido2db_check_proc.cfg: command_line $USER1$/check_procs -w $ARG1$ -c $ARG2$ -C ido2db
localhost.cfg: check_command check_procs!-C npcd -w 1:1 -c 1:1
```

Einige weitere nützliche Parameter für grep sind folgende:

- `-E`

  Mit dieser Option schalten Sie die Verwendung von erweiterten regulären Ausdrücken frei. Dies entspricht dem Aufruf von egrep statt grep.

- `-r`

  Mit dieser Option aktivieren Sie die Rekursion über Verzeichnisse. Auf diese Weise können Sie auch gleichzeitig alle Unterordner eines Verzeichnisses bequem durchsuchen lassen.

- `-A` und/oder `-B` beziehungsweise `-C`

  Als Vorgabe zeigt Ihnen grep nur die Zeile an, in der die gesuchte Zeichenkette enthalten ist. Über diese Kontext-Optionen können Sie das Programm hingegen anweisen, auch eine Anzahl vorhergehender (`-B` für before) oder folgender (`-A` für after) Zeilen auszugeben. Mit der Option `-C` (für context) geben Sie an, dass die entsprechende Anzahl voranstehender und folgender Zeilen ausgegeben werden sollen. Fügen Sie der Option jeweils eine Zahl für die Anzahl der Zeilen hinzu, also etwa `-A 1` für jeweils eine folgende Zeile oder `-C 2` für jeweils 2 voranstehende und 2 nachfolgende Zeilen und so weiter.

Um alle Service-Namen zu finden, die das als `check_procs` eingebundene Plugin verwenden, können Sie sich beispielsweise wie folgt einen Aufruf zusammenbauen (hier nur für eine Datei und deshalb mit nur einem Ergebnis, um die Darstellung kurz zu halten):

1. Test der reinen Suche:

```
icinga@moni:/usr/local/icinga/etc/moni-conf.d$ grep check_procs localhost.cfg
 check_command check_procs!-C npcd -w 1:1 -c 1:1
```

2. Erweiterung der Ausgabe mit `-A` beziehungsweise `-B`, bis das gewünschte Datum (hier die Zeile mit `service_description`) enthalten ist. Dabei gehen wir hier davon aus, dass sich zwischen jeder Konfigurationszeile eine Leerzeile befindet:

```
icinga@moni:/usr/local/icinga/etc/moni-conf.d$ grep check_procs localhost.cfg -B 4
 service_description NPCD  use template-service-generic  check_command check_procs!-C
npcd -w 1:1 -c 1:1
```

3. Beschränkung der Ausgabe auf das gewünschte Datum durch erneuten Aufruf von *grep* über die Ergebnismenge:

```
icinga@moni:/usr/local/icinga/etc/moni-conf.d$ grep check_procs localhost.cfg -B 4
| grep service_description
 service_description NPCD
```

Die Erstellung von zielgenauen Abfragen wird dabei sehr erleichtert, wenn Ihre Konfigurationsdateien immer gleich aufgebaut sind. Wenn Sie dieses Beispiel anstatt nur für *localhost.cfg* für alle Dateien aufrufen, erhalten Sie alle Service-Namen, unter denen das Plugin verwendet wird.

Im Folgenden möchten wir Ihnen Beispiele für Lösungen häufig auftretender Fragestellungen an die Hand geben:

- **Ausfiltern von nicht-aktiven Konfigurationsdateien**: Wenn Sie temporär nicht benötigte Konfigurationsdateien durch Umbenennung der Dateiendung aus der Konfiguration entfernt haben, möchten Sie diese möglicherweise von der Suche ausnehmen. In diesem Falle können Sie anstelle des Platzhalters * als Angabe der Dateien `*.cfg` verwenden. Die Suche wird dann nur über Dateien erfolgen, die die Endung `cfg` aufweisen, also aktiv genutzt werden.

Datei-Ausschluss bei Rekursion: Sie können diese Möglichkeit allerdings nicht mit der rekursiven Suche kombinieren. Als Äquivalent für eine rekursive Suche mit einer solchen Einschränkung können Sie den Aufruf von `grep` jedoch wie folgt mit einem Aufruf von find verzahnen:

```
icinga@moni:/usr/local/icinga/etc$ find . -iname "*.cfg" -exec grep -B 4
check_procs \{\} \;
```

Dabei wird das Konstrukt `\{\}` durch den jeweils gefundenen Dateinamen ersetzt und an den *grep*-Befehl übergeben.

- **Ausfiltern von leeren Zeilen**: Wenn Sie bei der Ausgabe (oder etwa beim Abzählen der Zeilen) leere Zeilen stören, können Sie diese durch Anhängen von `grep -v -E '^$'` ausfiltern.
- **Ausfiltern von Kommentarzeilen**: Sie können auskommentierte Zeilen ausfiltern, indem Sie `grep -v -E '^#'` nutzen. Dabei werden in dem Ergebnis alle Kommentarzeilen (beginnend mit #) gesucht und nur die nicht zutreffenden Zeilen, also eben die nicht auskommentierten, zurückgegeben. Der hier von uns vorgeschlagene, einfache reguläre Ausdruck geht dabei allerdings davon aus, dass sich das Kommentarzeichen am Anfang der Zeile befindet.

Die verschiedenen Filter können Sie miteinander kombinieren, indem Sie diese mit dem Pipe-Operator "|" verknüpfen (dabei geben Sie einen Dateinamen nur beim ersten Aufruf an, damit die weiteren Aufrufe auf die Ausgabe des vorherigen zugreifen). Um also etwa zunächst alle Kommentarzeichen zu ignorieren, dann eine Filterung aus dem vorherigen Beispiel vorzunehmen und anschließend alle Leerzeichen auszufiltern, schreiben Sie Folgendes:

```
icinga@moni:/usr/local/icinga/etc/moni-conf.d$ grep -v -E '^#' localhost.cfg
| grep check_procs -B 4 | grep -v -E '^$'
```

Durch passende Aneinanderreihung von Filtern können Sie mit dem einfachen Werkzeug *grep* so auch sehr komplexe Filterungen vornehmen. Sofern Ihre Konfigurationsdateien bezüglich der Struktur und der Formatierung gleich angelegt sind, können Sie auf diese Weise auch in sehr umfangreichen Umgebungen effizient suchen.

Automatisierte Änderungen: Falls Sie sich bei der Arbeit mit diesen Werkzeugen nun wünschen, im gleichen Zuge Änderungen in Form von automatisierten Ersetzungen vornehmen zu können, so seien Sie versichert, dass auch dies möglich ist. Wir müssen Sie an dieser Stelle allerdings auf externe Quellen zu den entsprechend empfehlenswerten Werkzeugen `sed` und `awk` verweisen, da deren Umfang den Rahmen dieses Buches sprengen würde.

Siehe auch

- Rezept zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Hilfeseiten des Kommandos `grep`: man grep
- Hilfeseiten des Kommandos `find`: man find

3.8 Prüfen der Konfigurationsdateien auf Fehler

Problem

Sie möchten sicherstellen, dass die Konfigurationsdateien (siehe Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*) nach etwaigen Änderungen korrekt sind beziehungsweise herausfinden, wo syntaktische Fehler vorliegen.

Lösung

Zunächst einmal müssen Sie sich keine Sorgen darüber machen, dass Ihr Nagios/Icinga-System nach Änderungen mit syntaktischen Konfigurationsfehlern einfach stoppen und Prüfungen abbrechen wird. Bei einem zum Einlesen der Änderungen notwendigen Neustart überprüft Ihr Monitoring-System automatisch, ob die Konfigurationsdateien syntaktische Fehler enthalten und verweigert gegebenenfalls den Neustart. Bei einem Neustart der ganzen Maschine würden solche Fehler dann aber dazu führen, dass das Monitoring-System schlicht nicht startet, weshalb die folgenden Hinweise für Sie wichtig sind.

Die Prüfung der Konfigurationsdateien beim Start entspricht einem Aufruf von Nagios/Icinga mit der Option `-v /pfad/konfigurationsdatei`, die Sie auch manuell veranlassen können:

```
root@moni:~# /usr/local/icinga/bin/icinga -v /usr/local/icinga/etc/icinga.cfg
[...]
Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
```

Die Pfade zum Programm und zu der Konfigurationsdatei müssen Sie dabei allerdings gegebenenfalls an Ihre Umgebung anpassen. Sofern die Konfigurationsdateien syntaktische Fehler enthalten, wird Ihnen dies dann mitgeteilt. Bei der bei einem Neustart durchgeführten Prüfung werden die erzeugten Ausgaben in eine Datei geschrieben, die Ihnen im Falle von aufgetretenen Fehlern mitgeteilt wird. Die folgenden Zeilen zeigen Ihnen ein entsprechendes Beispiel:

```
root@moni:~# service icinga restart
Running configuration check...CONFIG ERROR! Restart aborted. See
/usr/local/icinga/var/icinga.chk for details.
```

Sie müssen diese Datei (bei unserer Referenzinstallation */usr/local/icinga/var/icinga.chk*) dann jedoch manuell einsehen, weshalb sich zumindest bei größeren Änderungen durchaus der manuelle Aufruf der Prüfung anbietet. Dieser ist dann auch noch nicht mit einem Neustart verbunden.

Diskussion

Der mitgelieferte und automatisch durchgeführte Test ist sehr hilfreich. Sie erfahren hier bei einem Fehler die betroffene Datei und die jeweilige Zeilennummer, in der der Fehler festgestellt wurde. Auf diese Weise können Sie schnell mit einem Editor Ihrer Wahl an die fehlerhaften Stelle springen und den Fehler manuell korrigieren.

Beachten Sie dabei jedoch, dass Ihr Monitoring-System Sie jeweils nur auf den ersten gefundenen Fehler hinweist, da die Prüfung danach abgebrochen wird. Bei mehreren Fehlern werden Sie diese Schritte also gegebenenfalls wiederholen müssen.

Regelmäßige Durchführung: Führen Sie den Test der Konfigurationsdateien bei umfangreichen Änderungen regelmäßig durch, damit Sie Fehler jeweils an der zuletzt geänderten Stelle finden und diese nicht suchen müssen. Wenn Sie die Konfiguration an vielen Stellen ändern, werden Sie anderenfalls mehr Zeit benötigen, um die fraglichen Stellen herauszusuchen und zu korrigieren.

Wenn Ihre Installation sehr umfangreich ist und mehrere Kollegen daran mitarbeiten, können Sie auch ein Plugin einbinden, dass diesen Test regelmäßig automatisiert ausführt. So können Sie gegebenenfalls nachvollziehen, seit wann die Konfiguration Fehler aufweist. Dies kann Ihnen dann unter Umständen helfen, festzustellen, wie Sie den Fehler auf schnelle Weise beheben können, indem Sie identifizieren wann (und dadurch wer) die entsprechende Änderung vorgenommen hat. In Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* zeigen wir Ihnen, wie Sie Befehlsausgaben in das Format für Nagios/Icinga (siehe hierzu Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*) umwandeln können. Entsprechend könnten Sie auch den betreffenden Befehlsaufruf einbinden.

Siehe auch

- Kapitel zu den Konfigurationsobjekten von Nagios/Icinga: Kapitel 4, *Die Konfigurationsobjekte von Nagios/Icinga*
- Rezept zur Erstellung eines Wrapper-Scripts: Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*
- Rezept zur Einführung in die von Nagios/Icinga verwendete API: Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*

3.9 Sicherungen der Konfigurationsdateien anlegen

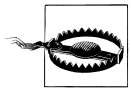
Problem

Sie möchten die relevanten Teile Ihrer Monitoring-Konfiguration sichern.

Lösung

Wir zeigen Ihnen im Folgenden am Beispiel unserer Referenzinstallation auf, welche Teile Sie bei einer Datensicherung berücksichtigen sollten. Grundsätzlich unterscheiden Sie dabei zwischen Konfigurationsdateien und den eigentlichen Daten. Manchmal sind die historischen Daten nicht wirklich wichtig und gleichzeitig relativ umfangreich, so dass Sie eventuell nur die Konfigurationsdateien sichern möchten. Sofern Sie das Backup für eine Aktualisierung oder eine Migration durchführen, müssen Sie auf die Übernahme der Altdaten unter Umständen sogar verzichten oder sonst manuelle (und aufwendige) Anpassungen beispielsweise an den Datenbanken vornehmen.

Wir gehen im Folgenden deshalb zunächst nur auf die Sicherung der Konfigurationsdateien und Plugins ein, nicht jedoch auf die dazu gespeicherten Daten. Diese betrachten wir im Rahmen der Diskussion noch einmal genauer und erläutern dazu, wann und warum diese wichtig sein können.



RESTORE – ACHTUNG: Sie sollten ein wie hier beschriebenes Backup NIEMALS ungeprüft einspielen. Gehen Sie wie folgt vor:

1. Beenden Sie Icinga / Nagios (beispielsweise mit `service icinga stop`).
2. Beenden Sie gegebenenfalls ido2db / ndo2db, falls Sie die IDOUtils / NDOUtils bereits installiert haben (beispielsweise mit `service ido2db stop`).
3. Entpacken Sie das Backup in ein separates Verzeichnis.
4. Suchen Sie die benötigten Dateien und Verzeichnisse heraus.
5. Sichern Sie jede zu überschreibende Konfigurationsdatei (beispielsweise mit dem Kommando `mv DATEINAME DATEINAME.ORIG`).
6. Kopieren Sie Dateien aus dem Backup schrittweise an die erforderliche Stelle oder nehmen Sie die benötigte Anpassungen manuell an den Originaldateien vor.

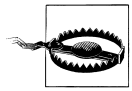
Sicherung der Konfigurationsdateien

Für ein wie in Rezept 1.4, *Icinga aus den Quellen installieren* beschrieben installiertes Icinga sind die relevanten Verzeichnisse zunächst folgende:

- Konfigurationsdateien für Icinga:
  /usr/local/icinga/etc
- Nagios-Plugins:
  /usr/local/icinga/libexec
- Zusätzliche Konfigurationsdatei für IDOUtils:
  /usr/local/icinga/modules/idoutils.cfg

Dabei enthalten sind dann die Konfigurationsdateien von IDOUtils, sofern Sie dieses installiert haben. Wenn Sie für Konfigurationsdateien und/oder Plugins weitere, nicht in unserem Beispiel enthaltene Verzeichnisse eingebunden haben, sollten Sie diese hier berücksichtigen und der Liste hinzufügen.

Dies gilt in jedem Falle für die entsprechenden Konfigurationsdateien für die Oberflächen und entsprechende Dateien beispielsweise für den Webserver. Wir empfehlen Ihnen, hier das gesamte Verzeichnis */etc* zu berücksichtigen, wobei dann automatisch auch die Konfigurationen für die Oberflächen, den Webserver und auch etwa für SNMP enthalten sind.



Referenzinstallation: Wenn Ihre Installation von der hier dargestellten Referenzinstallation abweicht, müssen Sie die Pfade vermutlich anpassen. Ermitteln Sie dafür jeweils den bei Ihnen gültigen Pfad zu den Daten und nutzen Sie diese anstatt der hier angegebenen. Für die in Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten* beschriebenen Installationsarten haben wir für Sie am Ende der jeweiligen Rezepte Tabellen mit den Pfaden zu den wichtigsten Konfigurationsdateien zusammengestellt.

Wenn Sie bei der Installation auch PNP4Nagios installiert haben, sollten Sie außerdem die folgenden Verzeichnisse sichern:

- Konfigurationsdateien für PNP4Nagios:
  /usr/local/pnp4nagios/etc
- Erstellte und/oder angepasste Vorlagen für Graphen:
  /usr/local/pnp4nagios/share/templates

Wenn Sie wie beschrieben auch NagVis installiert haben, sollten Sie zusätzlich das entsprechende Verzeichnis sichern:

- Konfigurationsdateien für NagVis (inklusive erstellter Karten):
  /usr/local/nagvis/etc

Sichern Sie von Ihnen verwendete Verzeichnisse und Sie können im Zweifel durch Neuinstallation und manuelles Einspielen der erforderlichen Dateien aus dem Backup schnell wieder ein ohne Abstriche laufendes Monitoring-System aufbauen.

Diskussion

Für den Fall, dass Sie keine Backup-Lösung im Einsatz haben, in die Sie diese Verzeichnisse einbinden, können Sie das im Folgenden vorgestellte, einfache Script verwenden.

Beispiel eines Scripts zur Sicherung von Dateien und Verzeichnissen

Das folgende Script packt die spezifizierten Dateien/Verzeichnisse mit Hilfe von tar in ein mit gzip komprimiertes Archiv:

```
#!/bin/bash
# Einfaches Backup-Script
# benötigt: tar und gzip

# Backupverzeichnis
BACKUP_PATH="/backup_icinga"
```

```
# Anzulegende Backupdatei
BACKUP_FILE="$BACKUP_PATH/`date +%Y-%m-%d--%H-%M`_backup_monitoring.tar"

# An dieser Stelle gegebenenfalls den Teil zur Datenbank-Sicherung einfügen

# Definition zu sichernder Dateien und Verzeichnisse
ARR_DIRS=( /etc /usr/local/icinga/etc /usr/local/icinga/libexec /usr/local/pnp4nagios/etc
 /usr/local/pnp4nagios/share/templates /usr/local/nagvis/etc )

# --------

# Verzeichnis anlegen, falls es nicht existiert
mkdir -p $BACKUP_PATH

# jede/n der Dateien/Ordner zu einem TAR-Archiv hinzufügen
for FILE in "${ARR_DIRS[@]}"
do
  tar rf $BACKUP_FILE $FILE 2> /dev/null
done

# Das Tar-Archiv komprimieren
gzip $BACKUP_FILE
```

Jedes Mal, wenn Sie dieses Script ausführen, wird der jeweilige Stand Ihrer Konfiguration in einer einzelnen Datei zusammengefasst. Sofern Sie die Plugins hier ebenfalls sichern, wird das komprimierte Archiv schnell einige Megabyte groß werden, ansonsten umfasst es meist nur einige Kilobyte.

Regelmäßige Ausführung mit cron

Sofern Sie das gerade angeführte Script automatisch und regelmäßig ausführen lassen wollen, können Sie dies vom Dienst cron erledigen lassen, der standardmäßig auf allen Distributionen läuft. Speichern Sie das Script dafür zunächst in einer Datei, die ausführbar sein muss. Wir weisen ihr hier den Dateinamen *backup.sh* zu und legen sie im Ordner */root/bin* ab:

```
root@buckap:~/bin# chmod +x backup.sh
```

Editieren Sie dann die von `cron` für den aktuellen Benutzer durchzuführenden Aufgaben mit dem Befehl `crontab -e` und fügen Sie eine Zeile wie die folgende an:

```
[...]
# m h dom mon dow command
0 3 * * * /root/bin/backup.sh
```

Die Datei wird nun jeden Tag um drei Uhr am Morgen ausgeführt werden. Sie können hier abweichende Intervalle wählen (siehe auch `man cron`).

Berücksichtigung historischer Daten (und von Datenbanken)

Historischen Daten, also die Entwicklung der Daten über die Zeit, kann eine hohe Bedeutung zukommen, obwohl sie für die eigentliche Funktion der Überwachung nicht

direkt relevant sind. Dies ist gerade dann der Fall, wenn es um Probleme geht, die sich über einen bestimmten Zeitraum entwickeln, beispielsweise bei Speicherüberläufen oder bei der Analyse einer sich anbahnenden Überlastung von Bandbreiten. Möglicherweise setzen Sie aber auch ein Berichtswesen (Reporting) ein, das auf diesen Daten basiert, oder Sie verwenden diese Daten sogar als Entscheidungsgrundlage. Im Folgenden zeigen wir Ihnen, welche zusätzlichen Daten relevant sind und wie Sie diese in Ihre Sicherung einbinden können.

Die Laufzeitdaten (also jeweils den aktuellen Status der Maschinen und Dienste sowie über die Weboberfläche hinterlegte Kommentare) speichern Nagios/Icinga zunächst in einer einzelnen Datei. Sofern Sie über NDOUtils/IDOUtils eine Datenbank eingebunden haben, möchten Sie diese eventuell ebenfalls sichern. Und wenn Sie einen Grapher wie PNP4Nagios einbinden, kommen zusätzlich die von diesem Grapher genutzten RRD-Datenbanken hinzu. Zusätzlich zu sichern ist damit (bei Installation nach Rezept 1.4, *Icinga aus den Quellen installieren*):

- Retention-Datei

  Dies ist eine einzelne Datei unter */usr/local/icinga/var/retention.dat*.
- NDOUtils/IDOUtils Datenbank

  Um eine Datenbank zu sichern, müssen Sie zunächst eine Datei mit den entsprechenden Inhalten (einen sogenannten Datenbank-Dump) erstellen. Dafür werden Sie die Zugangsdaten benötigen, die Sie bei der Installation angegeben haben (Erläuterungen zur Anfertigung der Sicherung finden Sie nachfolgend in diesem Kapitel).
- RRD-Datenbanken von PNP4Nagios

  Die Datenbanken und die dazugehörigen Beschreibungsdateien finden Sie im Verzeichnis unter */usr/local/pnp4nagios/var,* das Sie entsprechend komplett mitsichern können.

Wenn Sie das oben vorgestellte Script für die Sicherungen verwenden möchten, müssen Sie dieses für die Sicherung einer Datenbank zunächst erweitern. Die folgenden Zeilen erstellen mit Hilfe der Kommandozeilen-Anwendung `mysql` eineKopie einer kompletten Datenbank in Form einer einzelnen Datei. Anschließend wird die erstellte Datei komprimiert. Passen Sie die Daten bitte an Ihre Umgebung an und fügen Sie diese Zeilen unter der angegebenen Kommentarzeile in das obige Script ein:

```
[...]
# An dieser Stelle gegebenenfalls den Teil zur Datenbank-Sicherung einfügen
# benötigt mysql-cli

DBBACKUP_FILE="$BACKUP_PATH/`date +%Y-%m-%d--%H-%M`_backup_monitoring_db.sql"
USE_DBHOST=10.85.58.42
USE_DBPORT=3306
USE_DBNAME=moni_icinga
USE_DBUSER=sql_buckap
USE_DBPASS=PASSWORT
```

```
mysqldump -u $USE_DBUSER -p$USE_DBPASS -h $USE_DBHOST -P $USE_DBPORT $USE_DBNAME
--single-transaction > $DBBACKUP_FILE

gzip $DBBACKUP_FILE

[...]
```

Des Weiteren müssen Sie noch den in `$DBBACKUP_FILE` gespeicherten Dateinamen des Datenbank-Backups sowie die anderen zusätzlichen Dateien und Verzeichnisse für die Laufzeitdaten in die Zeile mit den zu sichernden Dateien aufnehmen, für die genannten Dateien und Verzeichnisse also etwa:

```
[...]
# Definition zu sichernder Dateien und Verzeichnisse
ARR_DIRS=( /etc /usr/local/icinga/etc /usr/local/icinga/libexec /usr/local/pnp4nagios/etc
 /usr/local/pnp4nagios/share/templates /usr/local/nagvis/etc $DBBACKUP_FILE
 /usr/local/icinga/var/retention.dat /usr/local/pnp4nagios/var )
[...]
```

Damit werden nun Ihre Datenbank und die Laufzeitdaten mitgesichert. Streng genommen wäre es bei der Datenbank ausreichend, lediglich einen Teil der Datenbank-Tabellen zu sichern. Bei der hier von uns vorgeschlagenen Art der Sicherung ist die Handhabung der Sicherung und der Wiederherstellungsoperationen allerdings deutlich einfacher, weshalb wir Ihnen die Sicherung der kompletten Datenbank empfehlen.

Weitere Sicherungen: Je nachdem, wie sehr Sie Ihre Installation angepasst haben, kann es nötig werden, weitere Daten zu sichern. Dies kann beispielsweise der Fall sein, wenn Sie Änderungen an den CGI- oder PHP-Scripten vornehmen, um die Oberfläche an Ihre Wünsche anzupassen, oder wenn Sie ein zusätzliches Plugin-Verzeichnis einbinden. Sie sollten bei jeder Änderung überprüfen, ob diese nur bereits gesicherte Daten betrifft oder ob Sie weitere Dateien und/oder Datenbanken in Ihre Sicherung einbinden müssen.

Siehe auch

- Installationen unter Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten* – insbesondere die hier besprochene Referenzinstallation Rezept 1.4, *Icinga aus den Quellen installieren*
- Hilfeseiten der hier verwendeten Befehle: `man bash`, `man tar`, `man gzip`, `man cron` und `man mysql`

3.10 Fehlern mit Debug- und Logdateien nachgehen

Problem

Sie haben ein Problem mit Nagios/Icinga beziehungsweise einem der Addons, dem Sie nachgehen möchten.

Lösung

Wir zeigen Ihnen im Folgenden für Nagios/Icinga und die einzelnen Addons, wo die relevanten Protokolldateien konfiguriert und abgelegt sind. Wir gehen dabei von unserem Referenzsystem aus (siehe Rezept 1.4, *Icinga aus den Quellen installieren*). Sofern Sie eine abweichende Installation verwenden, müssen Sie eventuell die betreffenden Pfade anpassen.

Performance: Beachten Sie bitte, dass die im folgenden vorgestellten Protokollierungen unter Umständen erhebliche Ressourcen erfordern. Sie sollten sie deshalb unbedingt wieder ausschalten, sobald Sie den fraglichen Fehler gefunden und korrigiert sowie entsprechende Tests durchgeführt haben.

Protokolldateien Nagios/Icinga

Die Hauptkonfigurationsdatei für Nagios/Icinga gibt vor, was auf welche Weise protokolliert wird. Bei einer Installation aus Quellcode findet sich diese standardmäßig unter */usr/local/nagios/nagios.cfg* beziehungsweise */usr/local/icinga/etc/icinga.cfg*. Hierin sind für die Protokollierung zur Laufzeit die folgenden Optionen relevant (an dieser Stelle mit den Standardwerten aus der Icinga-Installation):

```
log_file=/usr/local/icinga/var/icinga.log
log_rotation_method=d
log_archive_path=/usr/local/icinga/var/archives
use_daemon_log=1
use_syslog=1
use_syslog_local_facility=0
syslog_local_facility=5
log_notifications=1
log_service_retries=1
log_host_retries=1
log_event_handlers=1
log_initial_states=0
log_external_commands=1
log_passive_checks=1
```

Hier können Sie anpassen, was fortlaufend wohin protokolliert werden soll. Normalerweise sollten Sie diese Optionen aber alle bei ihren Standardwerten belassen können. Dabei werden die Protokolle dann zusätzlich in das Syslog geschrieben, das Sie typischerweise unter */var/log/syslog* finden.

Bei der Fehlersuche reichen diese Informationen aber häufig nicht aus. Die folgenden Optionen in der gleichen Hauptkonfigurationsdatei legen fest, ob und wie eine zusätzliche Fehlerprotokollierung (englisch auch debug-log genannt) durchgeführt wird:

```
# DEBUG LEVEL
# This option determines how much (if any) debugging information will
# be written to the debug file. OR values together to log multiple
# types of information.
# Values:
# -1 = Everything
# 0 = Nothing
# 1 = Functions
# 2 = Configuration
# 4 = Process information
# 8 = Scheduled events
# 16 = Host/service checks
# 32 = Notifications
# 64 = Event broker
# 128 = External commands
# 256 = Commands
# 512 = Scheduled downtime
# 1024 = Comments
# 2048 = Macros
debug_level=0

# DEBUG VERBOSITY
# This option determines how verbose the debug log out will be.
# Values: 0 = Brief output
# 1 = More detailed
# 2 = Very detailed
debug_verbosity=1

# DEBUG FILE
# This option determines where Nagios should write debugging information.
debug_file=/usr/local/icinga/var/icinga.debug

max_debug_file_size=1000000
```

Legen Sie bei der Fehlersuche debug_verbosity auf 2 (sehr detailierte Ausgabe) und debug_level auf -1 (alles) oder einen entsprechend angepassten Wert fest. Um nur bestimmte Werte zu protokollieren, können Sie die jeweiligen Zahlenwerte addieren und die Summe als Wert verwenden.

Protokolldateien Icinga-Web

Die Weboberfläche Icinga-Web verfügt über eine eingebaute Anzeigefunktion für die Protokolldateien. Nutzen Sie diese, sofern die Oberfläche noch läuft und es sich um kleinere Probleme handelt. Ansonsten werden die entsprechenden Daten auch im Dateisystem abgelegt. Wo genau das ist, wurde bei unserer Referenzinstallation in der Datei */usr/local/icinga-web/app/config/logging.xml* festgelegt; in unserem Fall wurde das Verzeichnis */var/log/icinga-web/* angelegt und genutzt.

Protokolldateien NDOUtils/IDOUtils

Die Datenanbindung über NDOUtils/IDOUtils hat vergleichbar mit Nagios/Icinga eine zentrale Konfigurationsdatei. Bei der Installation aus Quellcode liegt diese standardmäßig unter */usr/local/nagios/ndo2db.cfg* beziehungsweise */usr/local/icinga/ido2db.cfg*. Die für eine Fehlerprotokollierung generell wichtigen Optionen sind folgende:

```
# DEBUG LEVEL
# This option determines how much (if any) debugging information will
# be written to the debug file. OR values together to log multiple
# types of information.
# Values: -1 = Everything
# 0 = Nothing
# 1 = Process info
# 2 = SQL queries
debug_level=0

# DEBUG VERBOSITY
# This option determines how verbose the debug log out will be.
# Values: 0 = Brief output
# 1 = More detailed
# 2 = Very detailed
debug_verbosity=1

# DEBUG FILE
# This option determines where the daemon should write debugging information.
debug_file=/usr/local/icinga/var/ido2db.debug

# 100M
max_debug_file_size=100000000
```

Die Fehlerprotokollierung ist standardmäßig deaktiviert. Um sie zu aktivieren, setzen Sie `debug_level` auf -1 oder einen angepassten Wert und `debug_verbosity` auf 2.

Protokolldateien PNP4Nagios

PNP4Nagios legt von sich aus zunächst keine Protokolle an. Die Optionen hierzu finden Sie in der Konfigurationsdatei *./process_perfdata.cfg* im PNP4Nagios-Konfigurationsverzeichnis, das sich bei unserer Referenzinstallation unter */usr/local/pnp4nagios/etc/* befindet. Die relevanten Optionen sind hier diese:

```
#
# name of the log file
#
LOG_FILE = /usr/local/pnp4nagios/var/perfdata.log

#
# Loglevel 0=silent 1=normal 2=debug
#
LOG_LEVEL = 0
```

Erhöhen Sie gegebenenfalls die Protokollierungsstufe auf 1 oder 2 und überprüfen Sie dann die Ausgabe in der Logdatei `/usr/local/pnp4nagios/var/perfdata.log`.

Diskussion

Sowohl Nagios/Icinga als auch die erläuterten Addons ermöglichen Ihnen gegebenenfalls nachzuvollziehen, was vor sich geht. Es scheint uns an dieser Stelle noch einmal angebracht, auf unsere eingangs angeführte Warnung hinzuweisen: Bei hohen Protokollierungsleveln wird dann jeder durchgeführte Test nicht nur kurz angeführt, sondern es entsteht eine Flut von Einträgen. Dabei werden die einzelnen Verarbeitungsschritte bis hin zur exakten SQL-Anweisung, mit der die Daten in die Datenbank geschrieben werden, dargestellt. Verwenden Sie diese Funktionen also mit Bedacht.

Siehe auch

- Rezepte zur Installation von Nagios und Icinga aus den Quellen: Rezept 1.9, *Nagios aus den Quellen installieren* und Rezept 1.4, *Icinga aus den Quellen installieren*

3.11 Maßgeschneiderte Benachrichtigungen einrichten

Problem

Sie möchten Benachrichtigungen in Ihrer Art oder ihrer Frequenz an Ihre Bedürfnisse anpassen.

Lösung

Über die unterschiedlichen Optionen bei der Konfiguration von Kontakten (siehe Rezept 4.4, *Kontakte definieren (contact)*), Maschinen (siehe Rezept 4.1, *Maschinen einbinden (host)*) und Services (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) haben sie zahlreiche Möglichkeiten, die Benachrichtigungen an Ihre Wünsche anzupassen.

Referenzinstallation: Wir gehen in der Folge wieder von unserer Referenzinstallation aus, die wir in Rezept 1.4, *Icinga aus den Quellen installieren* beschrieben haben. Beachten Sie, dass etwa die bei der Installation vorgegebenen Namen der Kontakte in Nagios anders lauten und die Ablageorte bei anderen Installationsarten variieren können.

Eingrenzung des Empfängerkreises

An welche Kontakte Benachrichtigungen versendet werden, legen Sie zunächst über die Kontaktdefinition jeder einzelnen Maschine fest. Sie können hier Kontakte und/oder Kontaktgruppen angeben. In den bei der Installation erstellen Vorlagen wird bei allen Maschinen und Services die Kontaktgruppe `admins` referenziert, die wiederum aus dem Kontakt `icinga-admin` besteht.

Erwägen Sie hier Kontaktgruppen (siehe Rezept 4.8, *Kontakte in Gruppen zusammenfassen (contact-group)*) zu verwenden, um später gegebenenfalls leichter Änderungen, wie das Hinzufügen oder Entfernen weiterer Benutzer, durchführen zu können. Legen Sie sich dazu für jeden Empfängerkreis, sozusagen jede Rolle in Ihrer organisatorischen Struktur, eine Kontaktgruppe an und referenzieren Sie diese bei allen Maschinen und Services, für die sie Benachrichtigungen erhalten soll.

Eingrenzung auf bestimmte Ereignisse

Über die Konfigurationsoption notification_options bei Maschinen und Diensten sowie bei Kontakten können Sie gezielt angeben, bei welchen Ereignissen Benachrichtigungen versendet werden sollen. Die Bedeutung der Status haben wir Ihnen in Tabelle 3-2 zusammengestellt.

Tabelle 3-2: Statuscodes für Maschinen und Services

Status	bei Maschinen	bei Services
OK	Gerät läuft	Service läuft
n (none)	Status unbekannt	Status unbekannt
d (down)	Gerät läuft nicht	---
w (warning)	---	Warnschwelle überschritten
c (critical)	---	kritische Schwelle überschritten
u (unreachable)	Gerät nicht erreichbar	Gerät nicht erreichbar
r (recover)	Wechsel von d, u zu OK	Wechsel von w, c, u zu OK
f (flapping)	Zustand wechselt häufig (Flattern)	Zustand wechselt häufig (Flattern)
s (scheduled)	konfiguriertes Wartungsintervall	konfiguriertes Wartungsintervall

Wenn Sie beispielsweise Benachrichtigungen über ein Flattern (siehe Rezept 3.12, *Den Flattern-Status verstehen und nutzen*) zunächst nur innerhalb der Netzwerk-Administratoren, eine fehlende Erreichbarkeit aber auch den Endanwendern oder Kunden mitteilen möchten, können Sie dies über entsprechende Werte für Kontakte erreichen (hier in Form einer Vorlage für Kontakte):

```
define contact {
       register                        0
       contact_name                    template-administratoren
       host_notification_options       d,u,r,f,s,n
       service_notification_options    w,c,u,r,f,s
}

define contact {
       register                        0
       contact_name                    template-kunden
       host_notification_options       d,u,r,s,n
       service_notification_options    w,c,u,r,s
}
```

Beachten Sie, dass der Buchstabe f für den Zustand Flapping bei der Kontaktgruppe kunden im Gegensatz zur Kontaktgruppe administratoren fehlt. Andersherum möchten Sie eine fehlende Erreichbarkeit (also Maschinen, deren Status nicht bekannt ist, weil etwa der entsprechende Router ausgefallen ist) vielleicht nicht an die Administratoren melden, weil diesen die entsprechenden Abhängigkeiten klar sind, aber die Endnutzer oder Kunden dennoch darüber informieren. Auch dies können Sie über entsprechende Werte dieser Option erreichen – entfernen Sie im eben angeführten Beispiel bei den Administratoren einfach den entsprechenden Buchstaben u Bei den Maschinen-Benachrichtungen (host_notification_options).

Eingrenzung auf bestimmte Zeiten

Über die Zeitfenster können Sie bei Kontakten, Maschinen und Services angeben, wann Benachrichtigungen versendet werden sollen. Wie Sie Zeitfenster definieren, beschreiben wir in Rezept 4.5, *Zeitperioden definieren (timeperiod)*. Sobald Sie entsprechende Zeitfenster erstellt haben, können Sie diese über die Optionen host_notification_period beziehungsweise service_notification_period bei Kontakten oder notification_period bei den Maschinen oder Diensten referenzieren.

Anpassung der eigentlichen Benachrichtigung

Was genau eine Benachrichtigung enthält und wie sie versendet wird, ist über eine entsprechende Befehlsdefinition, also den Objekttyp command (siehe Rezept 4.3, *Befehle hinzufügen (command)*), festgelegt. Bei der Installation wurden hierzu die Befehle notify-host-by-email und notify-service-by-email eingerichtet:

```
# 'notify-host-by-email' command definition
define command{
       command_name notify-host-by-email
       command_line /usr/bin/printf "%b" "***** Icinga *****\n\nNotification Type: \
                    $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress: \
                    $HOSTADDRESS$\nInfo: \$HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" \
                    | /usr/bin/mail -s "** $NOTIFICATIONTYPE$ \
                    Host Alert: $HOSTNAME$ is $HOSTSTATE$ **" $CONTACTEMAIL$
}

# 'notify-service-by-email' command definition
define command{
       command_name notify-service-by-email
       command_line /usr/bin/printf "%b" "***** Icinga *****\n\nNotification Type: \
                    $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: \
                    $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState: \
                    $SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional \
                    Info:\n\n$SERVICEOUTPUT$\n" | /usr/bin/mail -s \
                    "** $NOTIFICATIONTYPE$ Service Alert: $HOSTALIAS$/$SERVICEDESC$ is \
                    $SERVICESTATE$ **" $CONTACTEMAIL$
}
```

Beachten Sie, dass die Definitionen jeweils lediglich aus zwei Zeilen bestehen, von denen die zweite ziemlich lang ist. Es handelt sich dabei um eine Kommandozeile, bei der der Befehl printf aufgerufen wird. Das Ergebnis von `printf` wird dann an den Befehl mail weitergeleitet. Bei den vielleicht zunächst etwas verwirrenden `$NAME$`-Angaben handelt es sich um Makros (siehe Rezept 4.3, *Befehle hinzufügen (command)*), die vor dem eigentlichen Ausführen durch Nagios/Icinga gegen entsprechende Werte ausgetauscht werden – und damit den Inhalt liefern, den Sie am Ende in der E-Mail lesen können.

Analog zu dem hier für das Standardprogramm `mail` konfigurierten Aufruf können Sie beliebige andere Programme aufrufen und über Makros verfügbare Inhalte übergeben. Über die sogenannten $USERn$-Makros (siehe Rezept 4.3, *Befehle hinzufügen (command)*) sowie Text in der Kommandozeile können Sie hier Inhalte hinzufügen. Damit können Sie jede Benachrichtigung nutzen, die Sie über die Kommandozeile aufrufen können, indem Sie den entsprechenden Aufruf als Befehls-Definition definieren.

Auf solche Weise definierte Befehle können Sie dann bei den Kontakt-Definitionen über die Optionen service_notification_commands und `host_notification_commands` einbinden. Da Sie dabei jeweils eine durch Kommata separierte Liste von Befehlen übergeben können, können Sie auch mehrere Benachrichtigungsbefehle nacheinander ausführen lassen.

Beispiele: Bei Icinga werden in der Datei `notifications.cfg` auch Beispiele für die Benachrichtigung via Email, SMS, Jabber und twitter ausgeliefert.

Diskussion

Die hier noch einmal zusammengefassten Optionen erlauben Ihnen, die Art und Häufigkeit der Benachrichtigungen in vielerlei Hinsicht anzupassen. Beachten Sie gegebenenfalls die Hinweise unter Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien* für Strukturierungsmöglichkeiten bei der Ablage sowie Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System* zur Nutzung von Vorlagen.

Vorrang: Beachten Sie, dass Meldungen, die aufgrund gesetzter Optionen nicht generiert werden, auch nicht an einen Kontakt weitergeleitet werden können. Sollten Sie also gewünschte Benachrichtigungen wider Erwarten nicht erhalten, dann kontrollieren Sie die entsprechenden Optionen der Maschinen beziehungsweise Services.

Wenn das Monitoring-System von vielen Benutzern genutzt wird, sollten Sie wie beschrieben eine Kontaktgruppe nutzen, um Rollen abzubilden. Sie können auf solche Weise erstellte Kontaktgruppen dann weiter als Benutzer für die Weboberfläche freischalten (siehe hierzu Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*). Kontaktgruppen können Sie allerdings in den hier betrachteten Versionen nur bei Icinga verwenden.

Siehe auch

- Rezepte zur Einbindung von Kontakten und Kontaktgruppen: Rezept 4.4, *Kontakte definieren (contact)* und Rezept 4.8, *Kontakte in Gruppen zusammenfassen (contactgroup)*
- Rezept zum Flattern-Status: Rezept 3.12, *Den Flattern-Status verstehen und nutzen*
- Rezepte zu Maschinen und Dienstdefinitionen: Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Definition von Zeitfenstern: Rezept 4.5, *Zeitperioden definieren (timeperiod)*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zur Vereinfachung der Konfiguration mit Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Rezept zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*

3.12 Den Flattern-Status verstehen und nutzen

Problem

Sie möchten den Flattern-Status verstehen und nutzen.

Lösung

Wir beschreiben Ihnen im Folgenden das Konzept für die Erkennung von häufig den Status wechselnden Maschinen und Diensten (die sogenannte Flattern-Erkennung). Dazu zeigen wir Ihnen, wie Sie diese aktivieren und den davon abgeleiteten Status flapping sinnvoll nutzen können.

Die zugrundeliegende Idee ist zunächst einfach, nämlich besonders häufig und möglicherweise unnötig auftretende Benachrichtigungen zu unterdrücken. Genau dieses Szenario tritt bei einem Service auf, der ständig zwischen `OK` und `CRITICAL` wechselt, wenn bei jedem Wechsel eine neue Benachrichtigung (siehe auch Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*) generiert wird. Nagios/Icinga kann versuchen, dies zu erkennen und setzt die Maschine oder den Dienst dann in den Status `flapping`.

Anstatt also häufige Meldungen zu den Status *OK* und *CRITICAL* zu versenden, würde das System dann optimalerweise nur noch eine Nachricht zu diesem neuen Status schicken. Außerdem erhalten Sie durch diesen zusätzlichen Status eine neue Steuerungsmöglichkeit. Sie können die entsprechenden Benachrichtigungen dazu beispielsweise auch an einen anderen Ansprechpartner leiten oder gänzlich unterdrücken.

Globale Aktivierung

Damit Nagios/Icinga diese Erkennung überhaupt verwendet, müssen Sie sie in der Hauptkonfigurationsdatei zunächst aktivieren. Bei unserer Referenzinstallation ist dies die Datei */usr/local/icinga/etc/icinga.cfg*. Die Ablageorte für alle in Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten* beschriebenen Installationen haben wir für Sie noch einmal in Tabelle 3-3 zusammengestellt.

Tabelle 3-3: Ablageort der Hauptkonfigurationsdatei bei verschiedenen Installationsarten

Installationsart	Hauptkonfigurationsdatei
Icinga aus Quellen	/usr/local/icinga/etc/icinga.cfg
Icinga Paketinstallation	/etc/icinga/icinga.cfg
Nagios aus Quellen	/usr/local/nagios/etc/nagios.cfg
Nagios Paketinstallation	/etc/nagios/nagios.cfg

Die relevante Option zur Aktivierung der Flattern-Erkennung ist dabei folgende:

```
enable_flap_detection=1
```

Beachten Sie, dass die Erkennung von Status-Flattern standardmäßig mit dem Wert 0 deaktiviert ist, Sie also diese Änderung vermutlich manuell vornehmen müssen. Vergessen Sie nicht, Nagios/Icinga anschließend neu zu starten, um die aktualisierte Konfiguration neu einzulesen.

Aktivierung für Maschinen und Dienste

Anschließend können Sie für einzelne Maschinen und Dienste festlegen, ob die Flapping-Erkennung für diese verwendet werden soll. Hierzu geben Sie in der Definition von Maschinen (siehe Rezept 4.1, *Maschinen einbinden (host)*) und Diensten (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) die Option `flap_detection_enabled` mit dem Wert `0` für deaktiviert und `1` für aktiviert an. Typischerweise werden Sie diesen Wert über entsprechende Vorlagen (siehe Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*) setzen. Wir empfehlen Ihnen, die Flapping-Erkennung für alle Maschinen und Dienste zu aktivieren, da Sie damit eine Vielzahl von Benachrichtigungen durch eine einzelne Flattern-Benachrichtigung ersetzen können.

Anpassung der Erkennung

Die genaue Berechnung ist durchaus komplex, weshalb wir hier auf die Art der Auswertung nicht genauer eingehen werden. Lesen Sie bei Interesse hierzu bitte den entsprechende Teil in der Icinga/Nagios-Dokumentation (siehe unten), in der die Berechnungen genau erläutert werden. In der Hauptkonfigurationsdatei (siehe oben) sind jedoch Schwellenwerte für die Erkennungslogik vorgegeben. Die zunächst verwendeten Werte sind dabei folgende:

```
low_service_flap_threshold=5.0
high_service_flap_threshold=20.0
low_host_flap_threshold=5.0
high_host_flap_threshold=20.0
```

Neben der Möglichkeit, diese globalen Vorgaben an Ihre Bedürfnisse anzupassen, können Sie bei der Definition von Maschinen (siehe Rezept 4.1, *Maschinen einbinden (host)*) und Diensten (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) über die Optionen `low_flap_threshold` und `high_flap_threshold` auch spezifische Werte angeben und damit die globalen Werte überschreiben. Zusätzlich können Sie über die Option `flap_detection_options` festlegen, welche Status für die Erkennung verwendet werden sollen. Durch Weglassen können Sie dann also auch Status von der Erkennung ausschließen. Wir empfehlen Ihnen, die Vorgaben zunächst nicht zu ändern, wenn Sie nicht einen entsprechenden spezifischen Bedarf haben.

Aktionen bei Auftreten von Flattern

Wenn Sie die Erkennung aktiviert haben und Nagios/Icinga eine Maschine oder einen Dienst als flatternd erkennt, so wird es

- die Maschine oder den Dienst als flatternd führen und entsprechend in eigenen Kategorien anzeigen (siehe Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*),
- für die Dauer des Zustands automatisch einen entsprechenden Kommentar zu der Maschine beziehungsweise den Dienst hinzufügen,
- gegebenenfalls eine Benachrichtigung aufgrund des Flatterns versenden (siehe Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*) und
- keine weiteren Benachrichtungen aufgrund der wechselnden Status der Maschine oder des Dienstes selbst mehr versenden.

Analog wird das System beim Verlassen des Flattern-Zustands den eingerichteten Kommentar entfernen, gegebenenfalls nochmal eine Benachrichtigung aufgrund des Flatterns schicken und wieder beginnen, die normalen Benachrichtigungen der Maschine oder des Dienstes zu versenden.

Diskussion

Durch die Flattern-Erkennung können Sie möglicherweise unpassend gesetzte Schwellenwerte für die Status WARNING und CRITICAL erkennen. Nicht jedes Flattern sollte Sie allerdings sofort zu einer Anpassung der Warnschwellen schreiten lassen. Häufig wird es allerdings so sein, dass Sie die Benachrichtigung (siehe Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*) über das Flattern an einen anderen Ansprechpartner verschicken lassen als die Warnung zum eigentlichen Dienst. Sie wollen vielleicht beispielsweise einen Kunden über Ausfälle informieren, aber bei häufigen Statuswechseln lediglich selbst informiert werden.

Entsprechend gibt es unterschiedliche Szenarien, wie genau Sie diesen zusätzlichen Status einsetzen. Bei wenig umfangreichen Installationen werden Sie ihn möglicherweise auch gar nicht benötigen und deaktiviert lassen.

Siehe auch

- Dokumentation zum Flattern-Status bei Icinga und Nagios: *http://docs.icinga.org/latest/de/flapping.html* beziehungsweise *http://nagios.sourceforge.net/docs/3_0/flapping.html*
- Rezept zur Erstellung eigener Benachrichtigungen: Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*
- Kapitel mit den Installationsrezepten: Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*
- Rezepte zu Maschinen- und Dienstdefinitionen: Rezept 4.1, *Maschinen einbinden (host)* und Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Vereinfachung der Konfiguration mit Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Rezepte zur Nutzung der Oberflächen: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*

3.13 Kommentare, Acknowledgments und Downtimes nutzen

Problem

Sie möchten ein überwachtes System warten beziehungsweise ein bestehendes Problem beseitigen. Dabei wollen Sie vermeiden, dass von Ihrem Monitoring-System ein Alarm ausgelöst wird oder sich ein Kollege zeitgleich desselben Problems annimmt.

Lösung

Die Oberfläche von Nagios/Icinga (siehe Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*) bietet Ihnen die Möglichkeit, temporäre Notizen in Form von Kommentaren zu Maschinen beziehungsweise Diensten zu hinterlegen, Probleme als in Bearbeitung zu markieren und geplante Ausfallzeiten zu hinterlegen. Diese Möglichkeiten können die Zusammenarbeit mit Kollegen an Ihrem Monitoring-System erheblich vereinfachen. Wir zeigen Ihnen im Folgenden die einzelnen Schritte zur Nutzung und verweisen Sie auf nützliche Anwendungsbeispiele. Beachten Sie für dauerhafte Notizen bitte die in Rezept 4.11, *Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo)* beschriebenen Möglichkeiten, um konstante Informationen zu Maschinen oder Diensten hinzuzufügen.

Kommentare

Sie können über die Weboberfläche temporäre Notizen zu Maschinen oder Diensten hinterlegen, die Ihnen und Ihren Kollegen dann in den verschiedenen Ansichten angezeigt werden. Abbildung 3-35 zeigt am Beispiel eines Kommentars an einer Maschine, wie Ihnen die entsprechenden Informationen beim Darüberfahren mit der Maus in der Detail-Übersicht angezeigt werden. Auf den Detail-Seiten zu den betreffenden Maschinen und Diensten werden Ihnen diese Kommentare ansonsten als eigene Rubrik unter dem Status angezeigt.

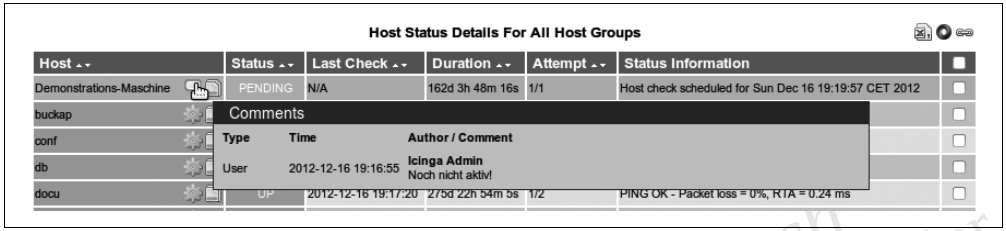


Abbildung 3-35: Kommentar beim Mouse-Over in der Icinga-GUI

Um einen Kommentar zu hinterlegen, können Sie zunächst die entsprechende Option `Add a new host/service comment` in der Detail-Ansicht wählen. Alternativ können Sie unter Icinga in den Detail-Übersichten eine oder mehrere Maschinen oder einen oder mehrere Dienste auswählen und diese Option dann im Dropdown-Feld darüber auswählen. Dabei können Sie entsprechend auch viele Maschinen auf einmal kommentieren. Kommentare bleiben dann bestehen, bis sie von einem Benutzer wieder gelöscht werden. Die Auswahl mehrerer Kommandos ist zur Zeit nur unter Icinga möglich.

Kommentare sind ein nützliches Mittel zur Kommunikation mit Ihren Kollegen, sofern Sie mit der Weboberfläche arbeiten. Wenn Sie beispielsweise bereits Erkenntnisse oder Hinweise zu einem Fehler haben, dieser aber eigentlich von einem Kollegen bearbeitet werden sollte, können Sie auf diesem Wege ebenfalls eine kurze Notiz mit ergänzenden Informationen hinterlegen.

Achnowledgements

Bei mehreren Verantwortlichen entsteht relativ häufig die Situation, dass unabhängig voneinander an der Lösung desselben Fehlers gearbeitet wird. Nagios/Icinga bieten Acknowledgements als Mittel, um diesem Problem zu begegnen. Der Status `acknowledged` kennzeichnet einen Fehler als *zur Kenntnis genommen*. Dieser Status wird unabhängig von den anderen Status gehandhabt, die über die Plugins zurückgeliefert werden.

Sofern Sie die Installation also nicht alleine betreuen, sollten Sie sich mit diesem Mechanismus vertraut machen. Die Nutzung ist unkompliziert: Sie können über die Detail-Ansicht von Maschinen/Diensten den Befehl `Add host/service acknowledgement` wählen, um ein Acknowledgement zu setzen. Alternativ können Sie auch hier die Detail-Über-

sicht nutzen, um mehrere Maschinen oder Dienste zu markieren und ein Acknowledgement für die gesamte Auswahl auf einmal zu erstellen.

Bei der Erstellung formulieren Sie dann einen geeigneten Kommentar. Sie können bei diesem angeben, ob er unabhängig von der Zeitdauer des Acknowledgements bestehen bleiben soll (`persistent comment`) oder nicht. Optional können Sie eine Ablaufzeit für das Acknowledgment angeben, nach der Nagios/Icinga es automatisch entfernt. Die Option, für die Dauer des Acknowledgments keine Benachrichtigungen (siehe Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*) zu versenden (`sticky acknowledgment`), ist für Sie vorausgewählt. Sie können diese Unterdrückung auch deaktivieren. Außerdem haben Sie hier die Möglichkeit zu verhindern, dass zu dem von Ihnen erstellten Acknowledgement selbst eine E-Mail versendet wird.

Nachdem Sie eine Maschine oder einen Dienst auf die beschriebende Weise mit einem Fehlerstatus markiert haben, wird Ihnen in der Oberfläche dazu das Symbol eines Schraubenschlüssels angezeigt (sowie das Kommentar-Symbol aufgrund des mit dem Acknowledgement hinterlegten Kommentars). Abbildung 3-36 zeigt Ihnen einen entsprechenden Dienst in der Detail-Übersicht.

SSH	OK	2012-12-17 14:09:21	98d 1h 31m 34s	1/5	SSH OK - OpenSSH_5.5p1 Debian-6+squeeze2 (protocol 2.0)
check_fs-root	CRITICAL	2012-12-17 14:10:01	15d 10h 38m 44s	5/5	/: 97%used(5GB/6GB) (>75%) : CRITICAL
check_iftraffic	OK	2012-12-17 14:09:21	50d 2h 20m 57s	1/5	Average IN: 0.18KBs (0.00%), Average OUT: 0.25KBs (0.00%) Total RX: 628.54 MBytes, Total TX: 387.76 MBytes

Abbildung 3-36: Ein Dienst mit Status Acknowledged in der Detail-Übersicht

So erkennen Sie und andere, dass an dem Problem bereits gearbeitet wird. Gerade bei vielen Prüfungen entlasten Sie mit diesem Vorgehen die Kollegen, auch wegen der Unterdrückung der Benachrichtigungen. Der hinterlegte Kommentar wird entsprechend jeweils angezeigt (in der Detail-Ansicht sowie in der Kommentar-Ansicht).

Acknowledgment expiry: Seit Icinga 1.7 gibt es die Möglichkeit, Acknowledgments ablaufen zu lassen. Dies bietet sich an, wenn man bereits weiß, bis wann das Problem behoben sein sollte. Ab diesem Zeitpunkt werden dann wieder normale Benachrichtigungen verschickt.

Geplante Ausfallzeiten (scheduled downtime)

Nagios/Icinga bietet Ihnen die Möglichkeit, bekannte Ausfallzeiten zu Diensten und/oder Maschinen zu hinterlegen. Als Folge wird Nagios/Icinga während dieser Zeit keine Benachrichtigungen (siehe Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*) zu den entsprechenden Maschinen/Diensten versenden. Sie können sich Ausfallzeiten also als im Voraus geplante Acknowledgements vorstellen.

Um eine solche Ausfallzeit zu hinterlegen, können Sie beispielsweise in der Detail-Ansicht der Maschine oder des Dienstes den entsprechenden Befehl an der rechten Seite auswählen oder mehrere Maschinen beziehungsweise Dienste in den Detail-Übersichten markieren und dann den entsprechenden Befehl aus dem Dropdown-Feld oben wählen (zur Zeit geht diese Massenbearbeitung allerdings nur unter Icinga). Sie können dabei zunächst einen Kommentar zu der Ausfallzeit hinterlegen.

Kommentare zu Ausfallzeiten: Beachten Sie, dass es sich hier nicht um einen Kommentar wie die oben diskutierten handelt. Er ist extra für die Ausfallzeit bestimmt und wird nur bei dieser angezeigt (in der Detail-Ansicht der Maschine/des Dienstes). Nagios/Icinga wird allerdings einen automatischen Kommentar (im Sinne der vorherigen Beschreibung) zu der Ausfallzeit erstellen.

Mit den weiteren Optionen, Verkettung von Ausfallzeiten und dynamische Ausfallzeiten, können Sie Start und Ende einerAusfallzeit hinterlegen. Zu diesen Zeiten werden dann keine Benachrichtigungen verschickt, auch wenn die Maschine oder der Dienst nicht erreichbar ist. Bei Maschinen können Sie zusätzlich angeben, ob abhängige Maschinen ebenfalls betroffen sind (Verkettung). Wenn Sie diese Option nutzen, werden alle diese Maschinen automatisch auch mit einer entsprechenden Ausfallzeit versehen.

Diskussion

Die Kommentare und zusätzlichen Status sind einfache, aber effiziente Werkzeuge, um die Zusammenarbeit am Monitoring-System zu realisieren. Damit diese die entsprechenden Prozesse steuern können, ist aber eine Umgewöhnung seitens der betroffenen Mitarbeiter notwendig, die normalerweise eine gewisse Zeit dauern wird. Rechnen Sie also zunächst damit, dass betroffene Mitarbeiter unter Umständen vergessen, auf Kommentare/Status zu achten beziehungsweise diese zu setzen. Aufgrund der großen Hilfe, die diese Mechanismen bedeuten, werden sie allerdings häufig gut aufgenommen und nach einiger Zeit in die Arbeitsabläufe integriert.

Siehe auch

- Rezepte zur Nutzung der Oberflächen: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*
- Rezept zur Integration mit Fremd-Systemen: Rezept 4.11, *Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo)*
- Rezept zur Erstellung eigener Benachrichtigungen: Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*

KAPITEL 4

Die Konfigurationsobjekte von Nagios/Icinga

Was Sie wie überwachen wollen, legen Sie für Nagios/Icinga über Konfigurationsdateien fest, die sogenannte Konfigurationsobjekte enthalten. Wie diese aufgebaut sind und voneinander abhängen, werden wir in diesem Kapitel erläutern. Ihnen stehen dabei eine ganze Menge von Konfigurationsobjekten zur Verfügung, die sich jeweils weiter durch entsprechende Optionen modifizieren lassen. Abhängig davon, wie Sie Ihr System einsetzen, werden Sie eventuell nicht alle dieser Objekte benötigen. Die Folgenden sind aber in jedem Fall erforderlich:

- `host` (siehe Rezept 4.1, *Maschinen einbinden (host)*)

  Mit diesen Gerätedefinitionen bilden Sie zunächst die zu überwachenden Maschinen ab. Dies beinhaltet dabei erstmal nur eine Prüfung zur Erreichbarkeit der Maschine, den sogenannten `host-check`.

- `command` (siehe Rezept 4.3, *Befehle hinzufügen (command)*)

  Mit diesen Befehlsdefinitionen binden Sie Programme, die sogenannten Plugins, in Nagios/Icinga ein und stellen damit grundsätzliche Test-Typen zur Verfügung. Die angesprochene Erreichbarkeitsprüfung muss ebenfalls auf diese Weise definiert sein.

- `service` (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*)

  Mit diesen Objekten legen Sie fest, welche per Befehlsdefinition zur Verfügung stehenden Tests zusätzlich zu der reinen Erreichbarkeitsprüfung wie genau auf welchen Maschinen durchgeführt werden sollen.

Technisch gesehen sind weitere Objekte erforderlich und weitere möglich. Da Ihnen mit der Installation aber eine lauffähige Konfiguration mit allen notwendigen Objekten erstellt wurde, stehen Ihnen diese bereits mit Vorgabewerten zur Verfügung. Sie müssen also gegebenenfalls nur herausfinden, an welchen Stellen hier Anpassungen für Ihre Umgebung notwendig sind. Wir erläutern Ihnen zusätzliche Objekttypen in den Rezepten:

- `contact` (siehe Rezept 4.4, *Kontakte definieren (contact)*)

  Kontakte dienen der Zustellung von Benachrichtigungen. Über diese Konfigurationsobjekte legen Sie entsprechende Kontakte an und weisen Ihnen die nötigen Merkmale wie etwa eine E-Mail-Adresse zu.

- `timeperiod` (siehe Rezept 4.5, *Zeitperioden definieren (timeperiod)*)

  Über die Definition von Zeitperioden bestimmen Sie die Gültigkeit anderer Objekte. Auf diese Weise können Sie zeitlich einschränken, wann bestimmte Prüfungen ausgeführt oder Benachrichtigungen versand werden.

- `host-group` (siehe Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*), `service-group` (siehe Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*), `contact-group` (siehe Rezept 4.8, *Kontakte in Gruppen zusammenfassen (contact-group)*)

  Mit diesen Objekten können Sie Gruppen der jeweiligen Objekten (Maschinen, Dienste und Kontakte) bilden. Dies kann Ihnen die Konfiguration erheblich vereinfachen und für Übersichtlichkeit sorgen.

- `host-dependency` und `service-dependency` (siehe Rezept 4.9, *Abbilden von Abhängigkeiten (host- und service-dependencies)*)

  Mit diesen Objekten können Sie zusätzliche Abhängigkeiten zwischen Maschinen und Diensten abbilden.

In Abbildung 4-1 sehen Sie die Abhägigkeiten zwischen den Konfigurationsobjekten. Darüber hinaus erläutern wir Ihnen noch weitere Optionen, die Sie in verschiedenen Konfigurationsobjekten verwenden können. Dies sind benutzerdefinierte Variablen (siehe Rezept 4.10, *Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen*) und solche zur Hinterlegung zusätzlicher Informationen für die Anzeige oder Integration mit externen Systemen (siehe Rezept 4.11, *Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo)*).

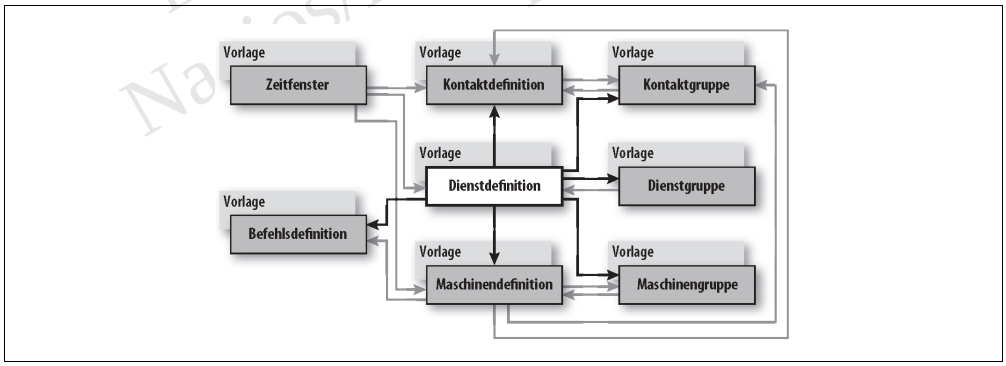


Abbildung 4-1: Abhängigkeiten zwischen den Konfigurationsobjekten

Bevor Sie in größerer Zahl Maschinen und Dienste zu Ihrer Konfiguration hinzufügen, sollten Sie unbedingt die Rezepte zur Verwendung von Vorlagen (siehe Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*) und strukturierten Ablage der Konfigurationsdateien (siehe Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*) aus dem vorhergehenden Kapitel lesen. Damit ersparen Sie sich möglicherweise viel nachträgliche Arbeit.

4.1 Maschinen einbinden (host)

Problem

Sie möchten eine Maschine zur Konfiguration hinzufügen.

Lösung

Wir zeigen Ihnen im Folgenden, wie Sie Maschinen in Nagios/Icinga definieren. Das Grundgerüst einer leeren Maschinen-Definition (Konfigurationsobjekt: host) ist folgendes:

```
define host {
       option                                  wert
}
```

Für Maschinen gibt es dabei insgesamt eine recht große Anzahl an Optionen. Deshalb sollten Sie sich an dieser Stelle mit dem Vorlagen-System vertraut gemacht haben (siehe Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*), um unnötige Redundanzen zu vermeiden. Wie Sie in Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien* genauer nachlesen können, weisen Sie definierten Maschinen später über Dienste (siehe hierzu Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) die eigentlichen Prüfungen zu.

Konzept host-check

Nagios/Icinga kennen zwei Arten von Prüfungen, den sogenannten host-check für Maschinen und den `service-check` für die eigentlichen Prüfungen, die wir Dienste nennen. Ein `host-check` soll überprüfen ob eine Maschine prinzipiell verfügbar ist, während ein `service-check` einen bestimmten Dienst auf dieser Maschine überprüft. Im Rahmen der Maschinen definieren Sie zunächst über die Option check_command nur eben diesen `host-check`. Die eigentlichen Prüfungen definieren Sie später als Dienste (Konfigurationsobjekt `service`, siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*).

Wichtig für Sie ist dieser `host-check`, weil Nagios/Icinga bei einem nicht erfolgreichen host-check gar nicht erst versucht, einen `service-check` auf dieser Maschine auszuführen. Hintergrund ist die Annahme, dass die Maschine nicht läuft, wenn der `host-check` fehlschlägt, und entsprechend auch keine Dienste auf dieser Maschine erreichbar seien. Typischerweise wird als `host-check` der Befehl `ping` eingesetzt (siehe Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*), der die Erreichbarkeit einer IP-Adresse testet und die Signallaufzeit als Rückgabewert liefert.

Optionen für Maschinen

Zunächst können Sie die bei Maschinen die allgemeinen Optionen register und use verwenden, um zu kontrollieren ob sich die Definition als Vorlage verhält (`register` auf den

Wert 0 setzen) und ob sie von einer anderen Vorlage abgeleitet ist (use mit einer durch Kommata separierten Liste von Namen von zu nutzenden Vorlagen). Für Hinweise, wie Sie die Vererbung und Mehrfachvererbung über Vorlagen nutzen können, beachten Sie bitte auch Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*.

Pflicht-Optionen: Für ihre Gültigkeit muss Ihre resultierende Konfigurationsanweisung (also gegebenenfalls nach dem Zusammensetzen aus referenzierten Vorlagen, siehe auch Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*) die folgenden Pflichtoptionen mit gültigen Werten belegen:

- `host_name`

  Ein interner Name für die Maschine beziehungsweise die Vorlage (falls Sie `register` auf `0` setzen). Mit diesem Namen referenzieren Sie die Maschine beziehungsweise Vorlage später in anderen Konfigurationsobjekten.
- `alias`

  Ein beschreibender Name für die Maschine, den Sie frei wählen können.
- `address`

  Die Adresse der Maschine: Sie können hier eine IP-Adresse oder einen Fully Qualified Domain Name (FQDNs), also den entsprechenden Domänennamen, verwenden. Sofern nicht gesetzt wird automatisch der Wert von `host_name` verwendet.
- `check_period`

  Die Referenz auf ein Zeitfenster (Konfigurationsobjekt timeperiod, siehe Rezept 4.5, *Zeitperioden definieren (timeperiod)*), während dessen Prüfungen der Maschine (eben der `host-check`) durchgeführt werden sollen.
- `max_check_attempts`

  Anzahl der durchzuführenden Prüfungen mittels `check_command` (siehe unten) bevor die Maschine als nicht verfügbar behandelt wird.
- `contacts` und/oder `contact_groups`

  Angabe durch Kommata getrennter Namen von als Konfigurationsobjekt `contact` (siehe Rezept 4.4, *Kontakte definieren (contact)*) definierten Kontakten und/oder als `contactgroup` (siehe Rezept 4.8, *Kontakte in Gruppen zusammenfassen (contactgroup)*) definierten Kontaktgruppen für Benachrichtigungen.
- `notification_period`

  Angabe eines (bereits definierten) Zeitfensters (Konfigurationsobjekt timeperiod, siehe Rezept 4.5, *Zeitperioden definieren (timeperiod)*), währenddessen Benachrichtigungen für diese Maschine versendet werden sollen.
- `notification_interval`

  Intervall für wiederholte Benachrichtigungen über einen anhaltenden Fehlerzustand.

Eine solche Minimal-Konfiguration würde allerdings nur als Platzhalter dienen, also zwar angezeigt aber dabei keinerlei Test durchgeführt werden. Einige der als Pflichtangaben

geführten Optionen könnten Sie dabei sogar noch weglassen, ohne das dies zunächst zu Fehlern führen würde. Wir empfehlen Ihnen hier aber sich an die Vorgaben der Dokumentation zu halten, um mögliche Folgeprobleme auszuschließen.

Weitere Optionen: Nagios/Icinga bieten Ihnen insgesamt über 40 verschiedene Optionen für die Konfiguration von Maschinen. Wir gehen hier zunächst nur auf die uns allgemein besonders wichtig erscheinenden ein. Spezifische weitere Optionen erläutern wir Ihnen in den Rezepten zum Flattern-Status (Rezept 3.12, *Den Flattern-Status verstehen und nutzen*), zu passiven Prüfungen (Rezept 3.2, *Das Konzept von aktiven und passiven Prüfungen*), zu benutzerdefinierten Variablen (Rezept 4.10, *Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen*) und zur Erweiterung um Informationen zur Anzeige und Einbindung von externen Systemen (Rezept 4.11, *Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo)*).

Die wichtigsten Optionen, die zusätzlich zu den Pflichtoptionen `check_period` und `max_check_attempts` den `host-check` beeinflussen, sind folgende:

- `check_command`

  Hier geben Sie eine Referenz auf einen Befehl (Konfigurationsobjekt `command`, siehe Rezept 4.3, *Befehle hinzufügen (command)*) an, der als `host-check` für diese Maschine ausgeführt werden soll.
- `check_interval`

  Dies ist die Frequenz in Zeiteinheiten (Standard: Minuten), mit der der `host-check` für diese Maschine durchgeführt werden sollen.
- `retry_interval`

  Hier definieren Sie die Frequenz, mit der Wiederholungsprüfungen nach einer fehlerhaften Prüfung durchgeführt werden sollen.
- active_checks_enabled und passive_checks_enabled

  Aktivieren Sie die Durchführung von aktiven beziehungsweise passiven Prüfungen mit dem Wert `1` oder deaktivieren Sie sie mit dem Wert `0`. Den Unterschied zwischen den beiden Durchführungsvarianten erläutern wir in Rezept 3.2, *Das Konzept von aktiven und passiven Prüfungen*.
- `parents`

  Hiermit wird eine Eltern-Kind-Beziehung zwischen den angegebenen durch Kommata getrennten Maschinen und der aktuellen Maschine eingerichtet. Diese Abhängigkeiten sind die Grundlage, auf der Nagios/Icinga eine Feststellung trifft, auf welchen Maschinen die Durchführung eines Tests Sinn macht. Das System geht dabei davon aus, dass es über die Eltern-Beziehungen eine Verbindung mit erreichbaren Maschinen geben muss, damit auch eine Maschine selbst erreichbar ist. Die Beziehung ist auch die Grundlage der Darstellung in der Status Map (siehe hierzu Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*).

- `hostgroups`

  Geben Sie die Referenz auf eine oder mehrere durch Kommata getrennte Maschinengruppen (Konfigurationsobjekt `hostgroup`, siehe Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*) an, der oder denen diese Maschine angehören soll.

Optionen die zusätzlich zu den Pflichtoptionen `contacts/contact_groups`, `notification period` und `notification_interval` die Benachrichtigungen steuern:

- `notifications_enabled`

  Aktivieren oder Deaktivieren Sie Benachrichtigungen für diese Maschine (Wert `0` = aus, `1` = an).

- `notification_options`

  Hier können Sie Zustände (siehe Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*) angeben, für die Benachrichtigungen versendet werden sollen.

Sonstige Optionen, die nicht in die bisherigen Kategorien passen:

- `display_name`

  Mit dieser Option können Sie einen alternativen Name angeben, der in der Weboberfläche anstelle des Wertes von host_name angezeigt wird (jedoch nicht unter Icinga-Web).

- `retain_status_information` und `retain_nonstatus_information`

  Hier steuern Sie, ob die zustandsbezogenen beziehungsweise nicht zustandsbezogenen Daten zusätzlich zur Aufbewahrung im Speicher auch auf der Festplatte abgelegt werden sollen, damit Sie nach einem Neustart weiter zur Verfügung stehen (**`0`** = aus, **`1`** = an). Sie sollten diesen Wert aktiviert lassen, da es sonst zu Fehlern kommen kann.

- `event_handler_enabled`

  Mit dieser Option können Sie die Ereignisverarbeitung für diese Maschine aktivieren oder deaktivieren (`0` = aus, `1` = an). Die Ereignisverarbeitung erlaubt Ihnen, bei Statuswechseln der Maschine Kommandos auszuführen.

- `flap_detection_enabled`

  Hier können Sie die Erkennung und Verarbeitung des Flattern-Zustandes für diese Maschine kontrollieren (`0` = aus, `1` = ein, siehe Rezept 3.12, *Den Flattern-Status verstehen und nutzen*)

- `process_perf_data`

  Diese Option erlaubt Ihnen für diese Maschine vorzugeben, ob Performancedaten verarbeitet, also zusätzlich zum eigentlich Status ermittelte Informationen zur Performance anzeigt und gegebenenfalls weiterleitet werden sollen (Wert `0` = nein, `1` = ja). Wenn Sie Graphen für diese Maschine erstellen lassen möchten, müssen Sie diese Option normalerweise aktivieren. Voraussetzung ist allerdings, dass das Plugin auch Performancedaten liefert.

Diskussion

Wie Sie sehen, gibt es eine Vielzahl von Einstellungsmöglichkeiten. Sie können diese zwar für jede Maschine einzeln definieren, aber dies ist selten erforderlich und deshalb meist nicht empfehlenswert. Nutzen Sie daher Vorlagen (siehe Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*), um die für Sie relevanten Optionen auf sinnvolle Werte zu setzen und leiten Sie dann die nachfolgenden Definitionen von dieser Vorlage ab. Auf diese Weise müssen Sie nur jeweils von Ihrem Standard abweichende Optionen setzen.

Betrachten Sie zum Beispiel die bei der Installation generierte Konfiguration (hier für unser Referenzsystems). Dabei wurde unter anderen eine Vorlage für Maschinen mit dem Namen `generic-host` erstellt, die die folgenden Optionen setzt:

```
define host{
        register                        0

        name                            generic-host

        notifications_enabled           1
        event_handler_enabled           1
        flap_detection_enabled          1
        failure_prediction_enabled      1
        process_perf_data               1
        retain_status_information       1
        retain_nonstatus_information    1
        notification_period             24x7
}
```


Die Bedeutung der relevanten Optionen können Sie den obigen Aufstellungen entnehmen. Beachten Sie die Verwendung von `register`, um die Definition als Vorlage zu definieren. Der Wert für die Option `notification_period` referenziert ein an anderer Stelle definiertes Zeitfenster (siehe hierzu Rezept 4.5, *Zeitperioden definieren (timeperiod)*). Eine weitere erstellte Vorlage `linux-server` erbt und ergänzt diese Einstellungen:

```
define host{
        register                        0
        use                             generic-host

        name                            linux-server

        check_period                    24x7
        check_interval                  5
        retry_interval                  1
        max_check_attempts              10
        check_command                   check-host-alive
        notification_period             workhours
        notification_interval           120
        notification_options            d,u,r
        contact_groups                  admins
}
```

Beachten Sie erneut die Verwendung der Option `register` und wie hier über die Option `use` die Vorlage `generic-host` eingebunden wird. Der Wert der geerbten Option `notification_period` wird hier mit dem Verweis auf ein anderes Zeitfenster überschrieben. Alle anderen Optionen werden zusätzlich zu den geerbten definiert. Die erstellte Konfiguration enthält außerdem mit `localhost` die Definition einer Maschine unter Nutzung dieser zweiten Vorlage:

```
define host {
        use                             linux-server

        host_name                       localhost
        alias                           localhost
        address                         127.0.0.1
}
```

Über die Option `use` wird die Vorlage referenziert und damit alle entsprechenden Optionen übernommen. Da die Option `register` nicht gesetzt wird, ist diese Definition keine Vorlage. Zusätzlich zu den übernommen Optionen werden hier dann ein Kurzname, ein beschreibender Name und eine Adresse für die Maschine definiert.

Siehe auch

- Dokumentation zum Konfigurationsobjekt host bei Icinga: *http://docs.icinga.org/latest/de/objectdefinitions.html#host* beziehungsweise bei Nagios: *http://nagios.sourceforge.net/docs/nagioscore/3/en/objectdefinitions.html#contact*
- Rezept zur Vereinfachung der Konfiguration mit Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Rezept zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezepte zur Definition von Kontakten und Kontaktgruppen: Rezept 4.4, *Kontakte definieren (contact)* und Rezept 4.8, *Kontakte in Gruppen zusammenfassen (contactgroup)*
- Rezept zur Überwachung der Erreichbarkeit mit Ping: Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*
- Rezept zum Flattern-Status: Rezept 3.12, *Den Flattern-Status verstehen und nutzen*
- Rezept zu aktiven und passiven Prüfungen: Rezept 3.2, *Das Konzept von aktiven und passiven Prüfungen*
- Rezept zur Zusammenfassung von Maschinen in Maschinengruppen: Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*
- Rezept zur Definition von Zeitfenstern: Rezept 4.5, *Zeitperioden definieren (timeperiod)*

- Rezept zu benutzerdefinierten Variablen (Rezept 4.10, *Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen*)
- Rezept zur Erweiterung um Informationen zur Anzeige und Einbindung von externen Systemen (Rezept 4.11, *Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo)*)
- Rezept zur Standard-Oberfläche: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*
- Rezept zur Anpassung von Benachrichtigungen: Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*

4.2 Dienste: Befehle mit Maschinen verknüpfen (service)

Problem

Sie möchten einen neuen Dienst einrichten, in dem ein bereits konfigurierter Befehl (siehe Rezept 4.3, *Befehle hinzufügen (command)*) für ebenfalls bereits definierte Maschinen (siehe Rezept 4.1, *Maschinen einbinden (host)*) ausführt wird.

Lösung

Dienste sind das Bindeglied zwischen Befehlen und Maschinen. Sie definieren Dienste über `define service`-Anweisungen (Konfigurationsobjekt: service) in der folgenden Struktur:

```
define service{
       option                                  wert
}
```

Die Ihnen hierbei zur Verfügung stehenden Optionen werden wir auszugsweise im Folgenden erläutern. Eine vollständige Referenz finden Sie in der Dokumentation zu Ihrer Version.

Optionen für Services

Zunächst können Sie bei Diensten die allgemeinen Optionen `register` und `use` verwenden, um Vorlagen zu erstellen (`register` mit Wert `0`) beziehungsweise einzubinden (`use` mit dem Kurznamen der Vorlage). Die Optionen, die Sie zu jedem Dienst angeben müssen sind dann folgende:

- `host_name` und/oder `hostgroup_name`

  Geben Sie hier durch Kommata separiert die Kurznamen der Maschinen beziehungsweise Maschinengruppen an, denen der Dienst zugeordnet werden soll.
- `service_description`

  Der Name für den Dienst, unter dem er in der Weboberfläche angezeigt wird.

- `check_command`

  Die Referenz auf den Befehl, der aufgerufen werden soll (siehe Rezept 4.3, *Befehle hinzufügen (command)*), gegebenenfalls mit durch `"!"` separierten Argumenten.

- `max_check_attempts`

  Geben Sie hier die Anzahl der Wiederholungsversuche an, bevor der Dienst den Zustand wechselt. Damit können Sie kurzzeitige Fehler ignorieren anstatt für diese gegebenenfalls mehrere Benachrichtigungen zu erhalten.

- `check_interval`

  Die Anzahl der Zeiteinheiten, die zwischen normalen Prüfungen liegen soll (Standardeinheit Minuten).

- `retry_interval`

  Die Anzahl von Zeiteinheiten (Standardeinheit Minuten), die zwischen den Wiederholungsversuchen gewartet werden soll.

- `check_period`

  Referenz eines Zeitfensters (siehe Rezept 4.5, *Zeitperioden definieren (timeperiod)*), während dessen der Dienst überprüft werden soll.

- `notification_interval`

  Anzahl von Zeiteinheiten (Standardeinheit Minuten), die vor erneuten Benachrichtigungen wegen eines Fehlers gewartet werden soll.

- `notification_period`

  Referenz eines Zeitfensters (siehe Rezept 4.5, *Zeitperioden definieren (timeperiod)*), während dessen Benachrichtigungen versendet werden sollen.

- `contacts` und/oder `contact_groups`

  Geben Sie hier durch Kommata getrennt die Kontakte (siehe Rezept 4.4, *Kontakte definieren (contact)*) oder Kontaktgruppen (siehe Rezept 4.8, *Kontakte in Gruppen zusammenfassen (contact-group)*) an, die Zugriff auf diesen Dienst haben beziehungsweise über Zustandsänderungen benachrichtigt werden sollen

Zusätzlich zu diesen immer benötigten Optionen stehen Ihnen noch zahlreiche weitere Optionen zur Verfügung. Die wichtigsten stellen wir Ihnen im Folgenden kurz vor.

Weitere Optionen

Die Steuerung der Durchführung der Prüfungen können Sie mit den folgenden Optionen beeinflussen:

- `active_checks_enabled`

  Hier kontrollieren Sie, ob aktive Prüfungen über den `check_command` ausgeführt werden sollen oder nicht (`0` für nein, `1` für ja).

- `passive_checks_enabled`

  Hier kontrollieren Sie, ob passive Ergebnisse von Prüfungen für diesen Service entgegengenommen und verarbeitet werden sollen (`0` für nein, `1` für ja, siehe Rezept 3.2, *Das Konzept von aktiven und passiven Prüfungen*).

Optionen, mit dem Sie die Anzeige des Dienstes in der Weboberfläche beeinflussen können, sind folgende:

- `display_name`

  Hier können Sie einen Namen angeben, der anstelle des Wertes von `service_description` für die Anzeige in der Weboberfläche genutzt werden soll.

- `notes`

  Hier können Sie einen Text angeben, der in der Detailansicht des Dienstes angezeigt wird.

- `icon_image`

  Hier können Sie den Dateinamen eines Bildes angeben, dass als Icon für diesen Dienst verwendet werden soll (Pfad zu den Dateien ist in beispielsweise der quellbasierten unter */usr/local/icinga/share/images/logos*).

- `servicegroups`

  Hier können Sie eine durch Kommata separierte Liste von bereits definierten Dienste-Gruppen angeben (siehe Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*), denen dieser Dienst angehören soll.

Zusätzliche Optionen, mit denen Sie die Benachrichtigungen genauer steuern können, sind folgende:

- `notifications_enabled`

  Mit dieser Option schalten Sie Benachrichtigungen für diesen Dienst generell an oder aus (`0` für aus, `1` für an).

- `notification_options`

  Mit dieser Option können Sie genau angeben, für welche Ereignisse bei diesem Dienst Benachrichtigungen versendet werden sollen (siehe Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*) .

Weitere Optionen, die nicht in die bisherigen Kategorien passen, aber die Sie dennoch kennen sollten, da Sie bereits in der Standardkonfiguration vorkommen sind:

- `retain_status_information` und `retain_nonstatus_information`

  Hier steuern Sie, ob die zustandsbezogenen beziehungsweise nicht zustandsbezogenen Daten zusätzlich zur Aufbewahrung im Speicher auch auf der Festplatte abgelegt werden sollen, damit Sie nach einem Neustart weiter zur Verfügung stehen (`0` = aus, `1` = an).

- `event_handler_enabled`

  Mit dieser Option können Sie die Ereignisverarbeitung für diese Maschine aktivieren oder deaktivieren (`0` = aus, `1` = an).

- `check_freshness`

  Hier legen Sie fest, ob für diesen Dienst Aktualitätsprüfungen stattfinden sollen oder nicht (`0` für nein, `1` für ja, siehe Rezept 3.2, *Das Konzept von aktiven und passiven Prüfungen*).

- `flap_detection_enabled`

  Ob für diesen Dienst häufige Zustandswechsel als zusätzlicher Zustand `Flattern` erkannt werden sollen oder nicht, steuern Sie mit dieser Option (`0` für nein, `1` für ja, siehe Rezept 3.12, *Den Flattern-Status verstehen und nutzen*).

- `process_perf_data`

  Hier kontrollieren Sie, ob Nagios/Icinga für diesen Dienst Performancedaten verarbeitet, also zusätzlich zum eigentlich Status ermittelte Informationen anzeigt und gegebenenfalls weiterleitet (`0` = nein, `1` = ja).

Diese Auswahl an Optionen zeigt, wie vielfältig Sie Dienste anpassen können. Eine vollständige Referenz finden Sie in der Dokumentation zu Ihrer Version.

Diskussion

Aufgrund der großen Anzahl an Optionen sollten Sie auch bei Diensten unbedingt mit Vorlagen arbeiten. Definieren Sie für Sie sinnvolle Standardwerte in einer allgemeinen Vorlage, für die Sie die bei der Installation erstellte Vorlage `generic-service` als Grundlage nutzen können.

```
define service {
       register                        0

       name                            generic-service

       active_checks_enabled           1
       passive_checks_enabled          1
       parallelize_check               1
       obsess_over_service             1
       check_freshness                 0
       event_handler_enabled           1
       flap_detection_enabled          1
       failure_prediction_enabled      1
       process_perf_data               1
       retain_status_information       1
       retain_nonstatus_information    1
       is_volatile                     0
```

```
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           10
        retry_check_interval            2

        notifications_enabled           1
        notification_period             24x7
        notification_interval           60
        notification_options            w,u,c,r
        contact_groups                  admins
}
```

Beachten Sie hier insbesondere die Verwendung der Option `register`, um eine Vorlage zu definieren. Bei `check_period` und `notification_period` wird das bei der Installation angelegte und immer gültige Zeitfenster `24x7` referenziert (zu Zeitfenstern siehe Rezept 4.5, *Zeitperioden definieren (timeperiod)*). Die erstellte Konfiguration erhält außerdem eine zweite Vorlage als Beispiel für eine Ableitung von der eben angeführten Vorlage:

```
define service {
        register                        0
        use                             generic-service

        name                            local-service
        max_check_attempts              3
        normal_check_interval           1
        retry_check_interval            1
}
```

Beachten Sie hier die Nutzung der Option `use`, um die eben angeführte Vorlage zu referenzieren und damit alle Optionen von dort zu übernehmen. Das Beispiel setzt dabei für drei bereits in der Vorlage vorhandene Optionen neue Werte. Sie könnten hier auch noch nicht gesetzte Optionen hinzufügen.

Unter Nutzung dieser Vorlagen können Sie dann mit wenigen Optionen die eigentlichen Dienste definieren. Der bei der Installation erstellte Dienst zur Durchführung des Befehls `check_ping` auf der lokalen Maschine umfasst entsprechend beispielsweise nur vier Optionen:

```
define service {
        use                             local-service
        host_name                       localhost
        service_description             PING
        check_command                   check_ping!100.0,20%!500.0,60%
}
```

Zunächst wird über die Option `use` die Vorlage `local-service` referenziert, so dass alle dort gesetzten Optionen übernommen werden. Zusätzlich definiert dieser Dienst dann über `host_name` zunächst, auf welcher Maschine er ausgeführt werden soll. Über `service_description` wird dann weiter der Name für den Dienst festgelegt und schließlich der auszuführenden Befehl mit der Option `check_command` referenziert. Hier werden dem Aufruf dabei zwei Optionen übergeben, bei denen es sich hier um die Schwellenwerte für die Status `WARNING` und `CRITICAL` wie vom Befehl `check_ping` benötigt handelt.

Gruppenzuweisung: Es kommt häufig vor, dass mehrere Geräte eines Typs vorhanden sind, auf denen die gleichen Dienste durchgeführt werden sollen. Sie können die Konfiguration hier auf zwei Wegen vereinfachen:

1. Sie referenzieren die Maschinen in einer Vorlage, dann müssen Sie sie bei der Definition des eigentlichen Dienstes nicht mehr angeben, oder
2. Sie fassen die Maschinen in einer Maschinengruppe zusammen (siehe Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*) und verwenden die Option *hostgroup_name* um den Dienst allen Maschinen dieser Gruppe(n) zu vernüpfen.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezepte zur Einbindung von Maschinen und Maschinengruppen: Rezept 4.1, *Maschinen einbinden (host)* und Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*
- Rezepte zur Einbindung von Diensten und Dienstegruppen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* und Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*
- Rezept zur Definition von Zeitfenstern: Rezept 4.5, *Zeitperioden definieren (timeperiod)*
- Rezepte zur Einbindung von Kontakten und Kontaktgruppen: Rezept 4.4, *Kontakte definieren (contact)* und Rezept 4.8, *Kontakte in Gruppen zusammenfassen (contactgroup)*
- Rezept zur Erstellung eigener Benachrichtigungen: Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*
- Rezept zu aktiven und passiven Prüfungen: Rezept 3.2, *Das Konzept von aktiven und passiven Prüfungen*
- Rezept zum Flattern-Status: Rezept 3.12, *Den Flattern-Status verstehen und nutzen*

4.3 Befehle hinzufügen (command)

Problem

Sie möchten noch andere als die bereits eingerichteten Plugins für Prüfungen verwenden.

Lösung

Hier gehen wir davon aus, dass Sie ein bereits installiertes Plugin für die Benutzung in Nagios/Icinga einrichten wollen.

Plugins: Wie Sie eigene Plugins erstellen und zusätzliche Plugins einbinden, wird in den Rezepten in Kapitel 7, *Erstellung eigener und Einbindung externer Plugins* erläutert.

Um Plugins in Nagios/Icinga verwenden zu können, benötigen Sie jeweils eine entsprechende Befehlsdefinition (Konfigurationsobjekt command) in einer der verarbeiteten Konfigurationsdateien. Mit diesen Befehlsdefinitionen legen Sie jeweils einen Namen und eine dazugehörige Befehlszeile fest.

Testdatei: Wenn Sie die folgenden Schritte jetzt durchführen möchten, aber unsicher sind, in welcher Datei Sie die Definitionen ablegen sollen, legen Sie sich einfach eine entsprechende Datei für Tests an. Damit diese auch verwendet wird, müssen Sie sie entweder direkt in der Hauptkonfigurationsdatei referenzieren oder in einem dort referenzierten Ordner ablegen. Im Zweifel lesen Sie hierzu bitte Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*.

Wir werden Ihnen diesen Schritt im Folgenden am Beispiel des typischerweise in den mitinstallierten Plugins vorhandenen check_dummy erläutern. Eine einfache Befehlsdefinition für `check_dummy` sieht wie folgt aus:

```
define command{
       command_name                    check_demo_a
       command_line                    /usr/local/icinga/libexec/check_dummy 0 \
                                       "Alles ist gut."
}
```


Die Befehlsdefinitionen sind die einfachsten Konfigurationsoptionen in Nagios/Icinga, da sie immer genau aus den zwei Optionen `command_name` für den Namen und `command_line` für die Spezifikation des auszuführenden Programms bestehen. Gegenüber den anderen Konfigurationsobjekten gibt es hier keine Vorlagen oder weitere Optionen. Die obige Konfiguration ist also bereits eine vollständige Befehlsdefinition. Sie erlaubt Ihnen nun, den Befehl `check_demo_a` in der Definition von Diensten (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) zu verwenden. Dabei wird dann die Befehlszeile

```
/usr/local/icinga/libexec/check_dummy 0 "Alles ist gut."
```

ausgeführt werden. Der Rückgabewert und die Ausgabe dieses Programms werden dann von Nagios/Icinga verarbeitet und in entsprechende Status und Ausgaben umgewandelt.

Test von Plugins: Testen Sie Plugins zuerst auf der Kommandozeile, bevor Sie diese einbinden. Führen Sie sie dazu einfach in der Eingabeaufforderung aus. Im eben angeführten Beispiel sähe das also folgendermaßen aus:

```
icinga@moni:~# /usr/local/icinga/libexec/check_dummy 0 "Alles ist gut."

 OK: Alles ist gut.
```

Eine ausführliche Anleitung dazu, wie Sie Plugins richtig testen, finden Sie im Icinga Wiki unter *https://wiki.icinga.org/display/testing/Icinga+Plugin+Testing*.

Um andere Programme für die Benutzung in Nagios/Icinga vorzubereiten, fügen Sie eine entsprechende Definition mit einem von Ihnen gewählten Namen und der entsprechend angepassten Kommandozeile hinzu. Beispiele für diverse Plugins und Ihre Anbindung finden Sie im Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen* und Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*.

Diskussion

Die Definition von Befehlen ist die erste Stufe der Einbindung von Programmen als Plugins in Nagios/Icinga. Sie führt ohne darauf aufbauende Dienst-Definitionen noch nicht zur Ausführung einer Prüfung. Abbildung 4-1 zeigt diesen Aufbau.

Im Beispiel haben Sie das Plugin check_dummy eingebunden und für die Verwendung in Dienst-Definitionen vorbereitet.

Das Plugin check_dummy

Beim Plugin `check_dummy` handelt es sich, wie der Name schon nahelegt, um ein einfaches Programm, das lediglich die übergebenen Parameter testet und entsprechende Rückgabewerte liefert. Dafür erwartet es als ersten Parameter eine Zahl, die den zurückzugebenden Status entsprechend der Plugin-API für Nagios/Icinga angibt. Auf diese gehen wir in Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)* genauer ein.

Als zweiten Parameter verarbeitet das Plugin optional einen Text für die Ausgabe. Dieser dient als textuelle Beschreibung oder Begründung des Status. Nagios/Icinga unterscheiden bei diesem Text verschiedene Varianten für mehrere Zeilen und optionale Performancedaten. Diese erläutern wir ebenfalls in Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*.

check_dummy: Wenn Sie die eben angeführte Kommando-Definition ein wenig modifizieren möchten, können Sie auch die Status 1 und 2 anstatt 0 verwenden. Den Text für die Ausgabe-Zeile können Sie nach Wunsch anpassen, allerdings sollten Sie mit Sonderzeichen vorsichtig sein. Insbesondere die Zeichen "|" und ";" haben hier eine besondere Bedeutung.

Platzhalter (Makros)

Die sogenannten Makros sind ein Ersetzungs-Mechanismus innerhalb von Nagios/Icinga, der es Ihnen bei der Befehlsdefinition ermöglicht, Platzhalter anzugeben, die bei der Ausführung des Befehls gegen Daten aus den Konfigurationsobjekten ausgetauscht werden. Für Sie sind Makros von Nutzen, wenn Sie

- wiederholt die gleiche und/oder
- dynamisch wechselnde und/oder
- sensible Informationen

in Befehlsdefinitionen verwenden möchten, wie zum Beispiel Pfadangaben, Adressen von Maschinen oder Passwörter. Am einfachsten lässt dies vielleicht anhand eines Beispiels zu erläutern. Bei der Installation Ihres Systems wurden Befehle für Tests an der lokalen Maschine eingerichtet, zum Beispiel folgende:

```
# 'check_local_users' command definition
define command{
       command_name                            check_local_users
       command_line                            $USER1$/check_users -w $ARG1$ -c $ARG2$
}
```

Diese Befehlsdefinition verwendet drei Makros: `$USER1$`, `$ARG1$` und `$ARG2$`, die zum Zeitpunkt der Ausführung also gegen entsprechende Inhalte ausgetauscht werden. Die hier verwendeten Makro-Typen und die entsprechenden Inhalte, mit denen sie ersetzt werden, sind folgende:

- Makro für benutzerdefinierte Inhalte (`$USERn$`)

  Bei `$USER1$` handelt es sich um ein sogenanntes $USER$-Makro. Dieses definieren Sie in der Datei resource.cfg im Konfigurationsverzeichnis von Nagios/Icinga; `$USER1$` ist standarmäßig auf den Pfad zum Plugin-Verzeichnis gesetzt und wird normalerweise bei jeder Befehlsdefinition eingesetzt. Die Definitionen in der Datei bestehen aus einfachen Zuweisungen, wie beispielsweise für das `$USER1$` aus dem Beispiel in unserer Installation "`$USER1$=/usr/local/icinga/libexec`". Unterstützt werden von den aktuellen Versionen von Nagios und Icinga 256 $USER$-Makros.

- Befehlsargument-Makro (`$ARGn$`)

  In Dienst-Definitionen können Sie Argumente definieren, die dann in den Befehlszeilen eingesetzt werden (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*). Solche Argumente referenzieren Sie dann mittels `$ARG1$`, `$ARG2$`, … . Im vorliegenden Beispiel sieht die Befehlsdefinition die Angabe von zwei Grenzwerten für die Status `WARNING` und `CRITICAL` vor, die als erstes beziehungsweise zweites Argument anzugeben sind.

Darüber hinaus stehen Ihnen weitere Makros für andere Inhalte zur Verfügung. In der folgenden Übersicht werden alle Makro-Gruppen aufgezeigt:

- Daten aus Konfigurationsobjekten

  Diese Makros erlauben Ihnen Zugriff auf die in den Konfigurationsobjekten gesetzten Optionen von:

  – Maschine (`$HOST...$`) und -gruppe (`$HOSTGROUP...$`)

  – Dienste (`$SERVICE...$`) und -gruppe (`$SERVICEGROUP...$`)

  – Kontakt (`$CONTACT...$`) und -gruppe (`$CONTACTGROUP...$`)

- On-Demand-Macros (`$HOST...:host_name$`, `$HOSTGROUP...:hostgroup_name$` und so weiter)

  Die Daten zu den Konfigurationsobjekten werden aus den Konfigurationsobjekten für die jeweils aktiven Maschinen/Dienste gesetzt werden. Sie können jedoch gezielt

Werte bestimmter Maschinen/Services abfragen, indem Sie eine entsprechende Referenz zum dem jeweiligen Makro erstellen.

- Benachrichtigungen (`$NOTIFICATION...$`)

  Mit diesen Makros können Sie in Benachrichtigungen auf Details wie beispielsweise den Empfänger oder die Information, um die die wievielte Benachrichtigung es sich handelt, zugreifen.

- Zusammenfassende Makros (`$TOTAL...$`)

  Hier stehen Ihnen zusammenfassende Makros, beispielsweise für die Gesamtzahl von Maschinen oder die Anzahl von kritischen Services und Ähnliches mehr, zur Verfügung.

- Zeit-Makros (`$...TIME$`)

  Diese Makros ermöglichen Ihnen unter anderem den Zugriff auf die bei der Ausführung aktuellen Zeit.

- Datei-Makros (`$...FILE$`)

  Diese ermöglichen Ihnen die Verarbeitung von Speicherpfaden und Dateinamen, wie Sie vom laufende Nagios/Icinga verwendet werden.

- Verschiedene Macros

  Hierzu zählen die bereits angeführten Befehlsargument-Makros (`$ARG...$`) und die Makros für benutzerdefinierte Variablen (`$USER...$`). Darüber hinaus stehen Ihnen einige weitere systemweite Daten wie beispielsweise die E-Mail des Administrators oder die Startzeit des Serverprozesses zur Verfügung.

- Benutzerdefinierte Variablen

  Benutzerdefinierte Variablen und wie Sie auf diese zugreifen, erläutern wir in einem eigenen Rezept (siehe Rezept 4.10, *Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen*). Mit diesem können Sie eben eigene Variablen definieren und dann ebenfalls über Makroersetzungen verwenden.

Eine vollständige Liste der von Ihrer spezifischen Version unterstützten Makros finden Sie in der jeweiligen Dokumentation zu Nagios beziehungsweise Icinga (siehe folgende Verweise).

Siehe auch

- Dokumentation zu den Befehlsdefinitionen unter Nagios: *http://nagios.sourceforge.net/docs/3_0/objectdefinitions.html#command* beziehungsweise unter Icinga: *http://docs.icinga.org/latest/de/objectdefinitions.html#objectdefinitions-command*
- Dokumentation zu Makros unter Nagios: *http://nagios.sourceforge.net/docs/3_0/macros.html* und *http://nagios.sourceforge.net/docs/3_0/macrolist.html* beziehungsweise unter Icinga: *http://docs.icinga.org/latest/de/macros.html und http://docs.icinga.org/latest/de/macrolist.html*

- Kapitel zur Erstellung eigener und Einbindung externer Plugins: Kapitel 7, *Erstellung eigener und Einbindung externer Plugins*
- Rezept zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Anleitung zum Testen von Plugins: *https://wiki.icinga.org/display/testing/Icinga+Plugin+Testing*
- Kapitel zur Überwachung lokaler Parameter: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezept zur Einführung in die von Nagios/Icinga verwendete API: Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*
- Rezept zur Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen: Rezept 4.10, *Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen*

4.4 Kontakte definieren (contact)

Problem

Sie möchten unterschiedliche Personen in Nagios/Icinga abbilden, um Zugriffsrechte oder Benachrichtigungen gezielt zu setzen/zuzustellen.

Lösung

Die Benutzer für Nagios/Icinga werden über die Konfigurationsoption contact definiert und eingerichtet. Diese Definitionen haben dabei einerseits die Funkion Benutzerkonten abzubilden (siehe Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*) und andererseits Benachrichtigungsoptionen für diese Kontakte festzulegen. Zunächst müssen Sie dazu in jedem Falle einen entsprechenden Kontakt definieren. Das Grundgerüst einer Kontakt-Definition ist:

```
define contact {
       option                                  wert
}
```

Zwischen den Klammern fügen Sie dabei wie im Folgenden beschrieben Optionen mit Ihren Werten ein.

Optionen von Kontakten

Wenn Sie eine Definition als Vorlage verwenden möchten, können Sie dies zunächst mit der Option `register` und dem Wert `0` festlegen. Sofern Sie in Ihrer Definition eine bereits vorhandene Vorlage nutzen möchten, können Sie dies weiter über die Option `use` mit dem Namen der zu nutzenden Vorlage als Wert angeben. Diese beiden allgemeinen Optionen sind optional und kombinierbar.

Die Definition eines Kontakts benötigt dann mindestens die folgenden Optionen, sofern diese nicht bereits über eine eingebundene Vorlage mit Werten versehen wurden:

- `contact_name`

  Name für den Kontakt, wie Sie ihn innerhalb der Konfigurationsoptionen verwenden möchten.

- `host_notifications_enabled` und `service_notifications_enabled`

  Verwenden Sie hier Wert `0`, um keine Benachrichtigungen über Maschinen (`host`) beziehungsweise Dienste (`service`) an diesen Kontakt zu senden. Mit dem Wert `1` können Sie die Optionen hingegen aktivieren.

- `host_notification_period` und `service_notification_period`

  Referenz auf ein in der Konfiguration definiertes Zeitfenster (`timeperiod`), das für die Maschinen- beziehungsweise Service-Benachrichtigungen verwendet wird (siehe Rezept 4.5, *Zeitperioden definieren (timeperiod)*).

- `host_notification_options` und `service_notification_options`

  Spezifikation, welche Zustände (siehe Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*) an diesen Kontakt gemeldet werden sollen.

- `host_notification_commands` und `service_notification_commands`

  Referenzen auf in der Konfiguration definierte Kommandos zum Versenden der Benachrichtigungen – bei Angabe mehrerer, durch Kommata separierter Kommandos, werden diese nacheinander ausgeführt (siehe Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*)

Darüber hinaus sollten Ihnen die folgenden optionalen Anweisungen bekannt sein:

- `email`

  Hier können Sie eine E-Mail-Adresse angeben, um sie in einem entsprechenden Befehl zu verwenden (siehe hierzu auch Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*)

- `can_submit_commands`

  Über den Wert `0` legen Sie fest, dass dieser Kontakt keine Befehle über das Classic-Web-Frontend übermitteln darf – mit dem Wert `1` darf der Benutzer Kommandos übermitteln.

- `retain_status_information` und `retain_nonstatus_information`

  Mit dieser Option können festlegen, dass die Speicherung der für diesen Kontakt relevanten Statusinformation einen Neustart überdauern können soll. Dies bedingt

die Speicherung der Status in einer Datei. Aktivieren Sie die Option mit dem Wert `1` oder deaktivieren Sie sie mit dem Wert `0`.

Es gibt noch ein paar zusätzliche Optionen, die hier nicht betrachten werden. Sie finden diese in der Dokumentation zu Ihrer Version.

Zuweisung von Kontakten

Die Zuweisung von Kontakten zu Maschinen und/oder Diensten erfolgt über die Definition derselben (siehe Rezept 4.1, *Maschinen einbinden (host)* beziehungsweise Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*). Sowohl für Benachrichtigungen als auch für den Zugriff über die Weboberfläche müssen die entsprechenden Kontakte jeweils referenziert worden sein.

Bei einer Maschine beziehungsweise bei einem Dienst referenzierte Kontakte werden dann bei jeder Statusänderung auf Benachrichtigungen überprüft. Sofern die Benachrichtigungsoptionen und Zeitfenster zutreffen, werden dann für jeden entsprechenden Kontakt die hinterlegten Benachrichtigungsbefehle ausgeführt. Wir gehen im Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten* ausführlich auf die Anpassung von Benachrichtigungsoptionen ein.

Wenn Sie Kontakten den Zugriff über die Weboberfläche gewähren möchten, müssen Sie die Benutzer zusätzlich mit einem Passwort in der Benutzerverwaltung des Webservers hinterlegen (siehe Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*). Jeder Benutzer erhält dann Zugriff auf alle Maschinen und Services, bei denen Sie ihn als Kontakt referenziert haben. Mit den Standardeinstellungen erhalten die Benutzer dabei weiter Zugriff auf alle Services, die zu einer Maschine gehören, zu denen der jeweilige Benutzer als Kontakt angegeben ist. Diese und weitere allgemeine Optionen haben wir in Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche* ausführlicher beschrieben.

Diskussion

Mit der Installation von Nagios/Icinga wurde unter dem Namen `generic_contact` eine Vorlage für Kontakte anglegt, die alle nötigen Option auf Standardwerte setzt. Sofern Sie mit der Ablage der Konfigurationsdateien noch nicht vertraut sind, sollten Sie zunächst Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien* lesen.

Wenn Sie die Standard-Vorlage verwenden, werden Sie immer über alles benachrichtigt. Für die Benachrichtigungen wird mittels eines entsprechenden Befehls (siehe Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*) E-Mail genutzt. Entsprechend können Sie wie im Folgenden angeführt mit minimalem Aufwand Kontakte mit Nutzung des E-Mail-Versands erstellen:

```
define contact {
       use                             generic_contacts
       contact_name                    John Doe
```

```
        alias JDoe
        email                                   doe@localhost
}
```

Da die allgemeine Vorlage mit einem Befehl für den E-Mail-Versand verknüpft ist, sollten Sie hier entsprechend aber auch nur die betreffenden Kontakte definieren. Wenn Sie etwa Webbenutzer mit Rechten versehen möchten, die aber gar keine Benachrichtigungen erhalten sollen, sollten Sie sich dafür eine getrennte Vorlage einrichten, wie im folgenden Absatz beschrieben.

Kontakte für GUI-Rechte ohne Benachrichtigungen

Sofern Sie unterschiedlichen Benutzern Zugriff auf die Weboberfläche erteilen möchten, müssen Sie diese zunächst als Kontakte definieren. Wenn diese Benutzer jedoch keine Benachrichtigungen per E-Mail erhalten sollen, sollten Sie sich eine eigene Vorlage für diesen Kontakt-Typ anlegen, wie zum Beispiel folgende:

```
define contact {
        register                                0
        use                                     generic_contacts
        name                                    template-contact-gui-user
        host_notifications_enabled              0
        service_notifications_enabled           0
}
```

Über die Option `register` definieren Sie die Definition als Vorlage und mit der Option `use` geben Sie an, dass die automatisch erstellte Vorlage hier verwendet werden soll. Abweichend von der Standard-Vorlage legen Sie dann mit `name` einen Namen für die Vorlage fest und definieren über die Optionen `host_notifications_enabled` und `service notifications enabled`, dass dieser Benutzer keine Benachrichtigungen für Maschinen/Dienste erhalten soll.

Aufbauend auf dieser Vorlage können Sie nun Nutzer nur für die Vergabe von Oberflächen-Rechten erstellen:

```
define contact {
        use                                     template-contact-gui-user
        name                                    buchleser
}
```

Damit sich ein auf diese Weise erstellter Benutzer auch an der Weboberfläche anmelden darf, müssen Sie für ihn noch ein Passwort wie in Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche* beschrieben erstellen. Wenn Sie diesen Kontakt dann in Maschinen/Services referenzieren, wird der Kontakt Zugriff auf die entsprechenden Elemente der Benutzeroberfläche erhalten. Sie können mit der eben beschriebenen Optionen `can_submit_commands` noch festlegen, ob der oder die Benutzer auch Kommandos über die Oberfläche absetzen können sollen, etwa eine außerplanmäßigen Prüfung veranlassen oder Prüfungen aussetzen.

CGI-Optionen: Welche Rechte genau eingeräumt werden, hängt von den für die CGI-Skipte gesetzten Werten ab, auf die wir in Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche* eingehen.

Siehe auch

- Dokumentation zum Konfigurationsobjekt `contact` bei Icinga: *http://docs.icinga.org/latest/de/objectdefinitions.html#contact* beziehungsweise für Nagios: *http://nagios.sourceforge.net/docs/nagioscore/3/en/objectdefinitions.html#contact*
- Rezept zur Definition von Zeitfenstern: Rezept 4.5, *Zeitperioden definieren (timeperiod)*
- Rezept zur Anpassung von Benachrichtigungen: Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*
- Rezepte zu Maschinen und Dienstdefinitionen: Rezept 4.1, *Maschinen einbinden (host)* und Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*.
- Rezept zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*

4.5 Zeitperioden definieren (timeperiod)

Problem

Sie möchten eingrenzen, wann Maschinen/Dienste überprüft werden, wann Benachrichtigungen verschickt werden oder wann Abhängigkeiten zwischen Maschinen/Diensten gültig sind.

Lösung

Definieren Sie entsprechende Zeitfenster als timeperiod und referenzieren Sie diese in den jeweiligen Maschinen (siehe Rezept 4.1, *Maschinen einbinden (host)*), Diensten (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*), Kontakten (siehe Rezept 4.4, *Kontakte definieren (contact)*) oder Abhängigkeiten (siehe Rezept 4.9, *Abbilden von Abhängigkeiten (host- und service-dependencies)*). Als zwingende Optionen müssen Sie dabei einen Namen und einen Alias angeben. Eine minimale Definition eines Zeitfensters sieht so aus:

```
define timeperiod {
       timeperiod_name                 nie
       alias                           Zeitfenster ohne Zeiten
}
```

Da dieses Zeitfenster keine Zeiten angibt, trifft es nie zu.

Definition der Standard-Woche

Die Angabe der gültigen Zeiten erfolgt zunächst über die Wochentage, für die Sie die gültige Zeiten angeben. Wenn Sie einen Wochentag auslassen, gilt dieser als nicht belegt. Bei der Installation wurden zwei Zeitfenster angelegt. Das erste mit dem Namen `24x7` definiert alle Zeiten an allen Tagen:

```
define timeperiod {
       timeperiod_name                 24x7
       alias                           24 Hours A Day, 7 Days A Week

       sunday                          00:00-24:00
       monday                          00:00-24:00
       tuesday                         00:00-24:00
       wednesday                       00:00-24:00
       thursday                        00:00-24:00
       friday                          00:00-24:00
       saturday                        00:00-24:00
}
```

Das andere hingegen definiert im hier vorliegenden Beispiel Arbeitszeiten:

```
define timeperiod {
       timeperiod_name                 workhours
       alias                           Normal Work Hours

       monday                          09:00-17:00
       tuesday                         09:00-17:00
       wednesday                       09:00-17:00
       thursday                        09:00-17:00
       friday                          09:00-17:00
}
```

Standard-Konfiguration: Beachten Sie, dass die Beispielkonfiguration diese zweite, eingeschränkte Periode bei der Definition der Vorlagen `linux-server` und `generic-printer` für die Option `notification_period` verwendet. Entsprechend werden von diesen abgeleitete Maschinen-Definitionen nur zu diesen Zeiten Benachrichtigungen empfangen. Insbesondere betrifft dies bei der generierten Konfiguration die Maschine `localhost`, also die Monitoring-Maschine selbst.

Die Angabe der Tageszeiten können Sie dabei auch aus mehreren, durch Kommata separierten Zeiträumen zusammensetzen. Wenn also etwa Mittagspausen berücksichtigt werden sollen, funktioniert das so:

```
define timeperiod {
       timeperiod_name                 workhours
       alias                           Normal Work Hours

       monday                          09:00-12:00,13:00-18:00
       tuesday                         09:00-12:00,13:00-18:00
       wednesday                       09:00-12:00,13:00-18:00
```

```
        thursday                        09:00-12:00,13:00-18:00
        friday                          09:00-12:00,13:00-18:00
}
```

Weitere Definitionen von Zeiten

Zusätzlich zu auf diese Weise definierten Wochen-Zeiten ermöglicht Ihnen Nagios/Icinga, weitere Zeiträume festzulegen, um so beispielsweise Saisonzeiten oder periodische Zeiten wie Backup-Läufe zu definieren. Eine Auswahl wichtiger Schreibweisen haben wir für Sie in Tabelle 4-1 anhand von Bespielen aufgeführt.

Tabelle 4-1: Besondere Zeitfenster

Angabe	Bedeutung
2012-12-24	Datum in Schreibweise nach ISO-Norm
2012-12-24 – 2012-12-26	Zeitraum mit Start- und Enddatum
day 3	Jeder 3. eines jeden Monats
day 3 – 5	Jeder 3.-5. eines jeden Monats
sunday 3	Jeder 3. Sonntag eines jeden Monats
sunday 3 may	Jeder 3. Sonntag im Mai
2012-01-01 / 2	Jeder 2. Tag ab dem 1.1.2012

Zu diesen Angaben gehört jeweils noch die Spezifikation der Tageszeit als Wert, wie oben bei den Wochenzeiten erläutert, hier am Beispiel für Heiligabend:

```
define timeperiod {
        timeperiod_name                 heiligabend
        alias                           heiliger freier Abend

        december                        24 16:00-24:00
}
```

Es gibt hierzu noch weitere Schreibweisen, insbesondere um die in der Tabelle aufgeführten Zeiten noch weiter auf bestimmte Zeiträume einzugrenzen. Schauen Sie dazu bei Bedarf bitte in die Dokumentation zu Ihrer Version.

Einbindung von Ausnahme-Zeiträumen

Laut Dokumentation ist vorgesehen, von vorgegebenen Zeiträumen Ausnahmen auszuschließen. Sie sollen so beispielsweise Urlaubszeiten oder Feiertage wie den dargestellten Heiligabend von Ihrer bisherigen Definition ausnehmen lassen können. Zunächst können Sie solche Ausnahmen in der bisherigen Definition über die Konfigurationsoption `exception` angeben.

Alternativ können Sie Ausnahmen auch als eigene Zeitfenster definieren und dann über die Konfigurationsoption `exclude` in der Definition des eigentlichen Zeitfensters referenzieren.

Diskussion

Zeitfenster können Ihnen helfen, unnötige Benachrichtigungen oder Abfragen zu vermeiden. Nehmen wir einen Bereitschaftsdienst an, bei dem Sie nur zu Ihren Bereitschaftszeiten Benachrichtigungen erhalten möchten. Sie können diese Schichten dann als Zeitfenster definieren und den jeweiligen Kontakten über die entsprechenden Optionen zuweisen (siehe dazu Rezept 4.4, *Kontakte definieren (contact)*). Selbstverständlich können Sie dies auch spezifisch für ausgewählte Maschinen oder Dienste konfigurieren. Es ist also möglich, Ihren während einer Bereitschaft jeweils verantwortlichen Mitarbeitern (gegebenenfalls über eine Gruppe, siehe Rezept 4.8, *Kontakte in Gruppen zusammenfassen (contact-group)*) ein Zeitfenster zuweisen. Die Mitarbeiter erhalten außerhalb der regulären Arbeitszeit damit nur Benachrichtigungen zu den jeweils definierten Bereitschaftszeiten.

Siehe auch

- Dokumentation zu Zeitfenster bei Icinga: *http://docs.icinga.org/latest/de/timeperiods.html* beziehungsweise für Nagios: *http://nagios.sourceforge.net/docs/nagioscore/3/en/objectdefinitions.html#timeperiod*
- Objekte, in denen Sie Zeitfenster nutzen können: Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*, Rezept 4.4, *Kontakte definieren (contact)* und Rezept 4.9, *Abbilden von Abhängigkeiten (host- und service-dependencies)*
- Rezept zur Erstellung eigener Benachrichtigungen: Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*
- Rezept zur Abbildung von Abhängigkeiten: Rezept 4.9, *Abbilden von Abhängigkeiten (host- und service-dependencies)*

4.6 Maschinen zu Gruppen zusammenfassen (host-group)

Problem

Sie möchten Maschinen in der Oberfläche gruppiert anzeigen oder Konfigurationsanweisungen gleichzeitig ganzen Gruppen von Maschinen zuweisen.

Lösung

Sie können sich Maschinengruppen der entsprechenden Maschinen definieren. Diese werden dann automatisch in der Weboberfläche bei den Maschinengruppen dargestellt und Sie können sie auch für die Zuweisung von Konfigurationsoptionen verwenden. Die Definition von Maschinengruppen erfolgt über das Konfigurationsobjekt hostgroup in folgender Struktur:

```
define hostgroup {
       option                                  wert
}
```

Als Pflichtoptionen müssen die dabei für jede Maschinengruppe die folgenden Optionen angeben:

- `hostgroup_name`
  Hiermit legen Sie den Namen der Maschinengruppe fest.
- `alias`
  Mit dieser Option legen Sie einen beschreibenden Namen für die Maschinengruppe fest.

Damit hätten Sie dann eine zunächst leere Maschinengruppe definiert, die Sie mit Mitgliedern füllen können indem Sie sie bei Maschinendefinitionen über die Option `hostgroups` referenzieren (siehe Rezept 4.1, *Maschinen einbinden (host)*).

Optionen zum Angeben von Mitgliedern

Mit den folgenden Optionen können Sie direkt bei der Definition der Gruppe Maschinen als Mitglieder für diese spezifizieren:

- `members`
  Geben Sie hier eine durch Kommata separierte Liste mit Namen von Maschinen an, die der Gruppe angehören sollen.
- `hostgroup_members`
  Mit dieser Option können Sie eine durch Kommata separierte Liste mit Namen von Maschinengruppen angeben, deren Mitglieder dieser Gruppe angehören sollen.

Auch wenn Sie diese Optionen nutzen, können Sie die Gruppe weiterhin über Referenzierung bei der Definition von Maschinen mit zusätzlichen Mitgliedern füllen. In Rezept 4.11, *Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo)* stellen wir Ihnen weitere Optionen zur Anpassung der Darstellung und Integration mit externen Systemen vor.

Diskussion

Nutzen Sie Maschinengruppen, um Ihre Konfiguration zu vereinfachen. Bilden Sie leere Gruppen und weisen Sie jede Maschine in Ihrer Definition passenden Gruppen zu. Binden Sie Services dann an die Maschinengruppen statt an die Maschinen, um sie automatisch auf allen zur Gruppe gehörenden Maschinen auszuführen. Auf diese Weise können Sie später Maschinen hinzufügen und nur durch die Zuweisung zur passenden Gruppe eine Prüfung aller entsprechend hinterlegten Services realisieren. Gleichzeitig erhalten Sie in der Oberfläche die Möglichkeit, über diese Gruppen auf die Maschinen zuzugreifen und ihre Status und Services einzusehen.

Siehe auch

- Rezept zur Einbindung einer Maschine: Rezept 4.1, *Maschinen einbinden (host)*
- Rezept zur Integration mit Fremd-Systemen: Rezept 4.11, *Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo)*

4.7 Services zu Gruppen zusammenfassen (service-group)

Problem

Sie verwenden in Ihrer Installation viele unterschiedliche Dienste und möchten diese in der Weboberfläche zu Gruppen zusammenfassen.

Lösung

Zu diesem Zweck bietet Nagios/Icinga das Konfigurationsobjekt servicegroup. Sie definieren Dienstgruppen in der folgenden Form:

```
define servicegroup {
       option                                  wert
}
```

Um eine gültige Dienstgruppe zu definieren, müssen Sie die folgenden Pflicht-Optionen definieren:

- `servicegroup_name`

  Kurzname der Dienstgruppe
- `alias`

  Beschreibender Name der Dienstgruppe

Eine lediglich mit diesen Pflicht-Optionen definierte Gruppe wäre dabei zunächst leer. Sie können Sie über die `servicegroups` Option bei der Definition von Diensten füllen (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*). Alternativ können Sie die folgenden Optionen verwenden, um direkt bei Definition der Gruppe Mitglieder hinzuzufügen:

- `members`

  Durch Kommata separierte Liste von wiederum durch Kommata getrennten Tupeln, die jeweils erst den Kurznamen einer Maschine und anschließend einen Dienst auf dieser Maschine angeben.
- `servicegroup_members`

  Durch Kommata separierte Liste von Dienstgruppen, deren Mitglieder dieser Gruppe angehören sollen.

Als weitere Option können Sie `notes` verwenden, um einen Text zur Anzeige der Detailansicht der Gruppe zu hinterlegen.

Diskussion

Mit einer steigenden Anzahl an Maschinen und Diensten werden Sie die Möglichkeit, sich passende Übersichtsdarstellungen schaffen zu können, zu schätzen lernen. Wenn Sie sich eine Dienstgruppe mit den Diensten, die etwa ein bestimmtes Problem betreffen, definieren, haben Sie in der Detailansicht der entsprechenden Gruppe alle das Problem betreffenden Status zusammengefasst. Sie können beispielsweise die Dienste PING zweier Maschinen localhost und hcksp0 mit der folgenden Definition gruppieren:

```
define servicegroup {
       servicegroup_name                    sg-demo
       alias                                Demonstration
       members                              localhost,PING , hcksp0,PING
}
```

Wenn Sie einen bestimmten Dienst auf allen Maschinen, auf denen er läuft, zusammenfassen möchten, können Sie anstelle der Spezifikation einer Maschine auch den Platzhalter "*" verwenden:

```
define servicegroup {
       servicegroup_name                    sg-demo
       alias                                Demonstration
       members                              *,PING
}
```


Siehe auch

- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*

4.8 Kontakte in Gruppen zusammenfassen (contact-group)

Problem

In Ihrer Installation verwalten Sie viele Kontakte für Benachrichtigungen und/oder den Zugang zur Oberfläche, und diese wechseln häufig. Sie möchten deshalb die Arbeit mit den entsprechenden Kontaktdefinitionen vereinfachen.

Lösung

Fassen Sie solche Kontakte (Konfigurationsobjekt contact, siehe Rezept 4.4, *Kontakte definieren (contact)*) in geeigneter Weise zu einer Kontaktgruppe (Konfigurationsobjekt contactgroup) zusammen. Sie können diese Gruppen dann sozusagen als Rollen verwenden, die Sie den Maschinen beziehungsweise Diensten zuweisen. Änderungen an allen betroffenen Maschinen/Diensten können Sie dann wieder an einer zentralen Stelle durchführen.

Kontaktgruppen definieren Sie mit dem Konfigurationsobjekt:

```
define contactgroup {
       option                                  wert
}
```

Als Pflicht-Optionen müssen Sie dabei für jede Kontaktgruppe mindestens Werte für die folgenden Optionen definieren:

- `contactgroup_name`

  Geben Sie hier einen Namen für die Kontaktgruppe an, mit dem Sie diese innerhalb der Konfigurationsdateien referenzieren.

- `alias`

  Geben Sie hier zusätzlich einen beschreibenden Namen für die Kontaktgruppe an.

Sie können also eine Kontaktgruppe ohne Mitglieder erstellen, die zunächst nur einen Namen aufweist:

```
define contactgroup {
       contactgroup_name                       beispiel
       alias                                   Eine leere Gruppe
}
```

Dies macht Sinn, wenn Sie die Kontaktgruppe über die Definitionen von Kontakten referenzieren und so befüllen. Ansonsten können Sie der Kontaktgruppe mit den folgenden, optionalen Optionen direkt bei der Definition Mitglieder zuweisen:

- `members`

  Geben Sie hier eine durch Kommata separierte Aufzählung von Kontakten an, die der Kontaktgruppe angehören sollen.

- `contactgroup_members`

  Geben Sie hier eine durch Kommata separierte Aufzählung von Kontaktgruppen an, die dieser Kontaktgruppe angehören sollen.

Um etwa eine als Rolle zu verwendende Kontaktgruppe mit dem bei der Installation angelegten Benutzer `icinga-admin` als Mitglied anzulegen, könnten Sie die folgende Definition verwenden:

```
define contactgroup {
       contactgroup_name                       role-security-admin
       alias                                   Alle Benutzer die Sicherheit überwachen

       members                                 icinga-admin
}
```

Beachten Sie hierbei, dass die referenzierten Kontakte und/oder Kontaktgruppen vor dem nächsten Neustart vorhanden sein müssen. Ansonsten kann Nagios/Icinga die Referenz nicht auflösen und meldet einen Fehler.

Diskussion

Kontaktgruppen können Ihnen sehr dabei helfen, Ihre Konfiguration pflegeleicht zu gestalten. Verwenden Sie sie ruhig auch dann, wenn nur wenige Benutzer vorhanden sind – dann können Sie später möglicherweise hinzukommende Benutzer leicht einpflegen und über diese Gruppen einbinden. Sie können auf diese Weise beispielsweise Gruppen für Abteilungen pflegen und neue Mitarbeiter dann einfach durch Aufnahme in die Gruppe einbinden. Wenn Sie hingegen in Maschinen einzelne Kontakte referenziert haben, müssten Sie neue Mitarbeiter an allen entsprechenden Stellen eintragen. Über die Option `contactgroup_members` können Sie die Gruppen weiter zusammenfassen, etwa um eine Gruppe aller Mitarbeiter zu bilden.

Siehe auch

- Rezept zur strukturierten Ablage der Konfigurationsdateien: Rezept 3.6, *Aufbau und strukturierte Ablage der Konfigurationsdateien*
- Rezept zur Definition von Kontakten: Rezept 4.4, *Kontakte definieren (contact)*
- Rezept zur Benutzerverwaltung für Zugriffe auf die Weboberfläche: Rezept 3.1, *Benutzerverwaltung für Zugriffe auf die Weboberfläche*
- Rezept zur Erstellung eigener Benachrichtigungen: Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*

4.9 Abbilden von Abhängigkeiten (host- und service-dependencies)

Problem

Sie möchten Abhängigkeiten zwischen Ihren Maschinen und Diensten abbilden, um unnötige Benachrichtigungen zu vermeiden.

Lösung

Nagios/Icinga bietet zur Bildung von Abhängigkeiten eigene Objekttypen für Maschinen und Dienste. Diese erlauben es Ihnen nach Bedarf, Abhängigkeiten zu definieren, die dann entsprechend berücksichtigt werden. Dies bedeutet, dass eine abhängige Maschine dann gar nicht mehr geprüft wird, wenn die Maschine von dem diese abhängt, nicht erreichbar ist. Analog funktionieren die Abhängigkeiten für Dienste.

Wir zeigen Ihnen im Folgenden, wie Sie entsprechende Abhängigkeiten definieren und nutzen können. In der Diskussion führen wir Beispiele für die Verwendung an.

Abhängigkeiten zwischen Maschinen (*hostdependency*)

Um eine Maschine von einer anderen abhängig zu machen, können Sie zunächst eine Eltern-Kind-Beziehung verwenden, wie in Rezept 4.1, *Maschinen einbinden (host)* beschrieben (Konfigurationsoption `parents` in der Maschinen-Definition). Für Maschinen sollten Sie sogar primär diesen Mechanismus einsetzen, da Nagios/Icinga diese Abhängigkeiten auch anderweitig einsetzt: bei der Bestimmung, ob eine Maschine überhaupt erreichbar ist, mittels der sogenannten Erreichbarkeitslogik und bei der Darstellung in der Status-Map (siehe Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*).

Sie können zusätzlich zu diesem Mechanismus Konfigurationsobjekte vom Typ hostdependency verwenden, um erweiterte, von Ihnen definierte Abhängigkeiten festzulegen. Diese erlauben es Ihnen beispielsweise, eine Maschine in einem Schritt von einer ganzen Maschinengruppe abhängig zu machen oder die betroffenen Status (getrennt für die Ausführung und Benachrichtigung) exakt festzulegen. Dadurch erhalten Sie sehr viel mehr Kontrolle über die Art der Abhängigkeit. Allerdings wird diese von der Erreichbarkeitslogik und der Status-Map eben nicht berücksichtigt.

Sie definieren eine solche Abhängigkeit über das Konfigurationsobjekt `hostdependency` wie folgt:

```
define hostdependency {
       option                                wert
       [...]
}
```

Als Pflichtoptionen müssen Sie dabei die folgenden Optionen mit Werten versehen:

- `dependent_host_name` oder `dependent_hostgroup`

  Legen Sie hier den Namen der abhängigen Maschine(n) fest. Separieren Sie mehrere Angaben durch Kommata.

- `host_name` oder `hostgroup_name`

  Legen Sie hier die Maschine(n) oder Maschinengruppe(n) fest, von denen die zuvor definierte Maschine abhängt. Erneut können Sie mehrere Angaben durch Kommata voneinander trennen.

Darüber hinaus können Sie mit den folgenden Optionen auf Wunsch die Art der Abhängigkeit genauer definieren:

- `inherits_parent`

  Mit dieser Option können Sie festlegen, ob die Abhängigkeit vererbt werden soll (Wert `0` für nein, `1` für ja). Wenn Sie diese Option aktivieren, werden Abhängigkeiten der Maschine berücksichtigt, die Sie mit `dependent_host_name` oder `dependent hostgroup` definiert haben.

- `execution_failure_criteria` und `notification_failure_criteria`

  Mit diesen Optionen legen Sie die Zustände fest, die die Abhängigkeit für die Ausführung der Prüfung (execution) beziehungsweise die Benachrichtigungen (notifica-

tion) berücksichtigen soll. Die möglichen Zustände für Maschinen haben wir Ihnen in Tabelle 4-2 zusammengestellt. Sobald die über `dependent_host_name` oder `dependent_hostgroup` definierte(n) Maschine(n) einen der hier angegebenen Status aufweisen, wird die Ausführung beziehungsweise Benachrichtigung für die mit `host_name` oder `hostgroup_name` angegebene(n) Maschine(n) unterdrückt.

- `dependency_period`

  Außerdem können Sie die Abhängigkeit noch in der zeitlichen Gültigkeit einschränken, indem Sie mit dieser Option ein Zeitfenster (siehe hierzu Rezept 4.5, *Zeitperioden definieren (timeperiod)*) referenzieren, in denen die Abhängigkeit gelten soll.

Tabelle 4-2: Statuscodes für Abhängigkeiten bei Maschinen

Status	bei Maschinen
n (none)	keine
d (down)	Gerät läuft nicht
u (unreachable)	Gerät nicht erreichbar (sofern Eltern-Kind-Beziehung gesetzt)
o (up)	Gerät ist erreichbar

Nehmen wir einmal an, die Maschinen `suse` und `fred` wären abhängig von einer Maschine `moni`. Dies wäre beispielsweise der Fall, wenn es sich bei `moni` um einen Virtualisierungsserver handelte, auf dem die anderen beiden Rechner laufen. Falls `moni` durch eine Ausfall den Status `DOWN` (`d`) oder `UNREACHABLE` (`u`) erhält, soll das Monitoringsystem nur Benachrichtigungen für den Ausfall von `moni` versenden, nicht aber für die beiden virtuellen Maschinen. Eine passende Definition einer solchen Abhängigkeit würde dann beispielsweise wie folgt aussehen:

```
define hostdependency{
       host_name                       moni
       dependent_host_name             suse,fred
       notification_failure_criteria   d,u
}
```

Auf diese Weise können Sie sehr genau festlegen, von welchen anderen Maschinen eine bestimmte Maschine abhängig ist, bei welchen Status diese geprüft und bei welchen Benachrichtigungen versendet werden sollen.

Abhängigkeiten zwischen Diensten (*servicedependency*)

Um Abhängigkeiten zwischen Diensten abzubilden, gibt es keinen der Eltern-Kind-Beziehungen bei Maschinen entsprechenden Mechanismus, außer natürlich der Bedingung, dass die Maschine auf der der Dienst angesiedelt ist, auch erreichbar sein muss. Deshalb gibt es die Objekte vom Typ servicedependency, mit denen Sie manuell Abhängigkeiten zwischen Diensten einpflegen können. Diese definieren Sie in der folgenden Form:

```
define servicedependency {
       option                          wert
       [...]
}
```

Als Pflichtoptionen müssen Sie dabei die folgenden Optionen mit Werten versehen:

- `dependent_host_name` oder `dependent_hostgroup`

  Legen Sie hier den Namen der abhängigen Maschine(n) fest. Trennen Sie mehrere Angaben durch Kommata voneinander.

- `dependent_service_description`

  Neben der Angabe der Maschine müssen Sie für einen Dienst natürlich auch einen solchen festlegen. Hierzu dient diese Option, mit der Sie zunächst über dessen Namen den abhängigen Dienst angeben.

- `host_name` oder `hostgroup_name`

  Legen Sie hier die Maschine(n) oder Maschinengruppe(n) fest, von denen der der zuvor festgelegte Dienst abhängt. Erneut können Sie mehrere Angaben durch Kommata separieren.

- `service_description`

  Zusätzlich zur Angabe der Maschine müssen Sie bei Diensten auch einen Dienst angeben, von dem der oben festgelegt Dienst abhängig ist.

Die weiteren Optionen entsprechen denen, die bereits vorher schon bei den Abhängigkeiten zwischen Maschinen aufgeführt wurden. Dies sind also wieder

- `inherits_parent`,
- `execution_failure_criteria` und `notification_failure_criteria` sowie
- `dependency_period`.

Eine Beschreibung dieser Optionen finden Sie weiter vorne bei den Maschinen-Abhängigkeiten. Hier unterscheiden sich lediglich die Status, die Sie mit `execution failure criteria` und `notification_failure_criteria` angeben können. Die für Dienste relevante Statuscodes haben wir Ihnen in Tabelle 4-3 zusammengestellt.

Tabelle 4-3: Statuscodes für Abhängigkeiten bei Diensten

Status	bei Maschinen
n (none)	keine
o (ok)	Dienst meldet OK
w (warning)	Dienst meldet WARNING
c (critical)	Dienst meldet CRITICAL
u (unknown)	Dienst-Status ist unbekannt
p (pending)	Dienst wurde noch nicht geprüft

Um hier ein Beispiel zu nennen, nehmen wir an, auf der Maschine `fred` liefe ein SNMP-Dienst. Diesen Dienst nutzen Sie, um lokale Werte des entfernten Systems abzufragen. Für den Fall, dass der SNMP-Dienst auf `fred` nicht mehr erreichbar ist, wollen Sie keine weiteren Abfragen von Diensten ausführen, die über den SNMP-Dienst ausgeführt werden, und Sie wollen auch keine Benachrichtigungen über den Zustand derselbigen.

Eine entsprechende Definition würde dann wie folgt aussehen.

```
define servicedependency{
       host_name                            fred
       service_description                  check_snmp3
       dependent_host_name                  fred
       dependent_service_description        check_snmp_int-eth0-bw
       execution_failure_criteria           c,u
       notification_failure_criteria        c,u
 }
```

Diskussion

Die Möglichkeit über die Objekte `hostdependency` und `servicedependency` manuell Abhängigkeiten zu definieren, ist nur unter spezifischen Umständen notwendig. Typischerweise werden Sie zunächst Abhängigkeiten über die Eltern-Kind-Beziehungen zwischen Maschinen abbilden, die dann auch von der Erreichbarkeitslogik verwendet werden und deshalb von besonderem Nutzen sind. Für Dienste besteht immer automatisch eine Abhängigkeit zu der Maschine, mit der diese verknüpft sind.

Die zusätzlichen Abhängigkeiten sind Ihnen insbesondere in den folgenden Fällen von Nutzen:

- Alle Abhängigkeiten zwischen Diensten (außer der zu der entsprechenden Maschine) müssen Sie auf diese Weise manuell definieren, beispielsweise wenn ein Anwendungssystem auf eine laufende Datenbank auf einer ganz anderen Maschine angewiesen ist.
- Sofern die standardmäßig verwendeten Status für einen Anwendungsfall bei Ihnen nicht zutreffend sind, können Sie hier manuell festlegen, welche Status relevant sind.
- Wenn Sie Abhängigkeiten oder die dazugehörigen Benachrichtigungen zeitlich einschränken möchten, können Sie dazu die die Definition von Zeitfenstern nutzen.

Entsprechend bieten Ihnen diese Abhängigkeiten vielfältige Möglichkeiten, um Ihre Konfiguration zu erweitern und an Ihre Bedürfnisse anzupassen.

Siehe auch

- Dokumentation zu Abhängigkieten unter Icinga: *http://docs.icinga.org/latest/de/dependencies.html* beziehungsweise unter Nagios: *http://nagios.sourceforge.net/docs/nagioscore/3/en/dependencies.html*
- Beschreibung der Eltern-Kind-Beziehungen im Rezept zur Einbindung von Maschinen: Rezept 4.1, *Maschinen einbinden (host)*
- Beschreibung der Oberflächen und der Status-Map, die die Eltern-Kind-Beziehungen verwendet: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*

4.10 Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen

Problem

Sie möchten bei Objektdefinitionen nicht vorgesehene Daten verwenden.

Lösung

Nagios/Icinga ist darauf ausgelegt, auch nicht von den Entwicklern vorgesehene Daten verarbeiten zu können. Um Werte einzusetzen, für die es bislang keine Konfigurationsoption gibt, können Sie benutzerdefinierte Variablen verwenden. Dies ist sehr einfach möglich: Sie legen sich eigene Variablen an, indem Sie ihrem Namen einen Unterstrich '_' voranstellen und einen beliebigen Namen nutzen, also etwa `_INVENTARNR`.

Großschreibung: Sie können bei den Namen auch die Kleinschreibung nutzen. Da bei Namen von selbst definierten Variablen während der Ausführung aber stets eine Umwandlung in Großschreibung erfolgt, empfehlen wir Ihnen, diese auch gleich bei der Definition zu verwenden.

Auf diese Weise angelegte Optionen können Sie dann mit einem entsprechenden Wert versehen. Um also beispielsweise eine Maschine mit der Inventarnummer `0815` auszuzeichnen, fügen Sie der Konfiguration eine entsprechende Zeile hinzu:

```
define host {
       [...]
       _INVENTARNR                          0815
}
```

Nun ist die Aufnahme eines solchen Datums typischerweise nur interessant, wenn Sie sie in irgendeiner Form benötigen, wie insbesondere bei der Durchführung von Prüfungen. Der Zugriff auf selbst definierte Variablen erfolgt – wie bei den herkömmlichen Variablen auch – über Makros (siehe hierzu Rezept 4.3, *Befehle hinzufügen (command)*). Dabei müssen Sie allerdings eine Besonderheit beachten: Um Konflikte zwischen Variablen in den möglichen Objekttypen für Maschinen (`host`, siehe Rezept 4.1, *Maschinen einbinden (host)*), Dienste (`service`, siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) und Kontakte (`contact`, siehe Rezept 4.4, *Kontakte definieren (contact)*) zu vermeiden, müssen Sie der jeweiligen Variablen den entsprechenden Konfigurationstyp (`HOST`, `SERVICE` beziehungsweise `CONTACT`) voranstellen. Wenn Sie also die oben definierte Inventarnummer über eine Makro-Ersetzung in einem Kommando verwenden möchten, müssen Sie dort `$_HOST_INVENTARNR$` schreiben.

Diskussion

Die Erweiterung über benutzerdefinierte Variablen ermöglicht Ihnen, quasi beliebige Daten zu der Konfiguration hinzuzufügen und zu verwenden. Wir möchten Ihnen im Folgenden einige Anwendungsbeispiele hierzu aufführen:

- Das hier angeführte Beispiel einer Inventarnummer könnten Sie beispielsweise verwenden, um entsprechende Daten zu der jeweiligen Maschine aus einer Datenbank abzurufen. Eine ähnliche Anwendung ist die Pflege der MAC-Adressen als benutzerdefinierte Variablen, um entsprechende Daten etwa in Log-Dateien zu finden.
- Wenn Geräte an mehrere Netzwerke parallel angebunden sind, können Sie mit benutzerdefinierten Variablen die jeweils zusätzlichen Adressen und/oder Domänen für Maschinen in Ihre Konfiguration aufnehmen und dann in Kommandos für entsprechende Prüfungen einsetzen. Dies bietet sich beispielsweise an, wenn Sie ein separates Management-Netzwerk einsetzten, wie in Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen* betrachtet.
- Sie können über benutzerdefinierte Variablen auch Zugangsdaten wie Passwörter oder den Namen einer SNMP Community bereitstellen. Dies kann von Vorteil sein, wenn Sie die Pflege durch unterschiedliche Personen zulassen möchten, jedoch gleichzeitig eine Trennung der Zugriffsrechte benötigen. Über entsprechende Rechte auf unterschiedlichen Konfigurationsverzeichnissen lässt sich dies auf einfache Weise realisieren. Beachten Sie dabei aber, dass entsprechende Passwörter in Log-Dateien und Prozesslisten auftauchen können. Generell sollten Sie Zugangsdaten deshalb über die Benutzermakros verwalten und verwenden (siehe Rezept 4.3, *Befehle hinzufügen (command)*).
- Für Kontakte können Sie über benutzerdefinierte Variablen zusätzliche Details wie etwa Benutzernamen für Dienste, zum Beispiel einen Chat, verwalten. Über entsprechende Benachrichtigungskommandos (siehe Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*) können Sie dann anstelle der typischerweise genutzten E-Mails eben Nachrichten in diesen Diensten versenden.
- Icinga-Web erlaubt spezielle Berechtigungen anhand von benutzerdefinierten Variablen. So können Berechtigungen gezielt eingeschränkt werden, ohne dass auf Kontakt und Gruppenebene getrickst werden muss.

Wie Sie sehen, können Ihnen benutzerdefinierte Variablen auf höchst unterschiedliche Weise von Nutzen sein. Unsere Aufzählung erhebt dabei keinen Anspruch auf Vollständigkeit. Wenn Sie eine Aufgabe mit Nagios/Icinga lösen möchten und zunächst keinen Weg sehen, um Ihr Ziel zu erreichen, so sind es eventuell eben diese Variablen, die Ihnen eine entsprechende Möglichkeit verschaffen.

Siehe auch

- Rezept zur Kommandodefinition mit Erläuterung der Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezepte zu den relevanten Konfigurationsobjekten: Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* und Rezept 4.4, *Kontakte definieren (contact)*
- Das Rezept, in dem die Anbindung über mehrere Netzwerke beschrieben wird: Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*
- Das Rezept zur Anpassung der Benachrichtigungen: Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*

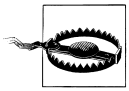
4.11 Erweiterung um Informationen und Integration mit Fremd-Systemen (hostextinfo und serviceextinfo)

Problem

Sie möchten in den Oberflächen zusätzliche Informationen zu Maschinen oder Diensten anzeigen.

Lösung

Icinga bietet eine Reihe von Konfigurationsoptionen, um zusätzliche Informationen in Form von Text oder Links zu integrieren. Diese können Sie im Rahmen der Definition von Maschinen (siehe Rezept 4.1, *Maschinen einbinden (host)*) und Diensten (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) angeben, oder (derzeit noch) in Form von zusätzlichen Konfigurationsobjekten als hostextinfo beziehungsweise serviceextinfo. Beachten Sie dabei bitte, dass die hier beschriebenen Mechanismen am besten für beständige Informationen geeignet sind, die sich nicht ständig ändern. Für temporäre Notizen zu Diensten oder Maschinen können Sie die hierzu vorgesehenen Funktionen der Oberflächen verwenden, die wir Ihnen in Rezept 3.13, *Kommentare, Acknowledgments und Downtimes nutzen* beschrieben haben.



Objekt versus Option: Beachten Sie bitte, dass die hier beschriebene Einbindung über eigene Objekte zwar noch funktioniert, aber offiziell veraltet ist. Sie sollten die Anweisungen also besser direkt in den eigentlichen Objekten (etwa zum Gerät oder Dienst) einfügen, damit Ihre Konfiguration auch bei einer Migration zu einer neueren Version noch funktionieren wird. Wir zeigen Ihnen die Möglichkeiten hier nur deshalb über die eigenen Objekttypen, weil sich dies bei der Beschreibung in einem eigenen Rezept anbietet.

Um die in der Folge beschriebenen Optionen über die zusätzlichen Konfigurationsobjekte zu verwenden, nutzen Sie für Maschinen die Syntax:

```
define hostextinfo {
       option                                  wert
       [...]
}
```

beziehungsweise für Dienste die Syntax:

```
define serviceextinfo {
       option                                  wert
       [...]
}
```

Folgende Pflichtoption bei Nutzung über die Konfigurationsobjekte müssen Sie dabei für Maschinen auf jeden Fall die folgende Option setzen:

- `host_name`

  Geben Sie hier die Maschine an, für die die folgenden Optionen gelten sollen.

Für Dienste sind bei Verwendung der Konfigurationsobjekte die folgenden Optionen obligatorisch:

- `host_name`

  Geben Sie hier die Maschine an, auf denen der gemeinte Dienst läuft.
- `service_description`

  Geben Sie hier den Dienst an, auf den sich die folgenden Optionen beziehen.

Alternativ können Sie die im Folgenden beschriebenen Optionen eben direkt in der Spezifikation von Maschinen beziehungsweise Diensten verwenden. Wie bereits erwähnt sollte dies die bevorzugte Variante sein, da die Konfigurationsobjekte wohl zukünftig nicht mehr unterstützt werden.

Notizen (notes)

Zunächst steht Ihnen über die Konfigurationsoption `notes` die Möglichkeit zur Verfügung, beliebigen Text hinzuzufügen, also beispielsweise so:

```
define hostextinfo {
       host_name                               Demonstrations-Maschine
       notes                                   Dies ist ein über die Direktive <em>notes \
                                               </em> angegebener Text. Dieser kann auch \
                                               etwas <strong>länger</strong> sein, wie \
                                               Sie hier sehen.
}
```

HTML-Tags: Sie können in diesen Zeichenketten auch HTML-Tags verwenden, wie hier zum Beispiel `<em>` und `<strong>` für eine entsprechende kursive beziehungsweise fette Darstellung.

Auf diese Weise hinterlegte Notizen werden Ihnen dann in den Details zu der jeweiligen Maschine beziehungsweise zu dem jeweiligen Dienst angezeigt. Über die Nutzung von

HTML-Auszeichnungen können Sie auch Links in diesen Notizen definieren und auf diese Weise andere, über Links erreichbare Systeme einbinden. Als weiteres Beispiel haben wir den gerade angeführten Text um einen Link erweitert:

```
define hostextinfo {
       host_name                              Demonstrations-Maschine
       notes                                  Dies ist ein über die <a href="docs. \
                                              icinga.org/latest/de/objectdefinitions. \
                                              html#objectdefinitions-hostextinfo"> \
                                              Direktive notes</a> angegebener Text. \
                                              Dieser kann auch etwas <strong>länger \
                                              </strong> sein, wie Sie hier sehen.
}
```

Abbildung 4-2 zeigt Ihnen, wie die obige Notiz in der klassischen Oberfläche von Icinga angezeigt wird. Beachten Sie bitte, dass diese Abbildung darüber hinaus die später in diesem Kapitel aufgeführten Modifikationen darstellt.

Verlinkungen (notes_url, action_url)

Da die Notizen, die wir Ihnen oben erläutert haben, nur in den jeweiligen Detail-Ansichten angezeigt werden, haben Sie auf diese nicht bei jeder Ansicht Zugriff. Gerade bei der angesprochenen Verlinkung auf andere Systeme ist es natürlich praktischer, wenn diese Links auch auf den Übersichtsseiten angezeigt werden. Hierzu bietet Nagios/Icinga Ihnen gleich zwei Optionen: `notes_url` und `action_url`. Mit diesen können Sie URLs einpflegen, die Nagios/Icinga Ihnen dann – über ein Icon – auch in den Übersichten anzeigen wird. Im folgenden Beispiel haben wir eine Maschine mit je einem URL für Ansicht und Aktion versehen:

```
define hostextinfo {
       host_name                              Demonstrations-Maschine
       notes_url                              http://internal.cms.org/?q=$HOSTNAME$
       action_url                             http://internal.cms.org/?q=cross-check:
                                              $HOSTNAME$
}
```

Sie können bei Icinga in den neueren Versionen hier auch mehrere URLs angeben, die zu diesem Zweck mit dem Zeichen ' gekapselt sein müssen. Wir haben die vorherige Definition im folgenden Beispiel entsprechend erweitert:

```
define hostextinfo {
       host_name                              Demonstrations-Maschine
       notes_url                              'http://internal.cms.org/?q=$HOSTNAME$' \
                                              'http://internal.cms.org/?q=show-history: \
                                              $HOSTNAME$'
       action_url                             'http://internal.cms.org/?q=cross-check: \
                                              $HOSTNAME$' 'http://internal.cms.org/ \
                                              ?q=escalate:$HOSTNAME$'
}
```

Bei der Verwendung von auf diese Weise gekapselten URLs müssen Sie beachten, dass für jeden zusätzlichen URL auch ein zusätzliches Icon benötigt wird. Die standardmäßig vorgesehenen Bilder *notes.gif* und *action.gif* werden nicht automatisch verwendet. Sie

können bei Angabe mehrerer URLs allerdings den letzten ohne Kapselung angeben – dieser wird dann die standardmäßigen Symbole verwenden. Für gekapselte URLs können Sie entweder eigene Grafiken hinterlegen oder aber Kopien der Standardsymbole anlegen. Bei der Installation aus Quellen sind diese Symbole unter */usr/local/icinga/share/images* abgelegt. In Tabelle 4-4 haben wir Ihnen die verschiedenen Installationspfade für die in diesem Buch besprochenen Installationsarten aufgeführt.

Tabelle 4-4: Verzeichnis mit den Icons bei unterschiedlichen Installationsarten

Installation	Icon-Verzeichnis
Installation Nagios aus Quellen	/usr/local/nagios/share/images/
Nagios Debian / Ubuntu	/usr/share/nagios3/htdocs/images
Nagios SLES|openSUSE	/usr/share/nagios/images/
Nagios RHEL|CentOS|Fedora	/usr/share/nagios/html/images/
Installation Icinga aus Quellen	/usr/local/icinga/share/images
Icinga Debian / Ubuntu	/usr/share/icinga/htdocs/images
Icinga SLES|openSUSE	/usr/share/icinga/images/
Icinga RHEL|CentOS|Fedora	/usr/share/icinga/images/

Wie diese Icons in der Detailansicht angezeigt werden, ist ebenfalls in Abbildung 4-2 am Beispiel der klassischen Icinga-Oberfläche zu sehen. Die vier Symbole am rechten Rand enthalten die Verlinkungen. In der Übersichts-Ansicht werden Sie die Symbole ebenfalls finden, allerdings ohne die Texte.

Ein Beispiel für die Nutzung dieser Option ist die Integration von PNP4Nagios in die Oberfläche von Icinga, wie wir sie in Rezept 1.4, *Icinga aus den Quellen installieren* beschrieben haben. Auf die gleiche Weise können Sie wie eben gezeigt auch eigene Dienste integrieren.

Icons (icon_image und icon_image_alt)

Bei allen Maschinen und Diensten verwendet Nagios/Icinga ein Symbol für die Anzeige. Bei der Voreinstellung ist dieses für alle Maschinen und Dienste das gleiche. Sie können dies aber anpassen, indem Sie die Optionen `icon_image` zur Spezifikation eines Symbols beziehungsweise `icon_image_alt` zur Angabe eines alternatives Textes verwenden. Die zu verwendenden Logos müssen sich dabei im Unterverzeichnis *logos* des oben genannten Verzeichnisses befinden, in unserem Fall also */usr/local/icinga/share/images/logos*. Das folgende Beispiel setzt für die referenzierte Maschine das Symbol *database.gif* und legt den entsprechenden Text fest:

```
define hostextinfo {
       host_name                       db
       icon_image                      database.gif
       icon_image_alt                  A different Icon
}
```

Beachten Sie, dass dieses Symbol zunächst nicht vorhanden ist und Sie es entsprechend erst anlegen müssen, um es verwenden zu können. Abbildung 4-2 zeigt Ihnen, wie die entsprechende Maschine in der klassischen Oberfläche angezeigt wird. Während Sie in der Übersicht nur das Symbol sehen werden, wird hier auch der alternative Text angezeigt.

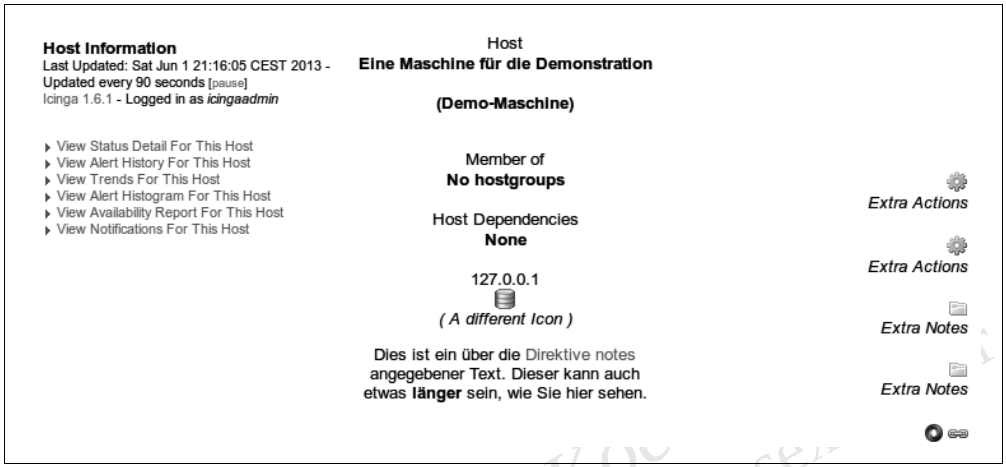


Abbildung 4-2: Erweiterte Informationen in der klassischen Oberfläche

Diskussion

Die hier dargestellten Erweiterungsmöglichkeiten beeinflussen die eigentlichen Überwachungsaufgaben nicht. Es sind lediglich Hilfsmittel für die Darstellung, die Sie nach Bedarf einsetzen können. Insbesondere die Einbindung von externen Systemen kann dabei eine große Hilfe darstellen. Die hier von uns verwendeten Definitionen über die eigens dafür vorgesehenen Konfigurationsobjekte sollten Sie allerdings nicht verwenden, sondern die Anweisungen stattdessen direkt in die Definition von Maschinen und Diensten integrieren.

Typischerweise werden Sie entsprechende Konfigurationen für eine Vielzahl von Maschinen eher über das Vorlagensystem vornehmen (siehe hierzu Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*). Wenn Sie etwa getrennte Vorlagen für Switches, Server und Drucker verwenden, ist es naheliegend, in diesen auch entsprechend Icons zu definieren. Ähnlich verhält es sich mit den vorherigen Beispielen, in denen wir bereits Makros zur Referenzierung der jeweiligen Maschine verwendet haben. Bei einer solch einzelnen Definition wäre es natürlich auch möglich gewesen, den jeweiligen Namen manuell anzugeben. Mit der gezeigten Referenzierung über Makros können Sie die entsprechende Anweisung aber auch in einer Vorlage unterbringen und sich so die Arbeit erleichtern.

Siehe auch

- Icinga-Dokumentation zum hostextinfo-Objekt: *http://docs.icinga.org/latest/de/objectdefinitions.html#hostextinfo*
- Icinga-Dokumentation zum serviceextinfo-Objekt: http://docs.icinga.org/latest/de/objectdefinitions.html#serviceextinfo
- Rezepte zu Maschinen und Dienstdefinitionen: Rezept 4.1, *Maschinen einbinden (host)* und Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Oberflächennutzung und temporären Kommentaren: Rezept 3.13, *Kommentare, Acknowledgments und Downtimes nutzen*
- Rezept zur Integration von PNP4Nagios in die Nagios/Icinga-GUI über die Option `action_url`: Rezept 1.4, *Icinga aus den Quellen installieren*
- Rezept zur Nutzung des Vorlagensystems: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*

KAPITEL 5

Lokale Parameter mit Standard-Plugins überwachen

In diesem Kapitel zeigen wir Ihnen, wie Sie lokale Plugins ausführen und einbinden können. Damit können Sie zunächst Parameter der Monitoring-Maschine selbst überwachen. Wir gehen dabei von den sogenannten Standard-Plugins aus. Dies ist eine Sammlung von Plugins, die typischerweise bei der Installation von Nagios/Icinga mitinstalliert wird.

Einige der Plugins sind Alternativen, um solche lokalen Tests auch auf entfernten Maschinen ausführen zu lassen. Wir werden Ihnen dabei folgenden Varianten solcher Aufrufe vorstellen:

- Einbindung per SSH (siehe Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten*) und Abfrage per `check_by_ssh` (Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*)
- Einbindung per SNMP (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* und Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*) und Abfrage per `check_snmp` (Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*)
- Einbindung per NSClient++ (siehe Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*) und Abfrage per `check_nt` (Rezept 5.9, *Überwachung einer Windows-Maschine mit NSClient (check_nt)*)
- Einbindung per NRPE (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*) und Abfrage per `check_nrpe` (Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*)

5.1 Anzahl angemeldeter Benutzer überwachen (check_users)

Problem

Sie möchten auf einer Maschine die Benutzeranzahl überwachen.

Lösung

Im Paket der Standard-Plugins ist zu diesem Zweck das Plugin `check_users` enthalten. Wir erläutern Ihnen im Folgenden, wie Sie dieses einbinden und verwenden können.

Befehlsdefinition und notwendige Optionen

Bei der Installation der Plugins wurde typischerweise bereits eine Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) eingerichtet:

```
define command{
       command_name                    check_local_users
       command_line                    $USER1$/check_users -w $ARG1$ -c $ARG2$
}
```

Dabei werden bereits die beiden Pflichtoptionen vorgesehen:

- `-w / --warning`

  Geben Sie mit dieser Option die Anzahl der Benutzer ein, ab der der Status WARNING zurückgegeben werden soll.

- `-c / --critical`

  Mit dieser Option spezifizieren Sie als zweiten Schwellenwert die Anzahl der Benutzer, ab der der Status CRITICAL zurückgegeben werden soll.

Die Werte werden über die Argumentmakros `$ARG1$` und `$ARG2$` übergeben (siehe dazu Rezept 4.3, *Befehle hinzufügen (command)*). Neben diesen beiden Pflichtoptionen sieht das Plugin keine weiteren Optionen vor.

Dienstdefinition

Neben der vorbereiteten Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) wird Ihnen bei der Installation standardmäßig auch eine Dienstdefinition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) zur Ausführung auf der lokalen Maschine eingerichtet. Bezugnehmend auf unser vorheriges Beispiel habe wir bei unserer Referenzinstallation (siehe Rezept 1.4, *Icinga aus den Quellen installieren*) die Schwellenwerte für `WARNING` und `CRITICAL` wir folgt auf 20 beziehungsweise 50 Benutzer gesetzt:

```
define service {
       use                             local-service
       host_name                       localhost
       service_description             Current Users
       check_command                   check_local_users!20!50
}
```

Lokale Plugins: Beachten Sie, dass Sie lokale Plugins wie dieses zunächst eben nur lokal, also auf der Monitoring-Maschine selbst ausführen können. Wir werden Ihnen im Rahmen der Diskussion später Hinweise darauf geben, wie Sie lokale Plugins auf entfernten Maschinen ausführen lassen können.

Test, Rückgabewerte und Graphen

Sie können das Plugin testen, indem Sie es wie folgt als Nagios- beziehungsweise Icinga-Nutzer von der Kommandozeile aus ausführen (passen Sie den Pfad gegebenenfalls an Ihre Umgebung an):

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch das Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit dem Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_users -w 10 -c 20
USERS OK - 3 users currently logged in |users=3;10;20;0
```

Da das Plugin durch ein Pipe-Symbol ("|") getrennt Performancedaten zurückgibt, erhalten Sie gegebenenfalls automatisch einen Graphen in PNP4Nagios. Abbildung 5-1 zeigt Ihnen den entsprechenden Graphen für den Zeitraum eines ganzen Jahres.

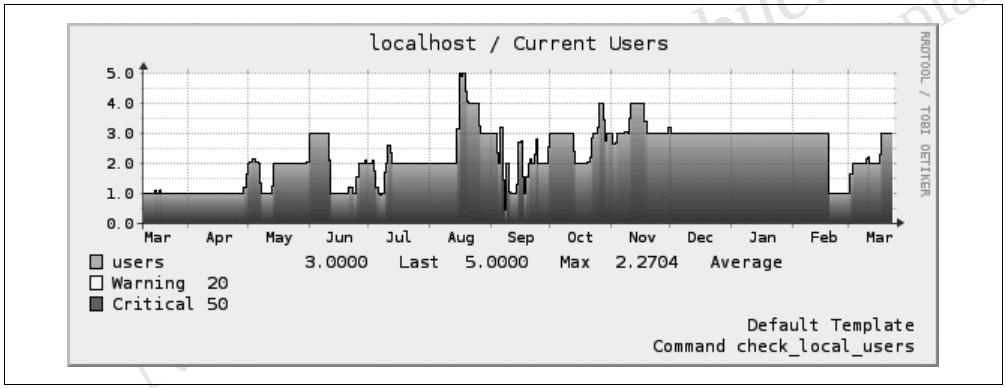


Abbildung 5-1: PNP4Nagios-Graph angemeldeter Benutzer über den Zeitraum eines Jahres

Diskussion

Die Anzahl der angemeldeten Benutzer ist ein wichtiger Indikator in Abhängigkeit von der Nutzung des jeweiligen Systems. Bei Maschinen, die eigentlich nicht von Benutzern bedient werden, stellt sie ein Sicherheitskriterium dar, dass Sie mit geringen Schwellwerten überwachen sollten. Bei Systemen, auf denen regelmäßig mehrere Benutzer angemeldet sind, ist der Wert hingegen für die Interpretation der Last hilfreich.

Um das Plugin auf entfernten Maschinen auszuführen, können Sie zwischen verschiedenen Varianten wählen. Wir empfehlen Ihnen hierzu die Variante über SNMP (siehe Rezept 2.5, *Plugins unter Linux über SNMP ausführen* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*). Alternativ können Sie das Plugin auch über NRPE (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*) oder SSH einbinden (siehe Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*).

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezepte zur Remote-Ausführung von lokalen Plugins über SNMP und Auswertung der entsprechenden Ergebnisse: Rezept 2.5, *Plugins unter Linux über SNMP ausführen* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*
- Rezept zur Remote-Ausführung von lokalen Plugins über SSH: Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*
- Rezept zur Remote-Ausführung von lokalen Plugins über NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

5.2 Prozessorauslastung überwachen (check_load)

Problem

Sie möchten bei einer Maschine die CPU-Auslastung überwachen.

Lösung

Zu diesem Zweck steht Ihnen das in den Standard-Plugins enthaltene `check_load` zur Verfügung. Wir zeigen Ihnen im Folgenden, wie Sie das Plugin einbinden und nutzen können.

CPU Load: Wann ein System ausgelastet ist, hängt neben der Anzahl laufender Prozesse von der Kapazität ab. Also der Anzahl der Prozesse, die parallel abgearbeitet werden können. Bei einem Prozessorsystem mit einem Kern ist (ohne Multithreading) der maximale Wert in der Regel 1. Wenn also hier der Load über einen längeren Zeitraum nahe an 1 ist, dann ist dieses System stark ausgelastet.

Bei einem Multicore-System hingegen wird der maximale Load in der Regel der Anzahl der vorhandenen Prozessorkerne entsprechen. Ein zwei Prozessorsystem mit Dual-Core-Prozessoren (in Summe also vier Prozessorkerne) wird demnach mit einer Load von 1 lediglich zu etwa 25% ausgelastet sein.

Als Faustformel können Sie den Wert `Load` zwischen verschiedenen System vergleichen, indem Sie diesen durch die Anzahl der vorhandenen Prozessor-Kerne teilen. Beachten Sie deshalb den Parameter `-r / --percpu`, falls Sie vergleichbare Werte angezeigt bekommen möchten.

Befehlsdefinition und notwendige Optionen

Bei der Installation wurde Ihnen für dieses Plugin bereits eine Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) erstellt. Bei unserer Referenzinstallation war dies die folgende Definition:

```
define command{
       command_name                    check_local_load
       command_line                    $USER1$/check_load -w $ARG1$ -c $ARG2$
}
```

Damit wird der Befehl unter Nutzung der Makros (siehe Rezept 4.3, *Befehle hinzufügen (command)*) `$USER1$` für den Pfad und `$ARG1$` sowie `$ARG2$` für die folgenden Pflicht-Parameter eingerichtet:

- `-w / --warning` und `-c / --critical`

  Mit diesen Optionen müssen Sie jeweils drei Schwellenwerte für die Status `WARNING` und `CRITICAL` angeben. Diese drei Werte geben dabei durch Kommata separiert jeweils die Schwellen für den Load über 1 Minute, 5 Minuten und 15 Minuten an.

Damit steht Ihnen eine minimalistische Definition für dieses Plugin zur Verfügung. Beachten Sie, dass das Plugin nur den Load der jeweiligen lokalen Maschine überprüfen kann. Die Angabe einer zu überprüfenden Maschine ist hier nicht vorgesehen.

Weitere Optionen

Für Anpassungen steht Ihnen genau eine weitere Option zur Verfügung:

- `-r / --percpu`

  Mit dieser Option schalten Sie die Ausgabe auf einen Wert pro Prozessorkern um. Der ohne diese Option gemessene Wert wird dann also durch die Anzahl der Prozessorkerne geteilt (siehe dazu den Tipp CPU Load).

Dienstdefinition

Die vorgegebene Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) sollte für Ihre lokale Maschine bereits als Dienst (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) definiert worden sein. Bei unserer Referenzinstallation geschah dies durch die folgende Dienstdefinition:

```
define service {
       use                             local-service
       host_name                       localhost
       service_description             Current Load
       check_command                   check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
}
```

Die dabei vorgesehenen Warnschwellen können Sie gegebenenfalls an Ihre Bedürfnisse anpassen.

Test, Rückgabewerte und Graphen

Die Ausführung des Plugins können und sollten Sie vor einer Einbindung auf der Kommandozeile testen. Um etwa die obigen Parameter zu überpüfen, können Sie als Nagios/Icinga Benutzer den folgenden Aufruf verwenden:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch das Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit dem Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_load -w5.0,4.0,3.0 -c10.0,6.0,4.0
OK - load average: 0.00, 0.03, 0.06|load1=0.000;5.000;10.000;0;
load5=0.030;4.000;6.000;0; load15=0.060;3.000;4.000;0;
```

Das Plugin gibt dabei jeweils durch das Pipe-Symbol "|" getrennt Performancedaten zur weiteren Auswertung zurück. Wenn Sie PNP4Nagios verwenden, erhalten Sie automatisch einen Graphen, wie beispielhaft in Abbildung 5-2 gezeigt.

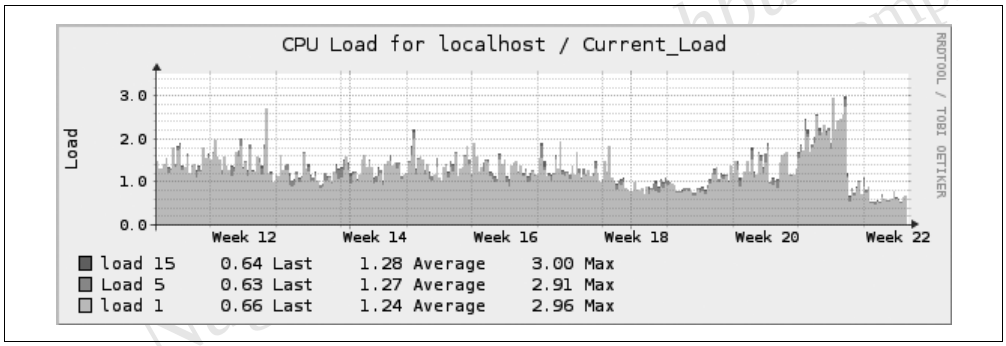


Abbildung 5-2: PNP4Nagios-Graph für check_load

Beachten Sie dabei, dass die Auflösung des Graphen vom genutzten Check-Intervall abhängig ist. Der dargestellte Graph basiert auf Checks im Minutentakt, bei einem längeren Intervall ist er gegebenenfalls entsprechend gröber aufgelöst.

Diskussion

Das Plugin wird automatisch eingerichtet und liefert so Daten zur Auslastung der CPU Ihrer Monitoring-Maschine. Da Sie diese Auslastung unbedingt beobachten sollten, ist dies sehr nützlich, aber eben zunächst auf die lokale Maschine begrenzt.

Typischerweise wollen Sie aber auch von anderen Maschinen die CPU-Auslastung erfassen. Sie haben hierzu verschiedene Möglichkeiten. Sofern Sie auf den zu überwachenden Maschinen SNMP nutzen, können Sie dieses auch für die Abfrage der CPU-Auslastung verwenden (siehe Rezept 2.5, *Plugins unter Linux über SNMP ausführen*). Alternativ kön-

nen Sie das hier besprochene lokale Plugin auch via NRPE (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*) oder SSH (siehe Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*) auf den zu überwachenden Maschinen ausführen.

Sofern Sie mit diesem Plugin unterschiedliche Maschinen überwachen, können Sie die Ausgaben gegebenenfalls zusätzlich mit der Option `-r` skalieren. Damit werden die Ausgaben und die Graphen der Tests leichter vergleichbar, geben dann allerdings nicht mehr direkt die auf der Maschine angezeigte Auslastung wieder.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros:: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Remote-Ausführung von lokalen Plugins über SNMP: Rezept 2.5, *Plugins unter Linux über SNMP ausführen*
- Rezept zur Remote-Ausführung von lokalen Plugins über NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*
- Rezept zur Remote-Ausführung von lokalen Plugins über SSH: Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*

5.3 Prozesse überwachen (check_procs)

Problem

Sie möchten die Anzahl aller oder bestimmter Prozesse überwachen.

Lösung

Das in den Standard-Plugins enthaltene `check_procs` ermöglicht Ihnen eine solche Prozessbeobachtung. Es zählt dazu die laufenden Prozesse, wobei Sie die zu berücksichtigenden Prozesse gegebenenfalls einschränken können. Wir zeigen Ihnen im Folgenden, wie Sie dieses Plugin einbinden und nutzen können.

Befehlsdefinition und notwendige Optionen

Bei der Installation der Plugins wurde typischerweise bereits eine Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) für Sie erstellt. Bei unserer Referenzinstallation war dies die folgende:

```
define command{
       command_name                    check_local_procs
       command_line                    $USER1$/check_procs -w $ARG1$ -c $ARG2$ \
                                       -s $ARG3$
}
```

Damit wurde das Plugin für die Benutzung mit mehreren optionalen Parametern definiert. Dieses Plugin funktioniert allerdings auch völlig ohne Optionen einwandfrei und zählt dann einfach alle laufenden Prozesse. Wenn Sie also keine Schwellenwerte für Warnungen verwenden und die berücksichtigten Prozesse nicht einschränken möchten, können Sie auch eine minimalistischere Definition wie die folgende verwenden:

```
define command{
       command_name                    check_local_procs
       command_line                    $USER1$/check_procs $ARG1$
}
```

Damit können Sie dann über die Dienstdefinition weitere Parameter inklusive der gewünschten Werte angeben, müssen dies aber nicht. Die vorgegebene Definition verpflichtet Sie hingegen zur Angabe der dort genannten Parameter.

Weitere Optionen

Wenn Sie nicht die Anzahl aller laufenden Prozesse überwachen wollen, können Sie die zu berücksichtigenden Prozesse zunächst einschränken. Dazu stehen Ihnen die folgenden Optionen zur Verfügung:

- `-s / --state`

  Mit dieser Option können Sie die Prozesse nach Status filtern. Welche Status Ihnen zur Verfügung stehen, hängt dabei von Ihrem System ab. Das Plugin verwendet die Status-Codes, die auch das Programm ps nutzt (siehe man-page des ps-Kommandos). Dabei können Sie mehrere Status zusammenfassen, indem Sie sie einfach aneinanderreihen, also etwa XZ für X und Z.

```
[...]
PROCESS STATE CODES
 Here are the different values that the s, stat and state output specifiers
 (header "STAT" or "S") will display to describe the state of a process.
 D Uninterruptible sleep (usually IO)
 R Running or runnable (on run queue)
 S Interruptible sleep (waiting for an event to complete)
 T Stopped, either by a job control signal or because it is being traced.
 W paging (not valid since the 2.6.xx kernel)
 X dead (should never be seen)
 Z Defunct ("zombie") process, terminated but not reaped by its parent.
[...]
```

Zombie-Prozesse: Das Testen auf die Statuscode XZ empfehlen wir Ihnen, auch bei Ihrem Monitoring-System selber auszubrobieren, um tote Child-Prozesse identifizieren zu können(siehe Rezept 8.1, *Überwachung von Unix-/Linux-Servern*).

- `-p / --ppid`

  Wenn Sie nur Kind-Prozesse eines bestimmten Prozesses berücksichtigen möchten, können Sie mit dieser Option die ID des Eltern-Prozesses angeben.

- `-z / --vsz`

  Mit dieser Option können Sie die Menge virtuellen Speichers angeben, die ein Prozess belegen muss, um bei der Zählung berücksichtigt zu werden.

- `-r / --rss`

  Wenn Sie anstatt wie bei der obigen Option den verwendeten physischen Speicher der Prozesse als Filterkriterium verwenden möchten, können Sie dazu diese Option nutzen. Geben Sie hierzu an, wieviel Speicher ein Prozess belegen muss, um berücksichtigt zu werden.

- `-P / --pcpu`

  Alternativ können Sie die Zählung auch über die CPU-Nutzung der Prozesse einschränken. Nutzen Sie hierzu diese Option und geben Sie an, wie viel CPU-Zeit in Prozent ein Prozess nutzen muss, um gezählt zu werden.

- `-u / --user`

  Um nur Prozesse eines bestimmten Benutzers zu zählen, können Sie mit dieser Option angeben, welchem Benutzer (ID oder Name) ein Prozess gehören muss, um berücksichtigt zu werden.

- `-C / --command`

  Wenn Sie nur Prozesse einer ganz bestimmten Anwendung beobachten möchten, können Sie mit dieser Option festlegen, welche das sein sollen. Geben Sie dazu den Namen des Programms (ohne Pfadangabe) an, wie er auch in der Ausgabe des Befehls `ps` angezeigt wird.

- `-a / --argument-array`

  Sie können Prozesse anstatt wie bei der vorhergehenden Option über den Namen alternativ auch über ihre Optionen spezifizieren. Geben Sie hierzu mit dieser Option eine Zeichenkette an, die in den Argumenten des Befehls enthalten sein muss.

- `--ereg-argument-array`

  Diese Option erlaubt wie die vorhergehende Option, die Argumente der zu berücksichtigenden Befehle anzugeben, allerdings als regulären Ausdruck.

Für die berücksichtigten Prozesse können Sie dann neben der Anzahl von Prozessen auch einen anderen Indikator wählen und Warnschwellen für die Status `WARNING` und `CRITICAL` angeben. Dazu stehen Ihnen die folgenden Optionen zur Verfügung:

- `-m / --metric`

  Mit dieser Option wählen Sie, welchen Indikator Sie für die betrachteten Prozesse auswerten möchten. Mögliche Werte sind diese:

  - `PROCS`: Die Anzahl der Prozesse – dies ist der Vorgabewert
  - `VSZ`: Die Menge des von den Prozessen belegten virtuellen Speichers
  - `RSS`: Die Menge des von den Prozessen belegten physischen Speichers

  - CPU: Die CPU-Nutzung der Prozesse in Prozent
  - ELAPSED: Die verwendete Rechenzeit der Prozesse in Sekunden
- -w / --warning und -c / --critical

  Mit diesen Optionen geben Sie die Warnschwellen an. Die Zahlenwerte werden dabei mit dem geprüften Wert (siehe Option -m) verglichen.

Neben der einfachen Möglichkeit, sämtliche laufenden Prozesse zu zählen, können Sie die zu betrachtenden Prozesse sowie den für diese auszuwertenden Indikator also auf vielfältige Weise modifizieren.

Dienstdefinition

Die bereits bei der Installation angelegte Befehlsdefinition wurde für die lokale Maschine vermutlich bereits als Dienst eingebunden (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*). Bei unserer Referenzinstallation war die dazugehörige Dienstdefinition die folgende:

```
define service{
       use                                local-service
       host_name                          localhost
       service_description                Total Processes
       check_command                      check_local_procs!250!400!RSZDT
}
```

Wie bereits erwähnt müssen Sie bei dieser Definition immer die zu berücksichtigenden Status sowie Warnwerte angeben. Um die alternativ vorgeschlagene, weiter vorne angeführte Definition einzubinden, können Sie die folgende Dienstdefinition verwenden:

```
define service{
       use                                local-service
       host_name                          localhost
       service_description                Total Processes
       check_command                      check_local_procs
}
```

Dabei würden ohne Warnschwellen alle Prozesse gezählt. Um mit dieser Variation die gleichen Optionen wie bei der Vorgabe-Definition zu verwenden, könnten Sie dabei die folgende Definition verwenden (siehe hierzu Rezept 4.3, *Befehle hinzufügen (command)*):

```
define service{
       use                                local-service
       host_name                          localhost
       service_description                Total Processes
       check_command                      check_local_procs!-w 250 -c 400 -s RSZDT
}
```

Dabei können Sie Optionen wie gewünscht setzen, also auch zusätzliche oder weniger.

Test, Rückgabewerte und Graphen

Um die Ausführung des Plugins zu testen, können Sie es von der Kommandozeile ausführen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch das Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit dem Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_procs
PROCS OK: 91 processes
```

Das Plugin gibt dabei keine Performance-Daten zurück, so dass Sie nicht ohne Weiteres Graphen zu dieser Metrik erstellen lassen können. Wir zeigen Ihnen im Rezept 4.3, *Befehle hinzufügen (command)*, wie Sie gegebenenfalls dennoch zu einem Graphen kommen und wie dieser aussieht.

Um die bei der Installation vorgeschlagenen (oder sonstige) Parameter zu testen, hängen Sie diese entsprechend einfach an:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_procs -w 250 -c 400 -s RSZDT
PROCS OK: 61 processes with STATE = RSZDT
```

Beachten Sie, wie sich die Anzahl der gezählten Prozesse sowie die Ausgabe des Plugins ändern.

Diskussion

Die Anzahl aller laufenden Prozesse ist durchaus eine interessante Kennzahl, aber in der Regel nicht unbedingt notwendig. Sehr viel wichtiger kann für Sie jedoch die gezielte Beobachtung ausgewählter Prozesse sein.

Wenn Sie die Anzahl aller Prozesse beobachten, wie weiter oben vorgeschlagen, haben Sie Alternativen bei der Auswahl von Warnungen. Zu diesen möchten wir Ihnen im Folgenden einige Möglichkeiten vorstellen:

- Option zum Auslösen eines Alarms, wenn die Gesamtzahl an Prozessen gegebene Grenzen überschreitet:

  ```
  -w 200 -c 400
  ```

- Optionen zum Auslösen eines Alarms, wenn einzelne Prozesse eine CPU-Last von mehr als 10 bzw. 20% verursachen:

  ```
  --metric=CPU -w 10 -c 20
  ```

- Optionen zum Auslösen eines Alarms, wenn einzelne Prozesse mehr als 50 beziehungsweise 100kB Speicher verbrauchen:

  ```
  --metric=VSZ -w 50000 -c 100000
  ```

- Optionen zum Auslösen eines Alarms beim Entstehen von Zombie-Prozessen:

  ```
  -w 1 -c 5 -s Z
  ```

Bei Nutzung der Option `metric` gibt das Plugin dann die Anzahl der die Schwellen überschreitenden Prozesse zurück. Als Beispiel hier eine Speicherprüfung mit absichtlich klein gewählten Werten:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_procs --metric=VSZ -w 10 -c 200000
VSZ CRITICAL: 12 crit, 40 warn out of 97 processes
```

Wenn Sie mehrere der Beispiele verwenden möchten, benötigen Sie jeweils eine Service-Definition mit einem eigenen Namen und entsprechenden Parametern.

Siehe auch

- Hilfeseiten des verwendeten Kommandos: `man ps`
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zum Einbinden einer Maschine: Rezept 4.1, *Maschinen einbinden (host)*
- Rezept für ein einfaches Wrapper-Script: Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*
- Rezept zur Überwachung eines Linux-Systems: Rezept 8.1, *Überwachung von Unix-/Linux-Servern*

5.4 Dateisysteme überwachen (check_disk)

Problem

Sie möchten den Füllstand lokaler Dateisysteme überwachen.

Lösung

Das in den Standard-Plugins enthaltene `check_disk` ist für die lokale Überwachung von Dateisystemen auf einer Maschine gedacht. Wir zeigen Ihnen im Folgenden wie Sie es einbinden und nutzen können.

Befehlsdefinition und notwendige Optionen

Das Plugin `check_disk` wurde bereits während der Installation eingerichtet. Die dabei verwendete Befehlsdefinition ist minimalistisch und sieht nur notwendige Parameter vor:

```
define command{
       command_name                    check_local_disk
       command_line                    $USER1$/check_disk -w $ARG1$ -c $ARG2$ \
                                       -p $ARG3$
}
```

Die über Makros vorgeplanten, notwendigen Parameter sind dabei folgende:

- `-w / --warning=WARNSCHWELLE` und `-c / --critical=KRITISCHE SCHWELLE`, alternativ `-W / --iwarning=%WERT` und `-K / --icritical=%WERT`

  Hier geben Sie die Schwellenwerte für die Status `WARNING` / `CRITICAL` an. Mit Hilfe von `-w` und `-c` prüfen Sie dabei auf den verbleibenden verfügbaren Speicherplatz (geben also an, wieviel Platz für einen Status `OK` noch frei sein muss). Mit `-W` und `-K` prüfen Sie hingegen auf den verbleibenden Platz von Inodes (Verwaltungseinträge für Dateien in manchen Unix-Dateisystemen, die die Anzahl der maximal möglichen Dateien begrenzen). Sie können dabei jeweils ein Prozentzeichen, also etwa `10%`, verwenden, um eine Angabe relativ zur Gesamtgröße zu treffen.

- `-p / --path=PFAD`, alternativ `--r / --ereg-path=PATH`, `-R / --eregi-path=PFAD` oder `-A / --all`

  Hier geben Sie die Einhängepunkte (mountpoints) der zu prüfenden Dateisysteme an. Wählen Sie dabei `-p`, um direkt den jeweiligen String anzugeben (mehrere `-p`-Parameter bei mehreren zu prüfenden Dateisystemen), oder `-r` (beachtet Groß-/Kleinschreibung) beziehungsweise `-R` (ignoriert Groß-/Kleinschreibung), um die Einhängepunkte mit einem regulären Ausdruck zu spezifizieren. Wenn Sie alle Dateisysteme überprüfen möchten, müssen Sie diese nicht aufzuzählen, sondern können direkt die Option `-A` nutzen.

Auf Befehlsdefinitionen und Makros sind wir in Rezept 4.3, *Befehle hinzufügen (command)* näher eingegangen.

Weitere Optionen

Sie können eine Vielzahl zusätzlicher Optionen nutzen, um das Plugin weiter an Ihre Bedürfnisse anzupassen. Hier eine Auswahl dieser weiteren Optionen:

- `-k / -m` oder `-u`

  Geben Sie Kilobytes (`-k`) beziehungsweise Megabytes (`-m`) als Maßeinheit vor oder wählen Sie mittels `-u` aus einer der Einheiten `bytes` (für Bytes) `kb` (für Kilobytes), `MB`, (für `GB` oder `TB` (Vorgabeeinheit ist MB).

- `-t`

  Geben Sie einen Timeout in Sekunden an, nach dem das Plugin aufhören soll, auf ein Ergebnis zu warten (Vorgabewert ist 10 Sekunden).

Sie können weiter Ausnahmen zu einer Auswahl definieren, Gruppen benennen oder das Plugin auf lokale Dateisysteme beschränken. Für eine vollständige Liste der möglichen Optionen rufen Sie das Plugin bitte mit der Option `--help` auf.

Dienstdefinition

Um die voranstehende Befehlsdefinition als Dienst (siehe dazu Rezept 4.3, *Befehle hinzufügen (command)* und Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) für die lokale Maschine einzubinden, können Sie folgende Definition verwenden:

```
define service{
       service_description                    Root Partition
       use                                    template-srv
       check_command                          check_local_disk!20%!10%!/
       host_name                              localhost
}
```

Dabei wird unter Nutzung der Vorlage `template-srv` der Befehl für die Maschine `localhost` für den Einhängepunkt `/` mit Warnschwellen bei 20% für `WARNING` und 10% für `CRITICAL` eingerichtet.

Test, Rückgabewerte und Graphen

Testen Sie die Funktion des Plugins mit den gewünschten Parametern von der Kommandozeile, bevor Sie es tatsächlich einbinden:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch das Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit dem Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_disk -W 90% -K 95% -p /
DISK CRITICAL - free space: / 3484 MB (63% inode=81%);| /=1971MB;;;0;5748
```

Die Rückgabe des Plugins enthält jeweils durch Pipes (|) getrennt entsprechende Performancedaten. Damit ist es unproblematisch, die Werte gegebenenfalls in einem Graphen darstellen zu lassen. Als Beispiel hier der PNP4Nagios-Graph eines entsprechenden Services.

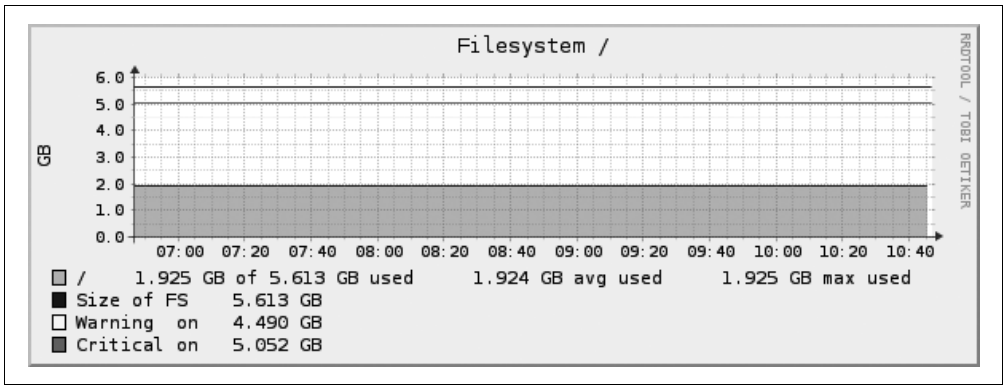


Abbildung 5-3: PNP4Nagios-Graph: Dateisystemcheck

Diskussion

Das Plugin `check_disk` ermöglicht Ihnen, schnell und unkompliziert auf der Monitoring-Maschine vorhandenen Dateisysteme zu überwachen. Wenn Sie weitere Server haben, deren Dateisysteme Sie mit diesem Plugin überwachen wollen, können Sie `check_disk` beispielsweise mithilfe des Plugins `check_by_ssh` (siehe dazu Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*) oder NRPE (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*) auf einem Zielserver aufrufen. Wir empfehlen Ihnen jedoch analog der Beschreibung in Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins* für diese Fälle die Einbindung über SNMP (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* und Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*).

Übergelaufene Dateisysteme können Ihnen schwerwiegende Probleme bescheren, wie den Ausfall ganzer Systeme. Deshalb sollten Sie unbedingt eine entsprechende Überwachung durchführen. Wenn Sie dabei mehrere Dateisysteme überwachen und Graphen zeichnen, ziehen Sie in Betracht, für jedes Dateisystem eine entsprechende Dienstdefinition zu erstellen. Dies ist besser als mehrere Dateisysteme mit nur einem Service zu überwachen, da Sie dann automatisch entsprechende Graphen für die einzelnen Dateisysteme erhalten.

Graphen der Dateisysteme ermöglichen Ihnen die zeitliche Entwicklung des Füllstandes nachzuvollziehen. Dies ist eine gute Grundlage für die Planung entsprechender Gegenmaßnahmen bei bestehenden Problemen oder vorausschauenden Ausbaumaßnahmen. Schauen Sie sich beispielsweise den Graphen in der Abbildung 5-4 an.

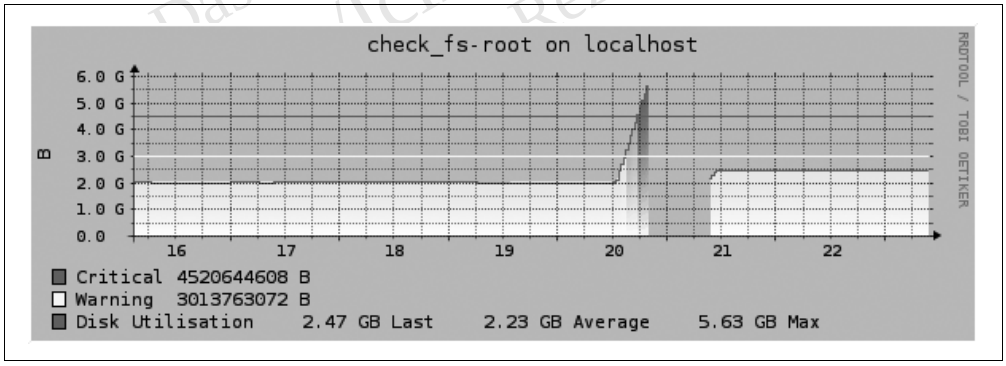


Abbildung 5-4: Beispiel: 7-Tage-Graph eines volllaufenden Dateisystems

Dieser Graph entstand durch ein Script, das auf dem Monitoring-System selbst systematisch die Platte vollgeschrieben hat, bis keine Daten mehr geschrieben werden konnten und dadurch die Aufzeichnung unterbrochen wurde. In diesem Falle ist aus dem Graphen direkt ersichtlich, wann das fehlerhafte Script gestartet wurde. Dadurch war der Fehler leicht zu identifizieren und zu beheben. Gleichzeitig ist zu erkennen, dass sich die ebenfalls in diesem Dateisystem befindenden und regelmäßig anfallenden Protokoll-

dateien auf absehbare Zeit kein Problem darstellen. Vergleichen Sie dazu den Graphen aus Abbildung 5-5 aus einem Backup-System.

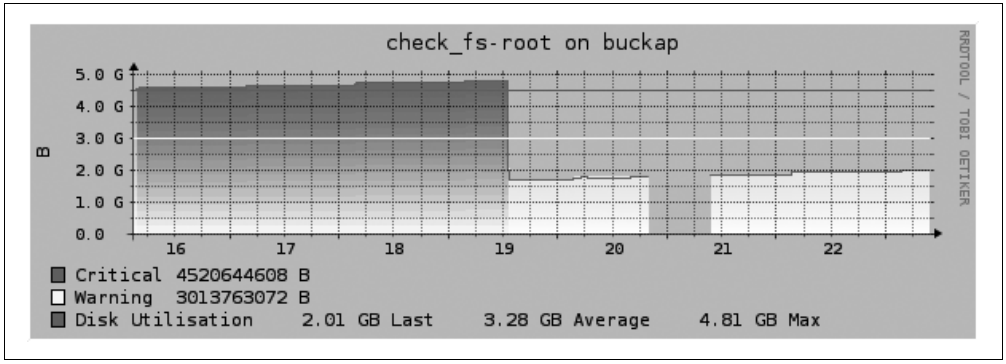


Abbildung 5-5: Beispiel: 7-Tage-Graph eines vollaufenden Backup-Systems

Sie können hier sofort sehen, dass das Backup-System `buckap` im Vergleich deutlich schneller vollläuft. Noch deutlicher wird dies, wenn Sie die Graphen dieser beiden Maschinen über einen längeren Zeitraum betrachten. Die Abbildung 5-6 stammt erneut von der zuerst angesprochenen Maschine.

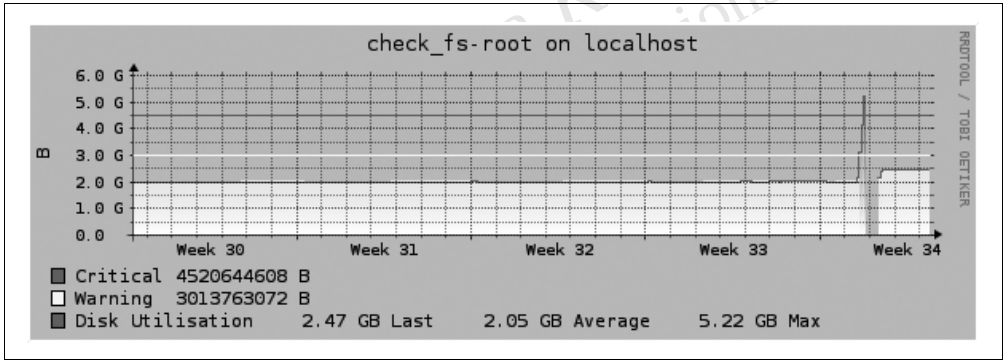


Abbildung 5-6: Beispiel: Monats-Graph des Script-Fehlers

Abbildung 5-7 zeigt den gleichen Zeitraum von dem Backup-System. Sie zeigen die gleichen Vorfälle, aber über den Zeitraum eines ganzen Monats. Die graphische Darstellung hebt bei dem in der ersten Maschine gezeigten Script-Fehler den kurzfristigen Charakter hervor. Die zweite hingegen zeigt, wie schnell das Backup-System vollläuft, und erlaubt eine auf Wochen genaue Abschätzung, wann das Dateisystem das nächste Mal voll sein wird.

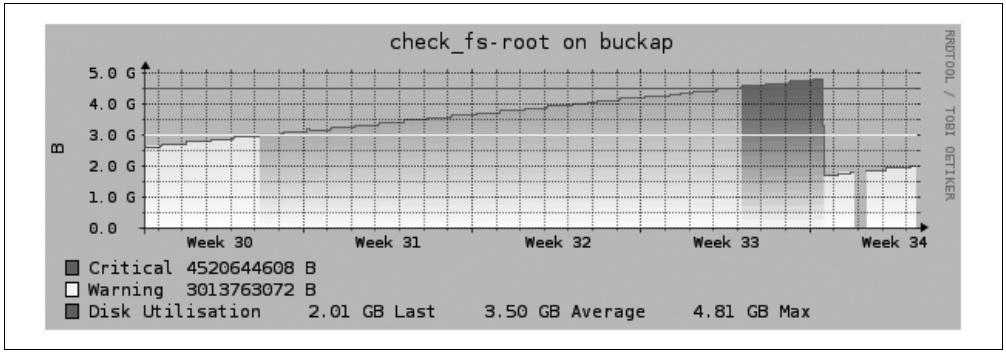


Abbildung 5-7: Beispiel: Monats-Graph des Backup-Systems

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Remote-Ausführung von lokalen Plugins über SSH: Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*
- Rezept zu Manubulons SNMP-Plugins: Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*

5.5 Festplatten-Status überwachen (check_ide_smart)

Problem

Sie möchten den Betriebszustand Ihrer Festplatten überwachen. Dazu wollen Sie die über SMART (Self-Monitoring, Analysis and Reporting Technology) bereitgestellten Werte in Ihr Monitoring-System einbeziehen.

Lösung

Für diesen Einsatz ist das in den Standard-Plugins enthalten `check_ide_smart` geeignet. Wir zeigen Ihnen im Folgenden, wie Sie dieses einbinden und verwenden können.

Befehlsdefinition und notwendige Optionen

Zur Einbindung benötigen Sie zunächst eine entsprechende Befehlsdefinition. Für eine Abfrage betreffender Statuswerte und Rückgabe einer entsprechenden Zusammenfassung im Nagios-Format können Sie die folgende Definition verwenden:

```
# check_ide_smart
define command{
       command_name                    check_ssh_ide_smart
       command_line                    $USER1$/check_ide_smart -n -d $ARG1$
}
```

Der Pfad zum Plugin und die variablen Parameter werden über die Makros `$USER1$` und `$ARG1$` gesetzt (siehe hierzu Rezept 4.3, *Befehle hinzufügen (command)*). Die zur Verfügung stehenden Optionen sind dabei folgende:

- `-n / --nagios`

  Mit dieser Option schalten Sie die Ausgabe auf das Nagios-Format um. Diese Option müssen Sie bei einer Einbindung in Nagios/Icinga angeben, da die Ausgabe vom Monitoring-System sonst nicht interpretiert werden kann.

- `-d / --device`

  Mit dieser Option geben Sie das zu prüfende Laufwerk an. In der voranstehenden Definition ist vorgesehen, dass dieser Wert im Rahmen der Service-Definition(en) gesetzt wird (siehe im Nachfolgenden).

Weitere Optionen

Die weiteren Optionen können Sie sich ansehen, indem Sie das Plugin mit der Option `--help` aufrufen. Diese erlauben es Ihnen, spezielle Tests zur genaueren Prüfung des Datenträgers anzustoßen. Sie sollten solche Tests allerdings nur mit Bedacht in automatisierter Form mittels Aufrufen aus einem Monitoring-System durchführen.

Dienstdefinition

Um die voranstehende Befehlsdefinition für das lokale Laufwerk `/dev/sda` einzubinden, können Sie die folgende Dienstdefinition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) verwenden:

```
define service{
       service_description             SMART /dev/sda
       use                             template-srv-hcksp
       check_command                   check_ide_smart!/dev/sda
}
```

Wir verwenden hier eine Vorlage `template-srv-hcksp`, die entsprechend vorhanden sein muss (siehe Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*). Passen Sie den Namen der Vorlage gegebenenfalls an Ihre Umgebung an. Beachten Sie des Weiteren, dass das gewünschte Laufwerk `/dev/sda` in der Zeile `check_command` als Parameter angegeben wird. Fügen Sie gegebenenfalls weitere Dienstdefinitionen (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) mit entsprechend angepassten Parametern für alle Ihre SMART-fähigen Laufwerke hinzu.

Test, Rückgabewerte und Graphen

Bevor Sie die Einbindung vornehmen, können und sollten Sie das Plugin auf der Kommandozeile testen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch das Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit dem Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ide_smart -d /dev/sda -n
OK - Operational (25/25 tests passed)
```

Beachten Sie, dass das Plugin keine Performancedaten zurückgibt. Entsprechend sind zu diesem Plugin keine Graphen vorhanden, die gezeichnet werden könnten.

Diskussion

Für die grundlegende Überwachung ist das Plugin bestens geeignet und es lässt sich wie gezeigt leicht einbinden. Falls dieses Plugin bei Ihnen einmal nicht den Status `OK` zurückgibt, sollten Sie auf jeden Fall eine gründlichere Prüfung des entsprechenden Laufwerks vornehmen. Das Plugin selbst können Sie hierzu von der Kommandozeile ohne den Parameter `-n` aufrufen, dann zeigt es Ihnen eine ausführliche Ansicht der ausgelesenen Parameter, wie im folgenden Beispiel:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ide_smart -d /dev/sda
Id= 1, Status=15 {PreFailure , OnLine }, Value=118, Threshold= 6, Passed
Id= 3, Status= 3 {PreFailure , OnLine }, Value= 94, Threshold= 0, Passed
Id= 4, Status=50 {Advisory , OnLine }, Value=100, Threshold= 20, Passed
Id= 5, Status=51 {PreFailure , OnLine }, Value=100, Threshold= 36, Passed
Id= 7, Status=15 {PreFailure , OnLine }, Value= 81, Threshold= 30, Passed
Id= 9, Status=50 {Advisory , OnLine }, Value= 95, Threshold= 0, Passed
Id= 10, Status=19 {PreFailure , OnLine }, Value=100, Threshold= 97, Passed
[...]
OffLineStatus=130 {Completed}, AutoOffLine=Yes, OffLineTimeout=10 minutes
OffLineCapability=123 {Immediate Auto SuspendOnCmd}
SmartRevision=10, CheckSum=41, SmartCapability=3 {SaveOnStandBy AutoSave}
```

Wenn hier Tests als fehlgeschlagen oder mit Warnungen ausgegeben werden, können Sie anhand der ID (zu IDs und spezifischen Anwendungen für unterschiedliche Betriebssysteme siehe *https://de.wikipedia.org/wiki/S.M.A.R.T.*) überprüfen, um welche Werte es sich handelt, und entsprechende Maßnahmen ergreifen.

Wenn Sie viele Laufwerke möglicherweise auf unterschiedlichen Maschinen überwachen, bietet es sich eventuell an, diese in entsprechenden Servicegruppen zu ordnen (siehe hierzu Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*). Zur Ausführung des Plugins auf anderen Maschinen als auf der lokalen können Sie zwischen ver-

schiedenen Varianten wählen (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*, Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)* und Rezept 2.5, *Plugins unter Linux über SNMP ausführen*).

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Wikipedia-Seite zu SMART: *https://de.wikipedia.org/wiki/S.M.A.R.T.*
- Rezept zu Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Rezept zu Dienstgruppen: Rezept 4.7, *Services zu Gruppen zusammenfassen (servicegroup)*
- Rezept zur Remote-Ausführung von lokalen Plugins über NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*
- Rezept zur Remote-Ausführung von lokalen Plugins über SSH: Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*
- Rezept zur Remote-Ausführung von lokalen Plugins über SNMP: Rezept 2.5, *Plugins unter Linux über SNMP ausführen*

5.6 Sensor-Daten überwachen (check_sensors)

Problem

Sie wollen von einer Linux-Maschine die Umweltsensoren überwachen.

Lösung

Hauptplatinen von modernen Servern, PCs und häufig auch von eingebetteten Systemen sind mit Hardware-Sensoren ausgestattet. Häufig lassen sich zumindest Werte wie anliegende Stromspannungen, gemessene Temperaturen und die Rotationsgeschwindigkeiten der Lüfter auslesen. Daraus können Sie Rückschlüsse über den Gesundheitszustand eines Systems ziehen.

Das Programm-Paket `lm_sensors` liefert bei Linux-Maschinen einen Abfrage-Mechanismus für diese Werte. Mit dem in den Standard-Proben enthaltenen `check_sensors` können Sie alle in diesem Paket (siehe die Webseite vom des lm-sensors-Projektes unter *http://www.lm-sensors.org/*) bekannten Sensoren abfragen. Wir zeigen Ihnen im Folgenden, wie Sie das Plugin testen und einbinden können.

lm-sensors installieren: Das Tool `lm-sensors` ist in den meistens Distributionen enthalten. Unter den von uns genutzten Distributionen können Sie das Tool wie folgt installieren:

Debian/Ubuntu:

```
Debian/Ubuntu
root@moni-wheezy-icinga:~# aptitude install lm-sensors
```

SLES/OpenSuSE:

```
moni-suse122-icinga:~ # zypper install sensors
```

RHEL/CentOS:

```
[root@moni-centos63-icinga ~]# yum install lm_sensors
```

Befehlsdefinition und notwendige Optionen

Zur grundlegenden Einbindung des Plugins für die lokale Abfrage auf der Monitoring-Maschine können Sie die folgende Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) verwenden:

```
define command{
       command_name                    check_local_sensors
       command_line                    $USER1$/check_sensors
}
```

Dabei müssen Sie keine weiteren Optionen angeben. Das Plugin kennt als Optionen lediglich die folgenden:

- `-v`

  Hiermit schalten auf eine ausführliche Ausgabe um (Beispiel siehe im Nachfolgenden).
- `--ignore-fault`

  Mit dieser Option können Sie gemeldete Fehler ignorieren.

Dienstdefinition

Um aufbauend auf der obigen Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) einen Service für die lokale Maschine zu definieren (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*), können Sie dann die folgende Definition verwenden:

```
define service{
       service_description             local-sensors
       use                             template-service
       check_command                   check_local_sensors
       host_name                       localhost
}
```

Um das Plugin auf Remote-Maschinen einsetzen zu können, müssen Sie allerdings weitere Schritte unternehmen. Möglichkeiten sind hier beispielsweise das Einbinden per SSH, SNMP und NRPE (siehe dazu den Diskussionsteil).

Test, Rückgabewerte und Graphen

Ob das Plugin auch alle gewünschten Werte erfasst, können Sie mit der Option -v für die ausführliche Ausgabe testen. Dabei erhalten Sie aneinandergereiht alle verfügbaren Sensordaten, etwa folgendermaßen:

```
icinga@quozad:~# /usr/lib/nagios/plugins/check_sensors -v
k10temp-pci-00c3 Adapter: PCI adapter temp1: +29.1°C (high = +70.0°C) (crit = +67.0°C,
hyst = +62.0°C) it8720-isa-0228 Adapter: ISA adapter in0: +1.20 V (min = +0.00 V, max =
+4.08 V) in1: +1.49 V (min = +0.00 V, max = +4.08 V) in2: +3.38 V (min = +0.00 V, max =
+4.08 V) +5V: +3.04 V (min = +0.00 V, max = +4.08 V) in4: +2.98 V (min = +0.00 V, max =
+4.08 V) in5: +1.62 V (min = +0.00 V, max = +4.08 V) in6: +3.38 V (min = +0.00 V, max =
+4.08 V) 5VSB: +2.99 V (min = +0.00 V, max = +4.08 V) Vbat: +3.12 V fan1: 2220 RPM (min
= 10 RPM) fan2: 0 RPM (min = 0 RPM) fan3: 0 RPM (min = 0 RPM) fan5: 0 RPM (min = 0 RPM)
temp1: +35.0°C (low = +127.0°C, high = +127.0°C) sensor = thermistor temp2: +40.0°C (low
= +127.0°C, high = +127.0°C) sensor = thermal diode temp3: +81.0°C (low = +127.0°C, high
= +127.0°C) sensor = thermistor cpu0_vid: +1.050 V intrusion0: OK
sensor ok
```

Die Werte entsprechen dabei denen der Ausgabe des im Paket enthaltenen Befehls sensors. Im hier beispielhaft betrachteten System mit der obigen Ausgabe sähe das etwa wie folgt aus:

```
icinga@quozad:~# sensors
it8720-isa-0228
Adapter: ISA adapter
in0: +1.20 V (min = +0.00 V, max = +4.08 V)
in1: +1.49 V (min = +0.00 V, max = +4.08 V)
in2: +3.38 V (min = +0.00 V, max = +4.08 V)
+5V: +3.06 V (min = +0.00 V, max = +4.08 V)
in4: +2.98 V (min = +0.00 V, max = +4.08 V)
in5: +1.50 V (min = +0.00 V, max = +4.08 V)
in6: +3.38 V (min = +0.00 V, max = +4.08 V)
5VSB: +2.99 V (min = +0.00 V, max = +4.08 V)
Vbat: +3.09 V
fan1: 1713 RPM (min = 10 RPM)
fan2: 0 RPM (min = 0 RPM)
fan3: 0 RPM (min = 0 RPM)
fan5: 0 RPM (min = 0 RPM)
temp1: +30.0°C (low = +127.0°C, high = +127.0°C) sensor = thermistor
temp2: +34.0°C (low = +127.0°C, high = +127.0°C) sensor = thermal diode
temp3: +81.0°C (low = +127.0°C, high = +127.0°C) sensor = thermistor
cpu0_vid: +1.050 V
intrusion0: OK

k10temp-pci-00c3
Adapter: PCI adapter
temp1: +22.0°C (high = +70.0°C)
 (crit = +67.0°C, hyst = +62.0°C)
```

Die eigentliche Funktion des Plugins, so wie sie von Nagios/Icinga ausgeführt wird, testen Sie ohne die Option -v (genauer gesagt ohne jegliche Option).

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch das Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit dem Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@quozad:~$ /usr/lib/nagios/plugins/check_sensors
sensor ok
```

Nun haben Sie auf einfache Weise sämtliche von `lm_sensors` erfasste Sensoren des Systems eingebunden und erhalten eine Warnung, falls die vor-eingestellten Warnschwellen über- beziehungsweise unterschritten werden. Eigene Warnschwellen müssten Sie in diesem Falle allerdings über die Konfiguration des zugrundeliegenden Pakets vornehmen, auf die wir hier allerdings nicht eingehen. Konsultieren Sie dazu bitte die Dokumentation zum Paket `lm-sensors`.

Diskussion

Die beschriebene Einbindung ermöglicht Ihnen sehr unkompliziert eine einfache Abfrage der Sensordaten der lokalen Maschine. Sie können das Plugin darüber hinaus auch auf weiteren Maschinen einsetzen, die von Ihrem Monitoring-System aus über das Netzwerk erreichbar sind. Sie können das Plugin über SSH (siehe Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten*) und die Abfrage per `check_by_ssh` (Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*) beziehungsweise per SNMP (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*) und die Abfrage per `check_snmp` (Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*) aufrufen. Alternativ können Sie den Aufruf auch per NRPE (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*) und `check_nrpe` (Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*) durchführen.

Mit Hilfe eines Wrapper-Scripts (siehe hierzu Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*) können Sie sich ausgewählte Daten aus dieser Sammlung auch graphisch darstellen lassen. Wir verwenden beispielsweise die angesprochene Einbindung über SNMP (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*), wobei wir `check_sensors` über SNMP als `check_sensors_full` mit der Option `-v` aufrufen (damit die Daten überhaupt angezeigt werden). Das für diese Einbindung beschrieben Wrapper-Script aus Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen* haben wir zu diesem Zweck leicht modifiziert. Die letzten vier Zeilen im Rezept sind folgende:

```
[...]
RESULT_NUM=`echo $REMOTE_RC1 | cut -d" " -f4`
RESULT_TEXT=`echo $REMOTE_RC2 | cut -d"-" -f2-`
echo $RESULT_TEXT
exit $RESULT_NUM
```

Um aus dem voranstehenden Beispiel die Daten für die Lüfterdrehzahl zu extrahieren und als Performance-Daten zurückzugeben, haben wir diese Zeilen in einer Kopie des Scripts wie folgt modifiziert:

```
[...]
RESULT_NUM=`echo $REMOTE_RC1 | cut -d" " -f4`
RESULT_TEXT=`echo $REMOTE_RC2 | cut -d"-" -f2-`
FAN1=`echo $RESULT_TEXT | egrep -o 'fan1: [0-9]+ RPM'`
FAN1_RPM=`echo $FAN1 | awk '{print $2}'`
echo $FAN1 \| fan=$FAN1_RPM
exit $RESULT_NUM
```

Wir verwenden hier die Befehle `echo`, `egrep` und `awk`, um die Ausgabe weiter zu verarbeiten (siehe Hilfeseiten zu den Befehlen). Der Aufruf von der Kommandozeile sieht danach folgendermaßen aus:

```
icinga@yas-moni~# /usr/local/icinga/libexec2/wrapper_check_snmp_fan.sh 192.168.42.241
check_sensors_full
fan1: 2922 RPM | fan=2922
```

Den zurückgegebenen Performancedaten entsprechend erzeugt PNP4Nagios dann auch einen Graphen dazu:

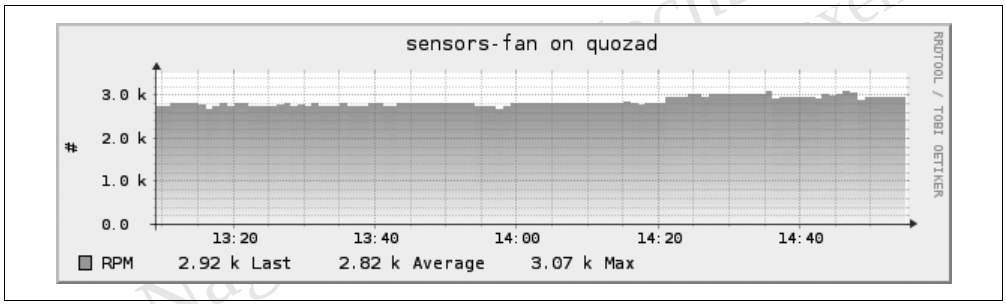


Abbildung 5-8: PNP4Nagios-Graph zur Lüfterdrehzahl

Analog können Sie auch mit den anderen zur Verfügung stehenden Daten verfahren. Im folgenden Beispiel wurde das Script so modifiziert, dass alle vier vorhandenen Temperaturen ausgelesen werden:

```
RESULT_NUM=`echo $REMOTE_RC1 | cut -d" " -f4`
RESULT_TEXT=`echo $REMOTE_RC2 | cut -d"-" -f2-`
TEMP1_TEXT=`echo $RESULT_TEXT | egrep -o '\) temp1: [+-]+[0-9]+\.[0-9]+' |
egrep -o 'temp1: [+-]+[0-9]+\.[0-9]+'`
TEMP1=`echo $TEMP1_TEXT | egrep -o '[0-9]+\.[0-9]+'`
TEMP2_TEXT=`echo $RESULT_TEXT | egrep -o 'temp2: [+-]+[0-9]+\.[0-9]+'`
TEMP2=`echo $TEMP2_TEXT | egrep -o '[0-9]+\.[0-9]+'`
TEMP3_TEXT=`echo $RESULT_TEXT | egrep -o 'adapter temp1: [+-]+[0-9]+\.[0-9]+'`
TEMP3=`echo $TEMP3_TEXT | egrep -o '[0-9]+\.[0-9]+'`
echo $TEMP1_TEXT, $TEMP2_TEXT, $TEMP3_TEXT \| temp1=$TEMP1 temp2=$TEMP2 temp3=$TEMP3
exit $RESULT_NUM
```

Die Ausgabe dieses Wrapper-Skipts auf der Kommandozeile sieht dann folgendermaßen aus:

```
root@yas-moni:~# /usr/local/icinga/libexec2/wrapper_check_snmp_temp.sh 192.168.42.241
check_sensors_full
temp1: +37.0, temp2: +50.0, adapter temp1: +37.8 | temp1=37.0 temp2=50.0 temp3=37.8
```

Und der dazugehörige Graph in PNP4Nagios stellt sich dann wie folgt dar:

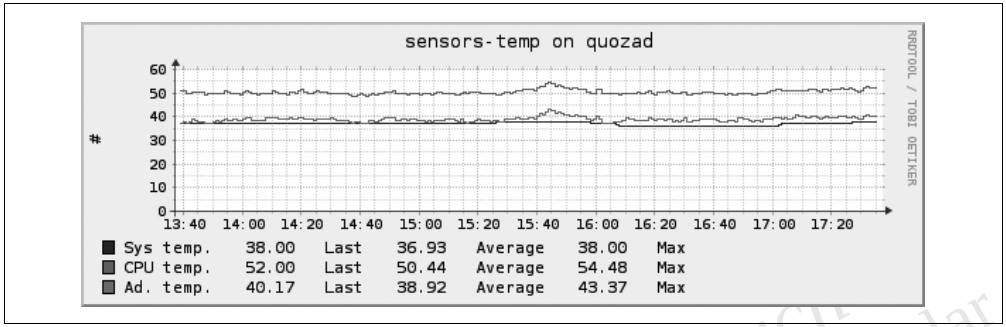


Abbildung 5-9: PNP4Nagios-Graph zu den Temperaturen

Um diese Daten graphisch auszuwerten, ist also durchaus zusätzlicher Aufwand erforderlich. Dafür können Sie auf diese Weise Sensordaten nach Bedarf auswerten und graphen.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Hilfeseiten von lm-sensors: `man sensors`
- Webseite vom des ln-sensors-Projektes: *http://www.lm-sensors.org/*
- Rezepte zur Remote-Ausführung von lokalen Plugins über SSH: Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten* und Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*
- Rezepte zur Remote-Ausführung von lokalen Plugins über SNMP: Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*, Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten* und Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*)
- Rezepte zur Remote-Ausführung von lokalen Plugins über NRPE: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*, Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*
- Rezepte zur Erstellung eines Wrapper-Scripts: Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*
- Hilfeseiten der verwendeten Befehle: `man echo`, `man egrep` und `man awk`

5.7 Log-Dateien überwachen (check_log)

Problem

Sie möchten ein bestimmtes Ereignis in einer Protokoll-Datei überwachen.

Lösung

In den Standard-Plugins ist mit `check_log` ein Plugin zu eben diesem Zweck enthalten. Wir zeigen Ihnen im Folgenden, wie Sie das Plugin einbinden und verwenden können.

`check_logfiles`: Das Plugin `check_log` ist ein sehr schlichtes Plugin, das nur rudimentäre Funktionen zu Auswertung von Logfiles besitzt. Wesentlich besser für die Durchsuchung von Logfiles nach einem definierten Muster ist das Plugin `check_logfiles` (siehe *http://labs.consol.de/lang/de/nagios/check_logfiles/*) geeignet. So werden beispielsweise rotierende Logdateien unterstützt und es können sowohl mehrere Muster als auch mehrere zu durchsuchende Logdateien angegeben werden. Es können auch Ausnahmen und Schwellenwerte definiert werden. Außerdem kann dieser Check auch Performancedaten ausgeben, die durch einen Grapher gesammelt und dargestellt werden können.

Befehlsdefinition und notwendige Optionen

Um das Plugin in Nagios/Icinga verwenden zu können, müssen Sie es zunächst als Befehl einbinden (siehe hierzu Rezept 4.3, *Befehle hinzufügen (command)*).

```
define command {
       command_name                    check_local_log
       command_line                    $USER1$/check_log -F $ARG1$ -O $ARG2$ \
                                       -q $ARG3$
}
```

In dieser Definition sind alle notwendigen Optionen zur Ersetzung über Makros vorgesehen. Diese Optionen sind folgende:

- `-F`

  Mit dieser Option spezifizieren Sie die zu untersuchende Protokolldatei.

- `-O`

  Damit das Plugin jeweils nur die neuen Einträge untersucht, geben Sie mit dieser Option eine temporäre Datei an, in der das Plugin die bereits untersuchten Einträge ablegt. Auf diese Weise kann es den Unterschied zur aktuellen Protokolldatei bestimmen.

- `-q`

  Mit dieser Option legen Sie fest, wonach das Plugin suchen soll. Geben Sie hierzu einen regulären Ausdruck an, der bei den gesuchten Zeilen zutrifft.

Neben diesen Optionen, die Sie alle angeben müssen, bietet das Plugin keine weiteren Optionen.

Dienstdefinition

Zur Einbindung der voranstehenden Kommandodefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) können Sie dann eine Dienstdefinition wie die Folgende verwenden (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*):

```
define service {
       service_description                    check_icingalog-plugin-timeouts
       use                                    template-service-generic
       check_command                          check_local_log!/usr/local/icinga/var/ \
                                              icinga.log!/tmp/oldlog.log!"Plugin timed \
                                              out"
       host_name                              localhost
}
```

Damit wird die Protokoll-Datei von Nagios/Icinga auf der lokalen Maschine untersucht und eine temporäre Datei unter */tmp/oldlog.log* für die Speicherung bereits untersuchter Einträge verwendet. Wenn mindestens ein Eintrag mit der Zeichenkette `Plugin timed out` gefunden wird, wird das Plugin in den Fehlerstatus `CRITICAL` wechseln und Ihnen die Anzahl der Treffer sowie die letzte Trefferzeile zurückgeben.

Zugriffsrechte für Protokolldateien: Sie werden als nicht priviligierter Benutzer häufig nicht auf die systemweiten Protokolldateien zugreifen dürfen. Sie können hier erwägen, die in Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten* beschriebenen Schritte zur Ausstattung des Plugins mit erweiterten Rechten zu wählen. Alternativ können Sie natürlich auch die Rechte für die Protokolldateien so setzen, dass der von Nagios/Icinga verwendete Benutzer Zugriff hat.

Test, Rückgabewerte und Graphen

Die temporäre Datei wird beim ersten Aufruf des Plugins erzeugt. Sie sollten deshalb unbedingt einen mehrmaligen Test von der Kommandozeile aus durchführen. Wir gehen bei den folgenden Beispielen von den voranstehenden in den Definitionen genutzten Beispielen aus. Der erste Aufruf ergibt dann Folgendes:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch das Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit dem Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~# /usr/local/icinga/libexec/check_log -F /usr/local/icinga/var/icinga.log
-O /tmp/oldlog.log -q "Plugin timed out after "
Log check data initialized...
```

Wie Sie der Ausgabe entnehmen können, ist das Plugin, genauer die temporäre Datei, damit initialisiert. Beim nächsten Aufruf wird also erstmals die eigentliche Logik ausgeführt, wie zum Beispiel folgendermaßen:

```
icinga@moni:~# /usr/local/icinga/libexec/check_log -F /usr/local/icinga/var/icinga.log
-O /tmp/oldlog.log -q "Plugin timed out after "
Log check ok - 0 pattern matches found
```

In einem Fehlerfall ändert sich die textuelle Rückgabe, wie etwa in dem folgenden Beispiel:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_log -F /usr/local/icinga/var/icinga.log
-O /tmp/oldlog.log -q "Plugin timed out after "
(1) < [1364017445] SERVICE ALERT: docu;check_ssh_apt;CRITICAL;SOFT;1;CRITICAL
- Plugin timed out after 10 seconds
```

Da das Plugin keine Performancedaten zurückgibt, kann zunächst auch kein Graph gezeichnet werden.

Diskussion

Beim Plugin `check_log` handelt es sich um ein schlichtes Plugin. Sie können die Rückgabewerte nicht steuern und erhalten bei Treffern immer den Status `CRITICAL`. Dennoch ist das Plugin gleichzeitig vergleichsweise mächtig, da Sie über den regulären Ausdruck für jede Log-Datei quasi beliebige Treffermengen definieren können. Es ist damit geeignet, Sie hinsichtlich aller Ereignisse zu warnen, die sich über einen regulären Ausdruck beschreiben und aus der zu überwachenden Protokolldatei auslesen lassen.

Die Einschränkung, dass das Plugin zunächst nur Dateien auf dem lokalen Server verarbeiten kann, können Sie auf unterschiedliche Weisen umgehen. Die von uns favorisierte Lösung hierzu ist die Nutzung eines SNMP-Servers auf den Remote-Maschinen und einen Einbindung wie im Rezept 2.5, *Plugins unter Linux über SNMP ausführen* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen* beschrieben. Alternativ können Sie das Plugin auch wie in Rezept über SSH Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten* oder wie in Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* beschrieben über NRPE einbinden.

Siehe auch

- Projekt-Seite der check_logfiles-Plugins: *http://labs.consol.de/lang/de/nagios/check_logfiles/*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Ausstattung von Plugins mit administrativen Rechten: Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten*

- Rezepte zur Remote-Ausführung von lokalen Plugins über SNMP und Auswertung der entsprechenden Ergebnisse: Rezept 2.5, *Plugins unter Linux über SNMP ausführen* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*
- Rezepte zur Remote-Durchführung lokaler Plugins über SSH und NRPE: Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten* beziehungsweise Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*

5.8 Überwachung einer Maschine mit NRPE (check_nrpe)

Problem

Sie möchten lokale Plugins auf einer entfernten Maschine ausführen. Die Einbindung in Ihr Monitoring-System wollen Sie mit Hilfe von NRPE vornehmen.

Lösung

NRPE steht für `Nagios Remote Plugin Executor` und das entsprechende Plugin `check_nrpe` ist bei der Installation der Nagiosplugins mit enthalten. Über NRPE können Sie lokale Plugins auf einer entfernten Maschine ausführen. Clientseitig benötigen Sie dazu einen installierten Agenten, der die Anfragen des Nagios/Icinga Monitoring Systems beantwortet.

Sofern keine fertigen Pakete vorhanden sind, können Sie auf den meisten Unix-basierenden Maschinen (inclusive MacOS, BSD und Solaris) NRPE aus dem Quellcode übersetzen. Die Quellen können Sie über den Link *http://sourceforge.net/projects/nagios/files/nrpe-2.x/* beziehen. Für die Installation und Konfiguration können Sie sich an Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* orientieren. Bei der Installation des NRPE-Clients bietet es sich an, die Nagiosplugins gleich mitzuinstallieren. Ihnen stehen dann sofort eine Vielzahl von Checks zur Verfügung, mit denen Sie unter anderem auch lokale Parameter der Zielmaschine auslesen können

Mit Hilfe von Cygwin können Sie den NRPE-Agenten zwar auch auf Windows Systemen einsetzten, hier empfehlen wir Ihnen allerdings den Einsatz des Pakets NSClient++. Dieses unterstützt nicht nur NRPE, sondern unter anderem auch NSClient (siehe dazu Rezept 5.9, *Überwachung einer Windows-Maschine mit NSClient (check_nt)*) und NSCA. Die Installation des NSClient++-Agenten habe wir im Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* beschrieben.

NSClient++ vs. NSClient: NSClient++ ist ein mächtiger Agent für Windows-Systeme, der die Protokolle NSClient, NRPE und NSCA unterstützt.

Über das check_nrpe Plugin lassen sich mit Hilfe des von NSClient++ bereitgestellten NRPE-Agenten viele komplexere Abfragen eines Windows-Systems umsetzen. Dazu gehört beispielsweise das detaillierte Auslesen und Auswerten von Protokolldateien, das Überwachen des Aktualisierungs-Status und das Ausführen von eigenen Scripten (siehe dazu Rezept 8.2, *Überwachung von Windows-Maschinen*). Die Konfiguration von NRPE ist allerdings auch aufwendiger als die von NSClient. Dabei sollten Sie allerdings berücksichtigen, dass die Entwicklung des NSClient-Protokols nach unserer Einschätzung nicht weiter verfolgt wird.

Wir haben den Verlauf einer Abfrage mit check_nrpe an einen Windows-Computer mit einem installiertem NSClient++-Agenten in Abbildung 5-10 dargestellt. Die Kommunikation erfolgt über den TCP-Port 5666. Sie müssen bei einer eventuell vorhandenen Firewall darauf achten, dass die Kommunikation vom Monitoring-Server zum zu überwachenden Client möglich ist.

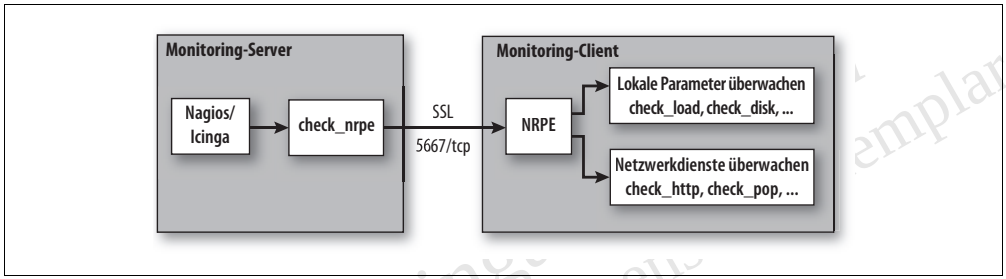


Abbildung 5-10: Abfrage einer Maschine mit NRPE

Befehlsdefinition und notwendige Optionen

Wir setzen im Folgenden voraus, dass Sie einen NRPE-Agenten auf dem zu überwachenden Zielsystem installiert haben. Für einen ersten Funktionstest können Sie das Plugin mit der IP-Adresse des Zielsystems und ohne weitere Parameter wie folgt aufrufen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nrpe -H CLIENTIP
NRPE v2.14
```

CHECK_NRPE: Error: Die Fehlermeldung `CHECK_NRPE: Error - Could not complete SSL handshake` weist nicht unbedingt auf ein Problem mit den SSL-Verschlüsselung hin. Es kann sich dabei auch um eine fehlerhafte Konfiguration des NRPE-Agenten auf dem Zielsystem handeln.

Wenn dieser erste Test erfolgreich verläuft, können Sie auf Ihrem Monitoring-Server ein Kommando definieren. Eine allgemeine Kommandodefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) für check_nrpe ohne Parameterübergabe sieht wie folgt aus:

```
# 'check_nrpe' command definition
define command {
       command_name                    check_nrpe
       command_line                    $USER1$/check_nrpe -H $HOSTADDRESS$ \
                                       -t 60 -p 5666 -c $ARG1$
}
```

Beachten Sie, dass NRPE keine Passwörter verwendet. Die Kommunikation erfolgt SSL-verschlüsselt und der Zugriff wird seitens des NRPE-Agenten auf notwendige IP-Adressen eingeschränkt (mindestens die Ihres Monitoring- Servers).

Als Pflichtangaben müssen Sie dabei die folgenden Optionen setzen:

- `-H`

  Geben Sie hier die Zielmaschine an. Wir verwenden hier das entsprechende Makro (siehe Rezept 4.3, *Befehle hinzufügen (command)*).

- `-t`

  An dieser Stelle geben Sie die Anzahl von Sekunden an, bevor das Plugin die Verbindung beendet. Der Standardwert ist 10 Sekunden, aber wir empfehlen Ihnen einen Wert von mindesten 60 Sekunden.

- `-p`

  Wenn Sie einen vom Vorgabewert 5666 abweichenden Port verwenden möchten, müssen Sie ihn, wie hier im Beispiel, mit dieser Option spezifizieren.

- `-c`

  Tragen Sie hier das auf dem Agenten auszuführende Kommando ein. Dabei sollte es sich aus Sicherheitsgründen um ein Alias-Kommando handeln, das auf dem Client-Rechner vorkonfiguriert ist.

Wir gehen auf die Konfiguration von Alias-Kommandos im Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* ein. Beispiele für das Auslesen von CPU-Auslastung, Speicher- und Festplatten-Nutzung, Windows-Zählern , Prozessen und Diensten haben wir für Sie im Rezept 8.1, *Überwachung von Unix-/Linux-Servern* und Rezept 8.2, *Überwachung von Windows-Maschinen* zusammengestellt.

Weitere Optionen

Manch ein unter dem Parameter `-c` angegebenes Kommando mag den Einsatz des folgenden Schalters erfordern:

- `-a`

  Optional tragen Sie hier zusätzliche Parameter ein. Dies ist seitens des NSClient++ NRPE-Agenten als Standardeinstellung aus Sicherheitsgründen nicht zulässig. Sie können dies zwar explizit freischalten, sollten aber bevorzugt auf definierte Alias-Kommandos zurückgreifen.

Dienstdefinition

Um entsprechende Prüfungen in Ihr Nagios/Icinga-Monitoringsystem einzubinden, benötigen Sie jetzt noch entsprechende Dienstdefinitionen (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*). Als Beispiel verwenden wir im Folgenden die

Abfrage der Prozessorlast auf einem Unix-System. Wir nutzen dabei das bei dem NRPE-Agenten mitgelieferte Alias-Kommando `check_load`, welches auf dem zu überwachenden System das im Verzeichnis */usr/lib/nagios/plugins/* hinterlegte Nagiosplugin `check_load` aufruft (wenn Sie die Plugins woanders abgelegt haben, müssen Sie den Pfad an dieser Stelle anpassen):

```
command[check_load]=/usr/lib/nagios/plugins/check_load -w 15,10,5 -c 30,25,20
```

Auf Ihrem Monitoring-System können Sie dann folgende Dienstdefinition verwenden:

```
define service{
       service_description                    CPU Load nrpe
       use                                    template-service-linux
       check_command                          check_nrpe!check_load
       hostgroup_name                         hg-linux
}
```

Auf einem Windows-System mit installiertem NSClient++-Agenten gibt es ebenfalls ein bereits vordefiniertes Alias-Kommando für die Abfrage der Prozessorlast namens `alias_cpu`:

```
alias_cpu = checkCPU warn=80 crit=90 time=5m time=1m time=30s
```

Hier wird das mit NSClient++ mitinstallierte Prüfmodul `checkCPU` aufgerufen, das dann die gewünschten Werte aus dem Windows-System ausliest. Die entsprechende Dienstdefinition für Ihr Nagios/Icinga-Monitoringsystem sieht dann wie folgt aus:

```
define service{
       service_description                    CPU Load nrpe
       use                                    template-service-win
       check_command                          check_nrpe!alias_cpu
       hostgroup_name                         hg-win
}
```

Wir verwenden hier die Vorlagen `template-service-win` beziehungsweise `template-service-linux` und weisen den Service allen Maschinen der Gerätegruppe `hg-win` beziehungsweise `hg-linux` zu. Passen Sie diese Einstellungen gegebenenfalls an Ihre Umgebung an.

Alias-Kommandos: Wenn Sie die Alias-Kommandos auf den Zielsystemen (unter Unix mit NRPE und unter Windows mit NSClient++) identisch benennen, könnten Sie alle zu überwachenden Systeme mit einer Dienstdefinition abdecken.

Test, Rückgabewerte und Graphen

Sie sollten die Ausführung des Plugins vor der Einbindung testen. Dazu stellen Sie als erstes sicher, dass Sie NRPE beziehungsweise NSClient++ auf dem betreffenden Zielsystem installiert haben (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* oder Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*), für die eben gezeigte Abfrage der CPU-Last etwa lässt sich das nachfolgende Kommando auf Ihrem Monitoring-Server nutzen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch das Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit dem Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nrpe -H 10.85.58.113 -t 60 -p 5666
-c alias_cpu
OK CPU Load ok.|'5m'=43%;80;90 '1m'=43%;80;90 '30s'=46%;80;90
```

Das Plugin `check_nrpe` liefert, abhängig von den Rückgabewerten und sofern möglich, Performancedaten zurück. Sofern Sie PNP4Nagios eingerichtet haben, werden diese dann automatisch ausgewertet. Im Rezept 8.2, *Überwachung von Windows-Maschinen* finden Sie zahlreiche Beispiele für den Einsatz dieses Plugins.

Diskussion

Die Agenten NRPE und NSClient++ verbieten beide standardmäßig die Übergabe von Parametern. Wir empfehlen Ihnen aus Sicherheitsgründen ausdrücklich, diese Sperre beizubehalten. Dies hat allerdings den Nachteil, dass Sie für jeden Test ein eigens definiertes Alias-Kommando benötigen. Wenn Sie die Übergabe von Parametern erlauben möchten, müssen Sie den Agenten auf dem Zielsystem entsprechend anpassen. Unter NRPE ist dies der Parameter `dont_blame_nrpe=1` und unter NSClient der Parameter `allow arguments = true` im Abschnitt `[/settings/NRPE/server]`. Aus Sicherheitsgründen raten wir Ihnen allerdings davon ab. Die detaillierte Konfiguration haben wir in den Rezepten zur Installation des Agenten beschrieben (siehe dazu Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* beziehungsweise Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*).

Sie können über NRPE nicht nur lokale Werte auf dem Zielsystem abfragen, sondern können auch die Erreichbarkeit von Netzwerkdiensten abfragen. Dies macht Sinn, wenn der Dienst beispielsweise vom Monitoring-Server aus nicht direkt erreichbar ist.

Siehe auch

- Rezept zur Installation von NRPE unter Windows: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*
- Rezept zur Installation von NRPE unter Unix: Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- NRPE auf Sourceforge: *http://sourceforge.net/projects/nagios/files/nrpe-2.x/*
- Webseiten des Projekts NSClient++: *http://www.nsclient.org/*
- NSClient++ auf Sourceforge: *http://sourceforge.net/projects/nscplus/files/nscplus/*

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Überwachung eines Windows-Rechners mit NSClient: Rezept 5.9, *Überwachung einer Windows-Maschine mit NSClient (check_nt)*
- Rezepte zur generellen Überwachung von Unix/Linux- und Windows-Maschinen: Rezept 8.1, *Überwachung von Unix-/Linux-Servern* und Rezept 8.2, *Überwachung von Windows-Maschinen*

5.9 Überwachung einer Windows-Maschine mit NSClient (check_nt)

Problem

Sie möchten eine Windows-Maschine überwachen. Die Abfrage Ihres Monitoring-Systems wollen Sie mit Hilfe von NSClient durchführen

Lösung

Bei der Installation der Nagiosplugins ist das Plugin `check_nt` enthalten. Damit lässt sich ein auf einem Windows Recher installierter Agent über das NSClient-Protokoll abfragen. NSClient wird von Agenten wie NSClient, NC_Net und NSClient++ unterstützt. Der Agent NSClient wird nicht mehr weiterentwickelt. Der Nachfolger von NSClient ist das aufrufkompatible Paket NC_Net, das allerdings ein erweitertes `check_nt`-Plugin benötigt. Das Paket NSClient++ ist ebenfalls kompatibel mit NSClient, unterstützt aber unter anderem zusätzlich den Dienst NRPE (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*).

Das `check_nt`-Plugin unterstützt nur eine Reihe von Basisprüfungen, wie Prozessor-Auslastung sowie Speicher- und Festplatten-Nutzung. Für komplexere Prüfungen empfehlen wir den Umstieg auf NRPE, das allerdings aufwendiger in der Konfiguration ist.

Sofern Sie das Protokoll NSClient für das Monitoring von entfernten Windows-Maschinen einsetzen wollen, empfehlen wir Ihnen, `check_nt` in Kombination mit dem Windows-Agenten `NSClient++` zu verwenden. Die Installation von `NSClient++` haben wir im Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* beschrieben.

Der Verlauf einer Abfrage mit `check_nt` an einen Windows-Computer mit einem installiertem NSClient++-Agenten haben wir in Abbildung 5-11 dargestellt. Beachten Sie, dass die Kommunikation standardmäßig über den TCP-Port 12489 erfolgt. Bei einer eventuell vorhandenen Firewall müssen Sie darauf achten, dass eine Kommunikation vom Monitoring-Server zum zur überwachenden Windows-Maschine möglich ist.

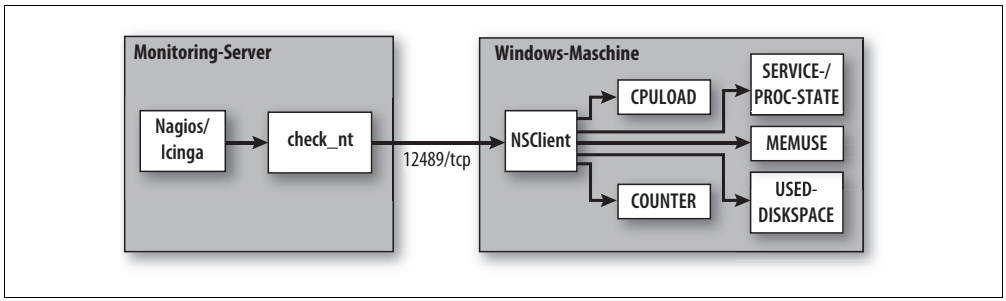


Abbildung 5-11: Abfrage einer Windows-Maschine mit NSClient

Im folgenden Abschnitt zeigen wir Ihnen, wie Sie darauf aufbauend das Plugin `check_nt` einbinden.

`check_nt` vs. `check_nrpe`: Die Entwickler des NSClient++-Agenten empfehlen ausdrücklich die Verwendung von NRPE (Plugin `check_nrpe`) anstelle von NSClient (Plugin `check_nt`). Trotzdem ist die Nutzung des Plugins `check_nt`, gerade als Einstieg in die Welt des Windows-Monitorings mit Nagios/Icinga, sehr hilfreich. Vorteil ist hier, dass die Abfrage über das Protokoll NSClient sehr einfach zu realisieren, ausgereift und weit verbreitet ist. Außerdem funktioniert NSClient auf allen Windows-Varianten seit Windows NT. Trotzdem sollten Sie in Betracht ziehen, die Werte stattdessen mit `check_nrpe` zu ermitteln, da Ihnen dieses wesentlich mehr Möglichkeiten bietet (siehe hierzu auch die nachfolgende Diskussion und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*).

Befehlsdefinition und notwendige Optionen

Für das Plugin wurde bei der Installation von den Standard-Plugins bereits eine Befehlsdefinition erstellt (siehe Rezept 4.3, *Befehle hinzufügen (command)*). Bei unserer Referenzinstallation war dies die folgende Definition:

```
# 'check_nt' command definition
define command{
       command_name                    check_nt
       command_line                    $USER1$/check_nt -H $HOSTADDRESS$ \
                                       -p 12489 -v $ARG1$ $ARG2$
 }
```

Davon ausgehend, dass Sie bei der Installation ein Passwort für die Benutzung angeben haben, müssten Sie dies bei dieser Einrichtung allerdings jedes Mal in dem entsprechenden Service definieren. Insbesondere bei Änderungen des Passworts müssten Sie die Konfiguration dann jedoch an mehreren Stellen nachpflegen, weshalb wir Ihnen eine andere Definition empfehlen. Hinterlegen Sie das Passwort als Benutzermakro (siehe Rezept 4.3, *Befehle hinzufügen (command)*) in der Datei `/usr/local/icinga/etc/resource.cfg`:

```
[...]
# Windows-NSClient-PW
$USER13$="PASSWORT"
```

Referenzieren Sie dieses Makro für das Passwort dann folgendermaßen in der Kommandodefinition:

```
# check_nt
define command {
       command_name                    check_nt
       command_line                    $USER1$/check_nt -H $HOSTADDRESS$ \
                                       -p 12489 -s $USER13$ -v $ARG1$ $ARG2$
}
```

Als Pflichtangaben müssen Sie dabei die folgenden Optionen setzen:

- `-H / --hostname`

  Hier geben Sie die Zielmaschine an. Wir verwenden dafür an dieser Stelle das entsprechende Makro (siehe Rezept 4.3, *Befehle hinzufügen (command)*).

- `-p / --port`

  Sofern Sie einen vom Vorgabewert 12489 abweichenden Port verwenden, müssen Sie ihn wie im Beispiel mit dieser Option spezifizieren.

- `-v / --variable`

  Mit dieser Option geben Sie an, welchen Wert auf der Zielmaschine Sie abfragen möchten. Ihnen stehen dabei die folgenden Schalter zur Verfügung, die gegebenenfalls weitere Angaben über den Schalter `-l` erforderlich machen:

  – `CLIENTVERSION`

    Hiermit fragen Sie die laufende Version von NSClient++ ab.

  – `CPULOAD`

    Hiermit fragen Sie den Load der CPU ab. Über die Option `-l` geben Sie dabei an, für welche Zeiträume in Minuten Sie den Wert wünschen und welche Schwellen für `WARNING` und `CRITICAL` Sie verwenden möchten, beispielsweise `-l 1,90,95,5,90,95,15,90,95` für die bei Linux typischen Zeiträume 1, 5 und 15 Minuten, jeweils mit Schwellen bei 90% und 95%.

  – `UPTIME`

    Hiermit fragen Sie ab, wie lange die Maschine schon läuft.

  – `USEDDISKSPACE`

    Hiermit fragen Sie Größe und Füllstand von Laufwerken ab, die Sie durch den Windows-typischen Laufwerksbuchstaben mit der Option `-l` spezifizieren. Also etwa `-l c` für das Laufwerk C:\. Schwellenwerte für `WARNING` und `CRITICAL` können Sie dabei mit den Optionen `-w` und `-c` angeben (siehe im Nachfolgenden).

  – `MEMUSE`

    Mit diesem Wert fragen Sie die Speicherauslastung ab. Schwellenwerte für `WARNING` und `CRITICAL` können Sie dabei wieder mit den Optionen `-w` und `-c` angeben (siehe im Nachfolgenden).

– SERVICESTATE

  Mit diesem Wert veranlassen Sie die Prüfung von Diensten, deren Servicenamen Sie durch Komma separiert mit der Option -l übergeben. Um etwa NSClient++ selbst abzufragen -l NSClientpp. Beim Test von der Kommandozeile können Sie zusätzlich noch die Option -d SHOWALL einsetzen, um eine Ausgabe für jeden der abgefragten Dienste anzuzeigen.

– PROCSTATE

  Wenn Sie anstelle von Diensten konkrete Prozesse abfragen möchten, können Sie dies mit diesem Wert erreichen. Übergeben Sie dazu den Namen des Prozesses mit der Option -l. Um etwa den Prozess von NSClient++ abzufragen, nutzen Sie beispielsweise -l "nsclient++.exe". Auch hier können Sie beim Test von der Kommandozeile zusätzlich noch die Option -d SHOWALL einsetzen, um eine Ausgabe für jeden der abgefragten Prozesse anzuzeigen.

– COUNTER

  Performancedaten liegen in Windows als sogenannte Zähler vor, die Sie mit diesem Wert für das Plugin abfragen können. Den abzufragenden Wert spezifizieren Sie dazu mit der Option -l. Als Beispiel hier die Abfrage der geschriebenen Bytes auf logische Laufwerke: -l "\\LogicalDisk\\Avg. Disk Bytes/Write". Achten Sie dabei bitte auf das hier notwendige Backslash-Escape. Sie können dieser Spezifikation des abzufragenden Wertes durch ein Komma separiert noch eine Beschriftung im Format des Befehls printf hinzufügen.

Wir gehen in Rezept 8.2, *Überwachung von Windows-Maschinen* ausführlicher auf das Auslesen von Windows-Zählern beziehungsweise Abfragen zu Prozessen und Diensten über check_nt ein. Dort haben wir auch Beispiele zusammengestellt.

Manche Werte für den Parameter -v erfordern den Einsatz der im Folgenden beschriebenen und ansonsten optionalen Schalter.

Weitere Optionen

Als weitere Optionen stehen Ihnen die folgenden zur Verfügung:

- -w / --warning und -c / --critical

  Sofern von der Abfrage unterstützt, können Sie mit diesen Optionen die Schwellenwerte für WARNING und CRITICAL angeben.

- -t / --timeout

  Mit dieser Option können Sie die für eine Abfrage maximal zur Verfügung stehende Zeit in Sekunden festlegen. Nach Ablauf dieser Zeit wird die Abfrage abgebrochen und ein Fehler zurückgeliefert.

- -l

  Mit dieser Option geben Sie gegebenenfalls spezifische Parameter je nach gewählter Variable an (siehe dazu Schalter -v).

Dienstdefinition

Zur Einbindung des Abrufs einer der Variablen können Sie dann eine entsprechende Dienstdefinition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) erstellen. Aufbauend auf der obigen Befehlsdefinition hier ein Beispiel für die Abfrage der Prozessorlast mit Hilfe der Variable `CPULOAD`:

```
define service{
       service_description                    CPU Load nisc
       use                                    template-service-win
       check_command                          check_nt!CPULOAD!-l 5,80,90
       hostgroup_name                         hg-win
}
```

In diesem Beispiel verwenden wir eine Vorlage `template-service-win`, weisen den Service allen Maschinen der Gerätegruppe `hg-win` zu und spezifizieren für die Abfrage den 5-Minuten-Load mit den Schwellenwerten 80% für `WARNING` und 90% für `CRITICAL`. Passen Sie diese Werte gegebenenfalls an Ihre Umgebung an.

Test, Rückgabewerte und Graphen

Wie immer sollten Sie die Ausführung des Plugins vor der Einbindung testen. Stellen Sie sicher, dass Sie `NSClient++` auf dem betreffenden Windows-Rechner installiert haben (siehe Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*). Für die eben gezeigte Abfrage der CPU-Last führen Sie auf Ihrem Monitoring-Server folgendes Kommando aus:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch das Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit dem Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nt -H 10.85.58.113 -p 12489 -s
PASSWORD -v CPULOAD -l 5,80,90
CPU Load 41% (5 min average) |   '5 min avg Load'=41%;80;90;0;100
```

In diesem Fall gibt das Plugin auch entsprechende Performance-Daten zurück. Wenn Sie also PNP4Nagios eingerichtet haben, dann erhalten Sie automatisch einen entsprechenden Graphen wie beispielsweise in Abbildung 5-12 für die hier gezeigte Abfrage der Systemlast.

Auf diese Weise können Sie einfach wichtige Daten einer Windows-Maschine abfragen. Im Rezept 8.2, *Überwachung von Windows-Maschinen* finden Sie weitere Beispiele für den Einsatz dieses Plugins.

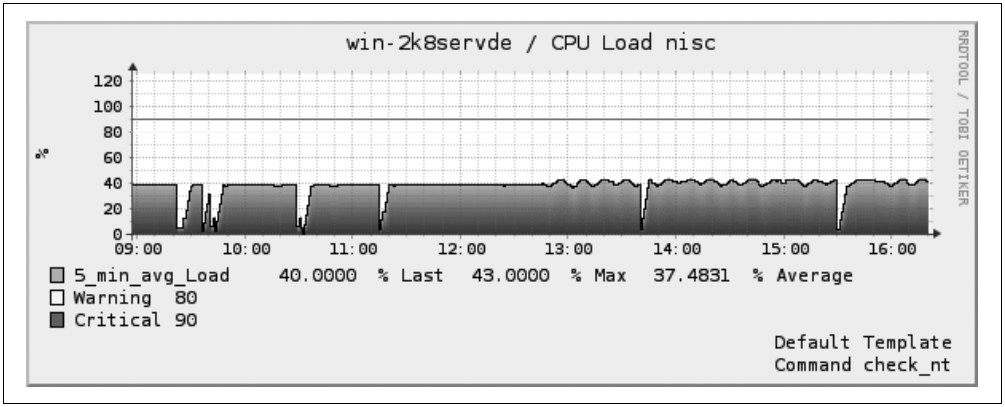


Abbildung 5-12: PNP4Nagios-Graph für über check_nt ausgelesene CPU-Daten

Diskussion

Es wurde bereits erwähnt, dass die Entwickler des NSClient++-Agenten nach unserer Einschätzung den Zugriff über das Protokoll NSClient durch das Plugin `check_nt` künftig nicht mehr unterstützen werden. Außer bei Alt-Installation sollten Sie Ihre Abfragen mittelfristig auf NRPE umstellen, also über das in Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)* beschriebene Plugin `check_nrpe` durchführen.

Siehe auch

- Rezept zur Vorbereitung von Windows-Maschinen auf Abfragen über NSClient und/oder NRPE: Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Webseiten des Projekts NSClient++: *http://www.nsclient.org/*
- NSClient++ auf Sourceforge: *http://sourceforge.net/projects/nscplus/files/nscplus/*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezepte zur Remote-Überwachung eines Windows-Rechners über NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)* und Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Rezept zur Überwachung eines Windows-Rechners: Rezept 8.2, *Überwachung von Windows-Maschinen*

5.10 Entfernte Ausführung über SSH (check_by_ssh)

Problem

Sie möchten auf einer entfernten Maschine lokale Plugins ausführen. Die Einbindung in Ihr Monitoring-System möchten Sie mit Hilfe der Secure Shell (SSH) vornehmen.

Lösung

Sie können das Plugin `check_by_ssh` verwenden, um über SSH gesichert Programme auf dem entfernten System auszuführen. Dabei wird dieses Plugin lokal ausgeführt, dient aber eben dazu, ein anderes Plugin auf einer entfernten Maschine auszuführen.

Voraussetzung ist, dass auf dem Zielsystem ein SSH-Server läuft und entsprechend konfiguriert ist. Diese Installation und Konfiguration des SSH-Servers beschreiben wir in Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten*. Wir gehen im Folgenden von einer Konfiguration wie in diesem Rezept beschrieben aus.

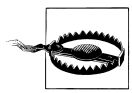
Plugin und Schlüssel vorbereiten

Bringen Sie zunächst das betreffende Plugin auf dem Zielsystem zum Laufen. Bei als Script realisierten Plugins ist dies häufig unproblematisch, da Sie nur für die passende Laufzeitumgebung sorgen müssen. Hier ist häufig lediglich eine Installation von PHP oder Perl erforderlich. Bei Binärdateien müssen Sie hingegen in aller Regel den Übersetzungsvorgang auf dem Zielsystem durchführen, also etwa die Standard-Plugins installieren, wie in den Rezepten in Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten* beschrieben.

Wir zeigen Ihnen im Folgenden die Integration am Beispiel des Plugins `check_apt` (siehe hierzu Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*), welches sich auf dem Zielsystem im Ordner */root/libexec* befindet. Stellen Sie zunächst lokal auf dem Zielsystem sicher, dass das Plugin wie gewünscht funktioniert. In unserem Beispiel sähe das folgendermaßen aus:

```
root@server:~# /root/libexec/check_apt
APT CRITICAL: 31 packages available for upgrade (9 critical updates).
```

Richten Sie dann nach Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten* einen SSH-Schlüssel für die Ausführung dieses Befehls in der Datei *~/.ssh/authorized_keys* ein. In unserem Falle für `check_apt` funktioniert das wie folgt:



Beschränkung der SSH-Schlüssel: Wir empfehlen Ihnen bei Schlüsseln ohne Passphrase unbedingt Beschränkungen wie hier vorgeschlagen zu verwenden. Wenn anderenfalls der Schlüssel in die falschen Hände gelangt, lassen sich mit diesem beliebige Kommandos auf dem Zielsystem ausführen.

```
command="/root/libexec/check_apt",from="moni",no-agent-forwarding,no-port-forwarding,
no-pty,no-user-rc,no-X11-forwarding ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC44KZFzYGGBW
nw2+wUKZYUqad1B4gllMksLBk3mWOmEFeKSGD8c+ACDcp5R2pgUluFXBcwllZoPOez2neEHZM/kkftaZOeBTFY/E1
UdaaWW8wXSeZTL3Y9quyvf6MSAIH9o+yqCEzYtjzj691vY1lwsOaICc6tmK4JG6xQfQJ3XZaUF/X1dXEcRMa4yIsd
koLociCZvwXWxypOJoWfMZLBhu3rOn2dLvfrZDfhhROMWjULGm5Ll3oEgYzay7zfSXaL8o+qzDhH2I6jGj+5vXFZm
QrnEXODqyU2tr/r/ipMzXCuorVkuFiYHpkycoqmhHguBhXrEUSfPyY/atvOIKU3 icinga@moni
```

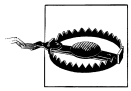
Entsprechend müssen Sie dann allerdings für alle auf diese Art auszuführenden Plugins jeweils eigene Schlüssel erstellen und den entsprechenden Befehl, wie hier gezeigt, in der Datei *authorized_keys* hinterlegen.

Test auf der Monitoring-Maschine

Testen Sie als Nächstes auf der Monitoring-Maschine, dass Sie den Befehl mit dem passenden Schlüssel über SSH ausführen können. Dieser Test ist hier unbedingt erforderlich und sieht für das gerade angeführte Beispiel etwa wie folgt aus:

```
icinga@moni:~$ ssh -T -i ~/.ssh/id_rsa-check_apt root@server
APT CRITICAL: 31 packages available for upgrade (9 critical updates).
```

Bei der erstmaligen Ausführung werden Sie dabei wie in Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten* beschrieben den Fingerabdruck des entsprechenden Zielrechners abgleichen und bestätigen müssen.



Notwendige manuelle Ausführung: Ohne diesen Schritt wird die automatische Durchführung fehlschlagen. Das SSH-System wird in diesem Fall nach der Authentifizierung des Fingerabdrucks fragen, die Nagios/Icinga aber nicht beantworten kann. Entsprechend erhalten Sie dann einen Timeout des Plugins.

Als Nächstes testen Sie die Ausführung über das Plugin `check_by_ssh`. Das Plugin erwartet in jedem Falle die Übergabe eines Befehls mit dem Parameter `-C`. Wir haben hier den auszuführenden Befehl allerdings bereits mit dem Schlüssel auf der Zielmaschine hinterlegt, so dass Sie hier einfach irgendetwas als Platzhalter übergeben:

```
icinga@moni:~$ ./check_by_ssh -i ~/.ssh/id_rsa-check_apt -l root -H server -C dummy
APT CRITICAL: 31 packages available for upgrade (9 critical updates).
```

Befehlsdefinition und notwendige Optionen

Sie benötigen dann einen entsprechenden Befehl für Nagios/Icinga (siehe hierzu auch Rezept 4.3, *Befehle hinzufügen (command)*):

```
define command{
       command_name                    check_ssh_apt
       command_line                    $USER1$/check_by_ssh \
                                       -i /home/icinga/.ssh/id_rsa-check_apt \
                                       -l root \
                                       -H $HOSTADDRESS$ \
                                       -C dummy
}
```

Dabei sind in der Definition zunächst wieder die Pflichtoptionen vorgesehen:

- `-H` / `--hostname`

  Wie üblich müssen Sie angeben, an welche Maschine Nagios/Icinga die Ausführung richten sollen. Wie bei den meisten Plugins verwenden Sie hier am besten das Makro $HOSTADDRESS$.

- `-C` / `--command`

  Es wurde bereits angesprochen, dass Sie ein auszuführendes Kommando übergeben müssen. Bei einer Konfiguration wie der hier vorgeschlagenen ist das Kommando allerdings mit dem Schlüssel hinterlegt, so dass Sie zwar ein Kommando übergeben müssen, dies aber ignoriert werden wird. Sie verwenden dann wie beschrieben mit `dummy` eine beliebige Zeichenkette.

Die anderen beiden im vorliegenden Beispiel verwendeten Parameter sind aus Sicht des Plugins optional (siehe folgender Abschnitt). Wir haben Sie hier aber angeführt, weil sie in dem von uns geschilderten Szenario notwendig sind. In einer weniger sicherheitsbewussten Konfiguration ohne im Schlüssel hinterlegten Befehl könnten Sie diese weglassen, wovon wir Ihnen jedoch abraten.

Weitere Optionen

Über folgende Optionen können Sie das Plugin weiter an Ihre Bedürfnisse anpassen:

- `-p` / `--port`

  Wenn Sie nicht den Standard-Port von SSH verwenden, können Sie mit dieser Option den zu verwendenden Port angeben.

- `-4` / `--use-ipv4` oder `-6` / `--use-ipv6`

  Standardmäßig wird das Plugin Version 4 von IP verwenden. Wenn Ihre Umgebung dies erfordert, können Sie mit dieser Option aber auch festlegen, dass Version 6 von IP verwendet werden soll.

- `-1` / `--proto1` oder `-2` / `--proto2`

  Mit dieser Option können Sie die zu verwendende Version des SSH-Protokolls festlegen.

- `-S` / `--skip-stdout` und/oder `-E` / `--skip-stderr`

  Wenn Sie die Standard- (`-S` beziehungsweise `--skip-stdout`) und/oder die Fehlerausgabe (`-E` beziehungsweise `--skip-stderr`) ignorieren möchten, können Sie mit dieser Option die Anzahl der zu ignorierenden Zeilen angeben. Wenn Sie keine Anzahl angeben, werden alle Ausgaben ignoriert.

- `-f`

  In der Voreinstellung wird das Plugin auf der Zielmaschine ein Terminal (**`tty`**) erstellen. Mit dieser Option können Sie veranlassen, dass es stattdessen einen Unterprozess des laufenden SSH-Servers verwendet.

- `-l` / `--logname`

  In der Voreinstellung wird das Plugin für die Anmeldung an der Maschine den Benutzernamen verwenden, unter dem es auf der lokalen Maschine ausgeführt wird. Um einen anderen Benutzernamen zu verwenden, spezifizieren Sie ihn mit dieser Option. Wir haben diese Vorgehensweise im vorangegangenen Beispiel verwendet.

- `-i` / `--identity`

  Für jeden Benutzer ist standardmäßig die Ablage eines Schlüssels vorgesehen. Dieser wird mit den Standardeinstellungen automatisch verwendet. Diese Option erlaubt Ihnen, einen abweichenden Schlüssel anzugeben. Wir haben diese Möglichkeit im vorangegangenem Beispiel genutzt, da bei der von uns vorgeschlagenen Konfigurationsmethode für jedes Plugin ja ein eigener Schlüssel erforderlich ist.

- `-o` / `--ssh-option`

  Das Plugin verwendet die im System für SSH hinterlegten Optionen automatisch. Mit dieser Option können Sie davon abweichende und/oder weitere Optionen für das SSH-Subsystem festlegen.

- `-q` / `--quied`

  In Abhängigkeit von Umgebung und Konfiguration können Warnmeldungen des SSH-Subsystems den Aufruf und damit Ablauf der Prüfung stören. Mit dieser Option können Sie veranlassen, dass alle diagnostischen Ausgaben und Warnausgaben ignoriert werden und die Ausführung in solchen Fällen dennoch funktioniert.

- `-w` / `--warning` und `-c` / `--critical`

  Typischerweise ist bei der Nutzung dieses Plugins gewünscht, dass der resultierende Status von dem ausgeführten Plugin abhängt. Wenn Sie stark schwankende Ausführungszeiten bei der Ausführung über SSH feststellen, können Sie mit diesen Optionen Schwellenwerte für diese hinterlegen. Das Plugin liefert dann allerdings gegebenenfalls einen Fehlerstatus, obwohl das dahinterliegende Plugin möglicherweise `OK` zurückgegeben hat.

- `-t` / `--timeout`

  Schließlich können Sie noch veranlassen, dass die Ausführung nach einem bestimmten Zeitraum abgebrochen wird. Der Vorgabewert für diesen Timeout ist 10 Sekunden. Sie können mit dieser Option aber auch einen davon abweichenden Zeitraum in Sekunden angeben.

Vollständige Pfadangaben: Verwenden Sie für die Optionen gegebenenfalls vollständige Pfadangaben, wie im eben angeführten Beispiel also */home/icinga* anstatt ~. Sie vermeiden damit Probleme, die durch die geänderte Umgebung bei Ausführung der Befehle durch das Monitoring entstehen können.

Dienstdefinition

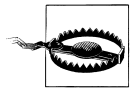
Die Kommandodefinition können Sie dann über eine entsprechende Service-Definition (siehe hierzu Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) ausführen lassen:

```
define service{
       service_description                    check_ssh_apt
       use                                    template-srv-hcksp
       check_command                          check_ssh_apt
       host_name                              server
}
```

Damit führen Sie das nur lokal ausführbare Plugin `check_apt` auf einem entfernten System aus und haben die Ausgaben in Ihrem Nagios/Icinga zur Verfügung. Analog können Sie auch mit allen anderen lokalen Plugins in diesem Kapitel verfahren.

Diskussion

Sofern auf Ihren Maschinen SSH-Server laufen, können Sie die Ausführung von Plugins mit den gezeigten Schritten relativ leicht auf diese Maschinen ausdehnen. Alle Parameter für die Ausführung sind dabei allerdings mit dem Schlüssel hinterlegt, so dass Sie Änderungen gegebenenfalls nicht zentral auf der Monitoring-Maschine vornehmen können. Sie müssen geänderte Parameter dann hingegen auf den jeweiligen Servern in den SSH-Konfigurationen anpassen.



Sicherheit: Wählen Sie zur Erhöhung der Sicherheit Benutzerkonten und SSH-Freigaben mit ausschließlich den benötigten Rechten. Sie müssen dann allerdings für jedes Plugin einen neuen Schlüssel erstellen und verteilen und bei Änderungen die entsprechenden SSH-Dateien anpassen. Dafür kann ein Angreifer, der Ihre Monitoring-Maschine übernommen hat, nicht gleich vollständig auf alle überwachten Server zugreifen. Das wäre nämlich das Risiko, das Sie in Kauf nähmen, wenn Sie jeweils etwa einen unbeschränkten Zugang für den Benutzer `root` wählen würden.

Die Einrichtung, die System-Belastung bei der Durchführung und der Aufwand bei Änderungen sind bei diesem Durchführungsweg allerdings eben nicht unerheblich. Unsere Empfehlung zur Remote-Ausführung lokaler Plugins ist deshalb generell der Weg über SNMP (siehe hierzu Rezept 2.5, *Plugins unter Linux über SNMP ausführen* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*). Den hier beschriebenen Weg über SSH können und sollten Sie deshalb vornehmlich dann verwenden, wenn Sie auf der beteroffenen Maschine keinen SNMP-Server verwenden können oder wollen.

SSH-Multiplexing: Sofern sie viele Plugins über SSH ausführen, sollten Sie sich den Mechanismus des SSH-Multiplexing anschauen (siehe hierzu Multiplexing im englischen Kochbuch zu OpenSSH unter *https://en.wikibooks.org/wiki/OpenSSH/Cookbook/Multiplexing*). Mit diesem können Sie aufgebaute SSH-Verbindungen wiederverwenden und dadurch die Rechenlast und Ausführungszeiten erheblich senken. Der Einsatz bedingt aber die Kapselung in einem weiteren Script, weshalb wir darauf hier nicht weiter eingehen.

Siehe auch

- Rezept zur Einrichtung des SSH-Servers unter Linux: Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten*
- Seiten des OpenSSH-Projekts: *http://www.openssh.com/*
- Kapitel zur Installation inklusive der Standard-Plugins: Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*
- Rezept zur Prüfung auf Systemaktualisierungen mit `check_apt`: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*
- Rezept zu Befehlsdefinitionen: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezepte zur Remote-Ausführung von lokalen Plugins über SNMP und Auswertung der entsprechenden Ergebnisse: Rezept 2.5, *Plugins unter Linux über SNMP ausführen* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*
- Multiplexing im OpenSSH-Kochbuch (englisch): *https://en.wikibooks.org/wiki/OpenSSH/Cookbook/Multiplexing*

5.11 Überwachen auf Aktualisierungen des Systems (check_apt)

Problem

Sie möchten ein Linux-System hinsichtlich verfügbarer Aktualisierungen überwachen.

Lösung

In den Standard-Plugins ist zu diesem Zweck bereits das Plugin check_apt enthalten, dass es Ihnen erlaubt, auf Debian basierende Systeme (Debian, Ubuntu, Mint und andere mehr) auf verfügbare Aktualisierungen hin zu prüfen. Wir stellen Ihnen dieses im Folgenden vor und zeigen Ihnen, wie Sie es einbinden und verwenden können.

Linux-Derivate: Wenn Ihr System nicht auf Debian basiert, können Sie das hier beschriebene Plugin nicht verwenden. Es stehen jedoch für andere Distributionen entsprechende Plugins bereit, die Sie analog einbinden und verwenden können. Im Falle von SuSE- und RHEL-basierenden Distributionen gibt es die Plugins check_zypper beziehungsweise check_updates. Beide sind Teil der jeweiligen Distribution. Wir gehen in diesem Buch aber nicht weiter auf diese ein, da sich die Anwendung im Wesentlichen nicht stark von `check_apt` unterscheidet.

Befehlsdefinition und notwendige Optionen

Eine einfache Einbindung von `check_apt` erreichen Sie mit der folgenden Befehlsdefinition (siehe hierzu auch Rezept 4.3, *Befehle hinzufügen (command)*):

```
define command{
       command_name                    check_apt
       command_line                    $USER1$/check_apt
}
```

Sie müssen diesem Plugin keine Optionen übergeben. Verfügbare Optionen beschreiben wir Ihnen im folgenden Abschnitt.

Weitere Optionen

Zur Anpassung an Ihre Bedürfnisse können Sie das Plugin mit den folgenden Optionen steuern:

- `-t` / `--timeout`

  Geben Sie einen Timeout in Sekunden vor, nach dem das Plugin aufhört, auf ein Ergebnis zu warten (Vorgabewert ist 10 Sekunden).

- `-u` / `--update`

  Mit dieser Option können Sie veranlassen, dass vor der Prüfung zunächst die Paketlisten aktualisiert werden. Sie können dabei Optionen für die ausgeführte Operation `apt-get update` übergeben.

- `-n` / `--no-upgrade`

  Durch diese Option wird ein mögliches Upgrade unterbunden.

- `-U` / `--upgrade` beziehungsweise `-d` / `--dist-upgrade`

  Mit diesen Optionen können Sie das System veranlassen, eine Aktualisierung durchzuführen. Sie können dabei Optionen an das dahinterliegende Aktualisierungssystem `apt` übergeben, auf die wir hier allerdings nicht weiter eingehen.

- `-i` / `--include` beziehungsweise `-e` / `--exclude`

  Wenn Sie mit einer der Optionen `-U` beziehungsweise `-d` eine Aktualisierung durchführen, können Sie mit Hilfe dieser Optionen einen regulären Ausdruck spezifizieren, der angibt, welche Pakete aktualisiert werden sollen (`-i`) beziehungsweise welche Pakete von der Aktualisierung ausgenommen werden sollen (`-e`). Vorgabe sind hier alle Pakete, die sich über die eingebundenen Paketquellen aktualisieren lassen.

`-c` / `--critical`

Sie können einschränken, welche Pakete bei verfügbaren Aktualisierungen den Status `CRITICAL` erzeugen sollen. Geben Sie dazu mit dieser Option einen entsprechenden regulären Ausdruck an. Sie können diese Option mehrfach verwenden. Vorgabe war in unserem Referenzsystem der reguläre Ausdruck `^[^\(]*\(.* (Debian-Security:|Ubuntu:[^/]*/[^-]*-security)`, der den kritischen Status auf Sicherheitsaktualisierungen beschränkt. Beachten Sie, dass bevor der Status `CRITICAL` erreicht wird, der Status `WARNING` ebenfalls erreicht sein muss.

Beachten Sie, dass die meisten dieser Optionen erfordern, dass das Plugin mit administrativen Rechten läuft (siehe hierzu Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten*).

Dienstdefinition

Die eben angeführte Befehlsdefinition können Sie mit der folgenden Dienstdefinition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) einbinden:

```
define service{
       service_description                     check_apt
       use                                     template-service-generic
       check_command                           check_apt
       host_name                               localhost
}
```

Beachten Sie bitte, dass dies zunächst nur für die lokale Maschine möglich ist. Für die Ausführung auf Remote-Maschinen lesen Sie bitte den nachfolgenden Diskussionsteil.

Test, Rückgabewerte und Graphen

Um das Plugin zu testen, können Sie es wie folgt auf der Kommandozeile ausführen (passen Sie den Pfad gegebenenfalls an Ihre Bedürfnisse an):

```
icinga@moni:~$ /usr/local/icinga/libexec/check_apt
APT OK: 0 packages available for upgrade (0 critical updates).
```

Ohne weitere Optionen liefert das Plugin den Status `WARNING`, sobald Aktualisierungen verfügbar sind, die aber nicht die Sicherheit betreffen, und `CRITICAL`, falls sicherheitsrelevante Aktualisierungen vorliegen.

Diskussion

Das Einspielen von Aktualisierungen, insbesondere wenn sie die Sicherheit betreffen, ist kritisch. Deshalb ist die Verwendung dieses Plugins sehr zu empfehlen. Sie können das Plugin auf die ausschließliche Prüfung von Sicherheitsupdates beschränken, indem Sie eine eigene Liste mit den entsprechenden Paketquellen vorgeben, die das Plugin lädt und aktualisiert. Diese beispielsweise unter `/etc/apt/sources.security.list` abgelegte Liste könnte für ein Debian Wheezy-System den folgenden Inhalt haben:

```
deb http://security.debian.org/ wheezy/updates main non-free contrib
deb-src http://security.debian.org/ wheezy/updates main non-free contrib
```

Ein Aufruf von apt-get mit den folgenden Parametern würde dann ausschließlich auf sicherheitsrelevante Updates prüfen:

```
root@moni:~# apt-get update -o Dir::Etc::SourceList=/etc/apt/sources.security.list
```

Ein entsprechend angepasster Aufruf von check_apt, der auf Sicherheitsupdates prüft, diese jedoch nicht einspielt, würde dann wie im Nachfolgenden angeführt aussehen. Da das Laden einer Paketquelle durchaus etwas dauern kann, haben wir hier den Timeout von 10 auf 60 Sekunden erhöht:

```
root@moni:~# /root/libexec/check_apt -t 60 -n -u'-o Dir::Etc::SourceList=
/etc/apt/sources.security.list'
```

Sie können das Plugin check_apt zunächst nur lokal einsetzen. An dieser Stelle sollten Sie sich deshalb gegebenenfalls die Rezepte zur entfernten Ausführung von lokalen Plugins ansehen.

Hierzu haben wir mehrere Durchführungsmöglichkeiten beschrieben, von denen wir Ihnen die Ausführung über SNMP (siehe hierzu Rezept 2.5, *Plugins unter Linux über SNMP ausführen* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*) empfehlen möchten. Die entsprechenden Rezepte behandeln eben aufgrund seiner Bedeutung auch gerade das Plugin check_apt, sind aber nicht auf dieses beschränkt. Wenn Sie keinen SNMP-Server auf der Zielmaschine betreiben möchten, können Sie aber auch NRPE (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*) oder SSH verwenden (siehe Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*). Das Rezept zur Ausführung über SSH beschreibt ebenfalls gerade dieses Plugin.

Siehe auch

- Rezept zu Befehlsdefinitionen: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Ausstattung von Plugins mit administrativen Rechten: Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten*
- Rezepte zur Remote-Ausführung von lokalen Plugins über SNMP und Auswertung der entsprechenden Ergebnisse: Rezept 2.5, *Plugins unter Linux über SNMP ausführen* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*
- Rezept zur Remote-Ausführung von lokalen Plugins über NRPE Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*
- Rezept zur Remote-Ausführung von lokalen Plugins über SSH: Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*

KAPITEL 6

Netzwerk-Dienste mit Standard-Plugins überwachen

In diesem Kapitel werden wir Standard-Plugins zur Überwachung von Diensten beschreiben. Im Gegensatz zu den im vorhergehenden Kapitel beschriebenen Plugins müssen diese normalerweise nicht auf der jeweiligen Maschine installiert werden. Sie werden auf dem Monitoring-Server ausgeführt und fragen den jeweiligen Dienst über das Netzwerk ab. Beachten sollten Sie hier, dass die entsprechenden Dienste dazu vom Monitoring-Server aus erreichbar sein müssen (siehe auch Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*).

6.1 Erreichbarkeit überwachen mit Ping (check_ping)

Problem

Sie möchten die prinzipielle Erreichbarkeit einer Maschine über das Netzwerk mit `ping` überwachen.

Lösung

Für den möglichst unkomplizierten Test einer Netzwerkverbindung ist das Plugin `check_ping` gedacht und geeignet. Das Plugin `check_ping` ist Teil der Standard-Plugins. Es nutzt das Internet Control Message Protocol (ICMP), um einzelne Pakete an ein Gerät zu senden und dessen Antworten auszuwerten. Wir zeigen Ihnen im Folgenden, wie Sie das Plugin `check_ping` nutzen können.

`check_icmp`: Alternativ zu `check_ping` steht Ihnen auch das Plugin `check_icmp` zur Verfügung. Dieses ist in der Einrichtung etwas komplizierter, dafür aber in der Nutzung deutlich performanter. Die Unterschiede zwischen beiden Plugins erläutern wir im Diskussionsteil von Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)* genauer.

Befehlsdefinition und notwendige Optionen

Um `check_ping` zunächst als Befehl einzuführen, können Sie die folgende minimalistische Definition verwenden:

```
define command {
       command_name                    check_ping
       command_line                    $USER1$/check_ping -H $HOSTADDRESS$ -w \
                                       $ARG1$ -c $ARG2$ $ARG3$
}
```

Dabei werden der Pfad und die Pflichtparameter sowie ein Platzhalter für weitere Optionen über Makros (siehe dazu Rezept zu Befehlsdefinitionen und Makros Rezept 4.3, *Befehle hinzufügen (command)*) gesetzt. Die erforderlichen Optionen haben die folgende Bedeutung:

- `-H`

  Angabe der Adresse, unter der das Gerät zu erreichen ist. Typischerweise können Sie diese über das Makro `$HOSTADDRESS$` dynamisch zur Laufzeit einsetzen lassen.

- `-w` und `-c`

  Angaben der Schwellenwerte für die Zustände `WARNING` und `CRITICAL`. Bei `check_ping` sind dies jeweils ein Paar aus durchschnittlicher Antwortzeit in Millisekunden und prozentualem Paketverlust, also etwa 100.00,20% für 100ms Antwortzeit und 20% Paketverlust .

Weitere Optionen

Neben den erforderlichen Optionen können Sie die folgenden, optionalen Schalter verwenden:

- `-4` oder `-6`

  Legen Sie hiermit manuell fest, welche Version des Internetprotokolls genutzt werden soll (Vorgabe ist Version 4).

- `-p`

  Mit diesem Schalter bestimmten Sie, wie viele Pakete das Plugin senden soll (der Vorgabewert ist 5 Pakete).

- `-t`

  Legen Sie über den Timeout fest, wie viele Sekunden das Plugin auf eine Antwort warten soll, bevor ein Fehler angenommen wird (die Vorgabe ist hier 10 Sekunden).

Dienstdefinition

Als passende Dienstdefinition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) zur eben angeführten Befehlsdefinition können Sie dann beispielsweise die folgende verwenden:

```
define service{
       service_description                    Ping
       use                                    template-srv
       check_command                          check_ping!100.0,20%!500.0,60%
       hostgroup_name                         hg-all
}
```

Damit definieren Sie den voranstehenden Befehl auf Basis einer Vorlage `template-srv` als Dienst unter dem Namen `Ping` für alle Mitglieder der Maschinengruppe `hg-all` mit Status WARNUNG beziehungsweise CRITICAL bei durchschnittlichen Antwortzeiten von über 100 beziehungsweise 500 Millisekunden oder Paketverlust von mehr als 20% beziehungsweise 60%.

Test, Rückgabewerte und Graphen

Sie können die Funktion und Rückgabe des Befehls auf der Kommandozeile testen. Für die Maschine `localhost` können Sie den Befehl mit den oben vorgestellten Parametern wie folgt testen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ping -H localhost -w 100,20% -c 500,60%
PING OK - Packet loss = 0%, RTA = 0.03 ms|rta=0.025000ms;100.000000;500.000000;0.000000
pl=0%;
     20;60;0
```

ICMP Echo Request unter Windows aktivieren: Auf Windows-Systemen sind `ICMP Echo- Requests` zunächst deaktiviert. Mit folgendem Kommandozeilenbefehl können Sie diese explizit zulassen:

```
C:\Windows\system32>netsh advfirewall firewall add rule name="ICMP Allow
incoming V4 echo request" protocol=icmpv4:8,any dir=in action=allow
Ok.
```

Anhand der Pipe »|« in der Rückgabezeile können Sie erkennen, dass das Plugin auch Performancedaten zurückgibt. Entsprechend können Sie sich diese auch graphisch darstellen lassen. Wir zeigen Ihnen in Abbildung 6-1 einen Beispielgraphen aus PNP4Nagios für die mittlere Antwortzeit:

Beachten Sie bitte, dass die hier abgebildeten Graphen auf einem Prüfintervall von einer Minute basieren. Wenn Sie größere Intervalle verwenden, werden Ihre Graphen eventuell entsprechend gröber aufgelöst.

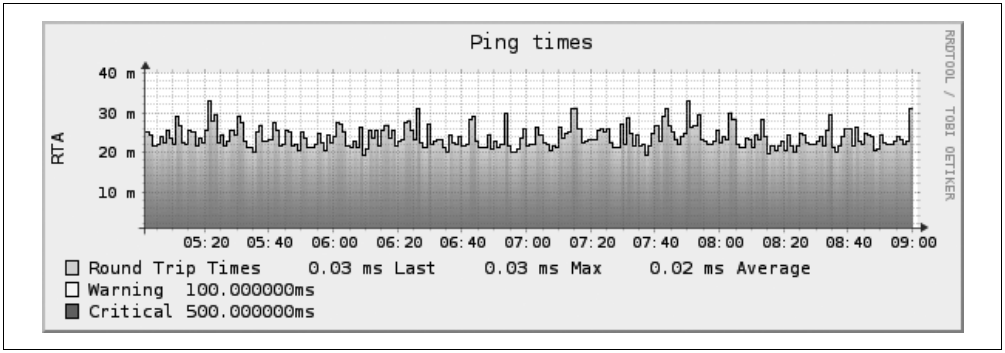


Abbildung 6-1: PNP-Graph Ping / Mittlere Antwortzeit

Diskussion

Die Erreichbarkeit über ICMP zu testen ist ein minimalistischer Ansatz – und gerade deshalb empfehlenswert, da Sie unter Umständen auf diese Weise noch Messwerte erhalten, wo komplexere Abfragen bereits scheitern. Nutzen Sie diese Pings deshalb, wie bereits bei der Installation vorgesehen, als sogenannten `host-check` für die Verfügbarkeitsprüfungen (siehe Rezept 4.1, *Maschinen einbinden (host)*). Die einzige Voraussetzung ist die, dass ICMP-Verbindungen zum zu überwachenden Gerät funktionieren müssen. Achten Sie darauf, dass ein eventuell vorhandener Paketfilter beziehungsweise eine Firewall ICMP-Pakete des Typs `echo request` nach außen und ICMP-Pakete des Typs `echo response` nach innen zulässt.

IPv4/6: Derzeit hat das Plugin `check_ping` unter Umständen ein Problem, wenn Sie die Maschine über Ihren Namen spezifizieren und dieser bei der Auflösung sowohl eine IPv4- als auch eine IPv6-Adresse zurück gibt. Es liefert dann den Status `CRITICAL` mit der Meldung `Network unreachable`. Sie können dieses Problem dann umgehen, indem Sie mit der Option `-4` die Nutzung von IPv4 erzwingen.

Beachten Sie bitte auch, dass Sie alternativ das ebenfalls im Standardpaket der Plugins enthaltene `check_icmp` verwenden können. Dieses ist etwas komplizierter bezüglich des Einrichtens, belastet den Monitoring-Server allerdings auch deutlich weniger (siehe dazu Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)*).

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Einbindung einer Maschine: Rezept 4.1, *Maschinen einbinden (host)*
- Rezept zur Überwachung der Erreichbarkeit mit ICMP: Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)*

6.2 Erreichbarkeit überwachen mit ICMP (check_icmp)

Problem

Sie möchten von einer Maschine über das Netzwerk die prinzipielle Erreichbarkeit über ICMP überwachen.

Lösung

Für einen möglichst einfachen Test der Erreichbarkeit über das Netzwerk ist das Internet Control Message Protocol (ICMP) prädestiniert. Dieses können Sie entweder über das Plugin `check_ping` (siehe Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*) oder direkter über `check_icmp` nutzen. Wir zeigen Ihnen im Folgenden, wie Sie `check_icmp` einbinden und verwenden. In der Diskussion werden wir die Unterschiede zum Einsatz von `check_ping` aufzeigen.

Ausführung mit administrativen Rechten: Das Plugin `check_icmp` benötigt bei der Ausführung administrative Rechte. Diese sollten bei der Installation entsprechend gesetzt worden sein (siehe Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten*).

Befehlsdefinition und notwendige Optionen

Das Plugin `check_icmp` ist in den Standard-Plugins enthalten. Um es zunächst als Befehl einzubinden, können Sie die folgende, minimalistische Definition in einer Ihrer Konfigurationsdateien verwenden:

```
define command {
       command_name                    check_icmp
       command_line                    $USER1$/check_icmp -H $HOSTADDRESS$ \
                                       $ARG1$
}
```

Dabei wird zur Laufzeit als einziger Pflichtparameter `-H` für die Zieladresse über das Makro `$HOSTADDRESS$` gesetzt (zu Befehlsdefinitionen und Makros siehe Rezept 4.3, *Befehle hinzufügen (command)*). Das Makro `$ARG1$` als Platzhalter erlaubt Ihnen gegebenenfalls, über die Dienstdefinition (siehe im Nachfolgendem) zusätzliche Optionen anzugeben.

Weitere Optionen

Weitere Optionen die Sie verwenden können sind unter anderem:

- `-w` und `-c`
  Angaben der Schwellenwerte für die Status `WARNING` und `CRITICAL`. Dies sind hier jeweils Wertepaare bestehend aus Angaben zu den Schwellenwerten für die durchschnittliche Antwortzeit und den Paketverlust im Format `100.00,40%` für 100 Millisekunden und 40 Prozent. Das Plugin verwendet als Vorgabe `200.00,40%` für `WARNING` und `500.00,80%` für `CRITICAL`.

- `-n`

  Mit diesem Parameter geben Sie die Anzahl der zu sendenden Pakete an (Vorgabewert: 5).

- `-t`

  Über diesen Timeout können Sie festlegen, wie viele Sekunden das Plugin auf auf eine Antwort wartet, bevor es von einem Paketverlust ausgeht (Vorgabwert: 10 Sekunden).

Für eine vollständige Übersicht aller möglichen Optionen rufen Sie das Plugin bitte mit der Option `--help` auf.

Dienstdefinition

Als Dienstdefinition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) zur voranstehenden Befehlsdefinition können Sie die folgende verwenden:

```
define service{
       service_description                    ICMP
       use                                    template-srv
       check_command                          check_icmp
       hostgroup_name                         hg-all
}
```

Mit dieser Definition bewirken Sie, dass das Plugin unter dem Namen `ICMP` für alle Maschinen der Gruppe `hg-all` durchgeführt wird. Die für den Service zu nutzende Vorlage (siehe Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*) heißt hier `template-srv`. Diesen Namen müssen Sie gegebenenfalls an Ihre Installation anpassen. Dabei werden die oben vorgestellten Vorgabewerte genutzt.

Test, Rückgabewerte und Graphen

Testen Sie vor der Einbindung von der Kommandozeile aus, dass das Plugin funktioniert. Um einen Testlauf auf der lokalen Maschine durchzuführen, verwenden Sie nachfolgende Syntax:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_icmp -H localhost
OK - localhost: rta 0.010ms, lost 0%|rta=0.010ms;200.000;500.000;0; pl=0%;40;80;;
rtmax=0.025ms;;;; rtmin=0.006ms;;;;
```

ICMP Echo-Request unter Windows aktivieren: Auf Windows-Systemen sind `ICMP Echo-Requests` zunächst deaktiviert. Mit folgendem Kommandozeilenbefehl können Sie diese explizit zulassen:

```
C:\Windows\system32>netsh advfirewall firewall add rule name="ICMP Allow
incoming V4 echo request" protocol=icmpv4:8,any dir=in action=allow
Ok.
```

An dem Pipe-Symbol »|« in der Rückgabe des Plugins können Sie erkennen, dass Performancedaten zurückgegeben werden. Entsprechend können Sie sich von den Ergebnissen gegebenenfalls Graphen zeichnen lassen. Die hier dargestellten Graphen stammen aus einem PNP4Nagios, das getrennte Graphen für die Antwortzeiten (siehe Abbildung 6-2) und Paketverluste vorsieht (siehe Abbildung 6-3). Die gezeigten Graphen basieren dabei auf einem Prüfintervall von einer Minute. Wenn Sie größere Intervalle verwenden, werden Ihre Graphen gegebenenfalls weniger fein aufgelöst.

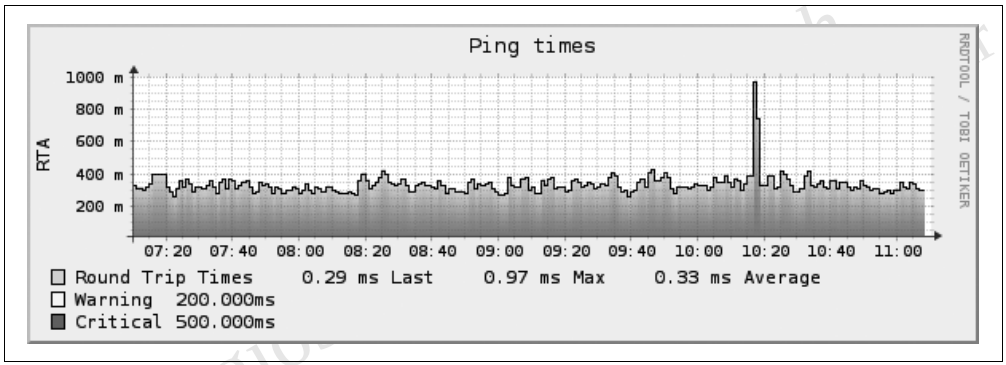


Abbildung 6-2: PNP4Nagios Graph: ICMP / Durchschnittliche Antwortzeit

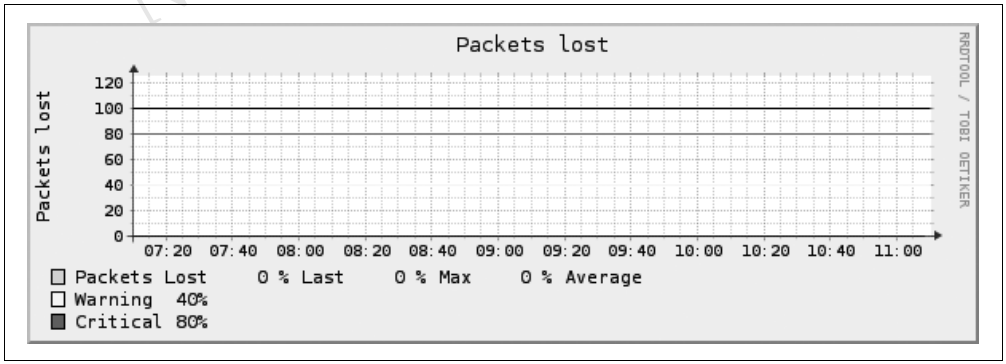


Abbildung 6-3: PNP4Nagios-Graph: ICMP / Paketverlust

Diskussion

Die prinzipielle Nützlichkeit von ICMP zum schnellen Testen der Erreichbarkeit von Maschinen haben wir Ihnen in der Diskussion von `check_ping` (siehe Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*) dargestellt. Beide Plugins verwenden das gleiche Protokoll, weshalb wir Ihnen die Unterschiede zwischen diesen Plugins aufzeigen möchten.

Zum einen ist `check_icmp` deutlich schneller in der Ausführung, da die Pakete parallel abgesetzt werden und nicht sequentiell wie bei `check_ping`. Deshalb sollten Sie die Verwendung von `check_icmp` insbesondere bei vielen Maschinen und/oder kleinem Prüfintervall in Betracht ziehen. Zum anderen verbraucht `check_icmp` weniger Systemresourcen da es nicht vom lokal installierten `ping` Kommando abhängig ist, dessen Ausgabe bei jedem Aufruf ausgewertet werden muss.

Ein weiterer wesentlicher Unterschied zu `check_ping` besteht darin, dass das Plugin `check_icmp` zur Ausführung administrative Rechte benötigt, da für die Erstellung eines ICMP-Packetes ein direkter Zugriff auf die Netzwerkkarte über ein sogenanntes RAW-Socket erforderlich ist. Dies ermöglicht auch das direkte Mitschneiden von Netzwerkverkehr und es lassen sich außerdem beliebige Pakete (inklusive aller Header-Optionen) erstellen, weshalb diese Funktionalität eben administrativen Benutzern vorbehalten ist.

Siehe auch

- Rezept zur Ausführung von Plugins mit administrativen Rechten: Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten*
- Rezept zur Überwachung der Erreichbarkeit mit Ping: Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Vereinfachung der Konfiguration mit Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*

6.3 Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)

Problem

Sie möchten einen beliebigen Internet-Dienst über seinen sogenannten Serviceport überwachen. Ihr Dienst ist über die Transportprotokolle TCP (Transport Control Protocol) und/oder UDP (User Datagram Protocol) erreichbar.

Lösung

Im Standardpaket der Plugins ist hierfür `check_tcp` enthalten, welches als symbolischer Link auch als `check_udp` (und in weiteren Variationen) zur Verfügung steht. Wir zeigen Ihnen, wie Sie mit diesen Plugins beliebige über das IP-Protokoll erreichbare Dienste prüfen können. Bei den folgenden, ebenfalls im Standardpaket enthaltenen Plugins handelt es sich um spezifisch angepasste Abwandlungen von `check_tcp`, die wir Ihnen gegebenenfalls in eigenen Rezepten detailliert vorstellen:

- E-Mail Dienste: `check_smtp`, `check_ssmtp`, `check_imap`, `check_simap`, `check_pop`, `check_spop` (siehe Rezept 6.14, *Überwachung der E-Mail-Auslieferungsdienste (check_pop, check_imap)* und Rezept 6.13, *Überwachung des E-Mail-Versanddienstes (check_smtp)*)
- File-Transfer-Protocol (FTP): `check_ftp` (siehe Rezept 6.10, *Überwachung eines FTP-Servers (check_ftp)*)
- Extensible Messaging and Presence Protocol (XMPP, auch als Jabber bekannt): `check_jabber` (siehe Rezept 6.4, *Überwachung eines Jabber-Servers (check_jabber)*)
- Network News Transfer Protocol (NNTP): `check_nntp` und `check_nntps`
- CLAMD-Protokol (Antivirus Scanner): `check_clamd`

Für die entsprechenden Dienste sollten Sie entsprechend diese spezifisch angepassten Versionen in Betracht ziehen, bevor Sie mit dem generischen Plugin `check_tcp` arbeiten.

Befehlsdefinition und notwendige Optionen

Das Plugin können Sie wie folgt als Befehl einrichten:

```
# 'check_tcp' command definition
define command {
       command_name                    check_tcp
       command_line                    $USER1$/check_tcp -H $HOSTADDRESS$ \
                                       -p $ARG1$ $ARG2$
```

Für Prüfungen auf Basis von TCP sind dabei alle Pflichtparameter vorgesehen. Die Variation als `check_udp` benötigt mehr Parameter, die in der folgenden, erweiterten Definition enthalten sind. Die zusätzlich notwendigen Parameter erläutern wir Ihnen im Anschluss:

```
# 'check_udp' command definition
define command {
       command_name                    check_udp
       command_line                    $USER1$/check_udp -H $HOSTADDRESS$ \
                                       -p $ARG1$ -s $ARG2$ -e $ARG3$ $ARG4
}
```

Der Pfad zum Plugin wird jeweils durch das Makro `$USER1$` gesetzt. Die Pflichtoptionen für beide Varianten sind folgende:

- `-H, --hostname=Adresse`

  Diese Option dient der Angabe der Netzwerk-Adresse der zu prüfenden Maschine. Standardmäßig wird diese Option zur Laufzeit über das Makro (siehe in Rezept 4.3, *Befehle hinzufügen (command)*) `$HOSTADDRESS$` gesetzt.

- `-p, --port=Port`

  Geben Sie hier die Portnummer an, unter der die Prüfung durchgeführt werden soll (Ganzzahl im Bereich 1-65535). Diese wird hier durch das Makro `$ARG1$` zur Laufzeit aus der Dienst-Definition gesetzt.

Für Prüfungen auf Basis von UDP sind des Weiteren die folgenden Optionen erforderlich, die bei der TCP-Variante dieses Plugins nur optional sind:

- `-s, --send="String"`

  Spezifizieren Sie hier die an den Server zu sendende Anfrage als Zeichenkette. In voranstehender Definition wird dieser Wert über das Makro `$ARG2$` gesetzt.

- `-e, expect="String"`

  Geben Sie hier eine oder mehrere Zeichenketten an, die als Antwort vom Server erwartet werden soll. In obiger Definition wird dieser Wert über das Makro `$ARG3$` gesetzt.

Fehlersuche: Zum Testen und zur Fehlersuche bei der Konfiguration mit den Optionen `-s und -e` empfehlen wir Ihnen den Einsatz der Option `-v` (verbose), um eine möglichst ausführliche Ausgabe zu erhalten.

Dabei sieht das Plugin eine möglichst einfache Verbindung zu dem Server vor. Bei TCP reicht dazu der Verbindungsaufbau (der sogenannte TCP-Handshake), während bei UDP das Senden einer Anfrage und Empfangen einer Antwort notwendig wird, da es hier keinen geregelten Verbindungsaufbau gibt. Die Status werden dabei bei TCP nur anhand der Antwortzeiten gesetzt. Bei UDP findet ein Vergleich der zurückgegebenen mit der zu erwartenden Zeichenkette statt. Bei TCP ist dieser Vergleich ebenfalls möglich, allerdings nicht erforderlich. Wie die Antworten gegebenenfalls ausgewertet und Status gesetzt werden, können Sie auf Wunsch mit den im Folgenden erläuterten Optionen genauer festlegen.

Weitere Optionen

Die folgenden Optionen ermöglichen Ihnen, das Verhalten des Plugins zu beeinflussen:

- `-4 oder -6`

  Mit dieser Optionen können Sie zwischen Version 4 und 6 des Internet Protocol wählen (Vorgabe ist derzeit 4).

- `-r, --refuse=ok|warn|crit`

  Mit dieser Option können Sie festlegen, wie das Plugin auf abgelehnte Verbindungen reagieren soll. Mögliche Optionen sind `ok`, `warn` und `crit` für den jeweiligen Status.

- `-M, --mismatch=ok|warn|crit`

  Welchen Status das Plugin zurückgibt, wenn der als erwartet angegebene String (siehe oben die Option `-e`) nicht mit dem erhaltenen übereinstimmt, legen Sie mit dieser Option fest. Mögliche Werte sind `ok`, `warn` und `crit` für den jeweiligen Status.

- `-w, --warning=Gleitkommazahl und -c, --critical=Gleitkommazahl`

  Mit dieser Option geben Sie die Anzahl der Sekunden an, die das Plugin zwischen dem Senden und Empfangen warten soll, bevor der Status auf `WARNING` (bei `-w`) beziehungsweise `CRITICAL` (bei `-c`) gesetzt wird.

- `-t, --time=ganze Zahl`

  Über diesen Parameter spezifizieren Sie den Timeout in Sekunden (Vorgabewert 10), nach dem das Plugin aufhört, auf eine Antwort zu warten und einen Fehler zurückgibt.

- `-d, --delay=ganze Zahl`

  Dieser Wert bestimmt die Anzahl von Sekunden zwischen dem Senden der definierten Zeichenkette und der Abfrage der Antwort. Nutzen Sie diese Option, wenn Sie bei dem abgefragten Server eine Reaktionszeit größer als eine Sekunde erwarten.

- `-m, --maxbytes=ganze Zahl`

  Wenn die hier definierte Anzahl von Bytes empfangen wurde, wird die Verbindung automatisch beendet.

- `-S, --ssl`

  Mit dieser Option aktivieren Sie die Verwendung von Secure Sockets Layer (SSL) für die Verschlüsselung der Verbindung.

- `-D, --certificate=ganze Zahl[,ganze Zahl]`

  Geben Sie bei der Verwendung von SSL hier die Anzahl von Tagen an, die das verwendete Zertifikat noch gültig sein muss, damit der Test erfolgreich ist. Der erste Wert ist die Warnschwelle für den Status `WARNING`, der zweite Wert die für den Status `CRITICAL`.

- `-q, --quit=String`

  Falls erforderlich können Sie hier eine Zeichenkette angeben, die abschließend an den Server gesendet wird, um die Verbindung zu beenden.

Die hier dargestellten Optionen sind dabei nur eine Auswahl. Für eine vollständige Liste rufen Sie das Plugin bitte mit der Option `--help` auf.

Dienstdefinition

Hier stellen wir Ihnen zwei minimale Dienstdefinitionen für den Einsatz mit TCP beziehungsweise UDP vor. Sie können einen Mumble-Server (Murmur genannt) überwachen, indem Sie mit folgender Dienstdefinition testen, ob auf Port `64738/tcp` ein Service angeboten wird:

```
define service{
       service_description                    murmur-tcp
       use                                    template-service-generic
       check_command                          check_tcp!64738
       host_name                              MUMBLE-Server
}
```

Für einen Test eines UDP-Ports müssen Daten versendet werden, die dem zu testenden Protokoll gemäß eine sinnvolle Nachricht enthalten muss. Sie können mit folgender Dienstdefinition einen NTP-Server überwachen. Wesentlich ist in diesem Beispiel, dass die an den NTP-Server gesendete Zeichenkette mit einem [beginnt und insgesamt 50Byte umfasst:

```
define service{
       service_description                    ntp
       use                                    template-service-generic
       check_command                          check_udp!123!-s "[.123456789abcdefghij\
                                              klmnopqrstvwxyz \0123456789.]" -e ""
       host_name                              Time-Server
}
```

Dies soll allerdings nur ein Beispiel sein. Im Falle von NTP empfehlen wir Ihnen, auf die ebenfalls in den Nagiosplugins enthaltenen Plugins check_ntp beziehungsweise die aktuelleren Plugins check_ntp_time und check_ntp_peer zurückzugreifen. Auf diese konnten wir in diesem Buch aus Platzgründen allerdings leider nicht weiter eingehen.

Test, Rückgabewerte und Graphen

Im Folgenden gehen wir von der voranstehenden Dienstdefinition für den Test des TCP-Serviceports eines Mumble-Servers aus. Die manuelle Ausführung des Checks ergibt beispielsweise:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_tcp -H 10.85.58.40 -p 64738
TCP OK - 0.000 second response time on port 64738|time=0.000338s;;;0.000000;10.000000
```

Da das Plugin die Latenzzeit als Performancedaten zurückgibt, erhalten Sie automatisch einen Graphen in PNP4Nagios. Abbildung 6-4 zeigt Ihnen den entsprechenden Graphen, der auf Basis eines Zeitraums von 24 Stunden erstellt wurde.

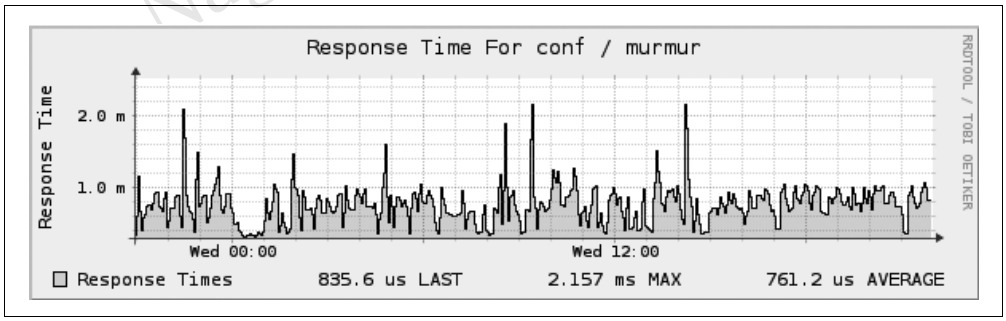


Abbildung 6-4: PNP4Nagios-Graph der gemessenen Latenz für 24 Stunden

Die Ausgabe ändert sich im Fehlerfall in Abhängigkeit vom aufgetretenen Fehler. Bei einem Timeout sähe die Ausgabe folgendermaßen aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_tcp -H 10.85.58.40 -p 64738
CRITICAL - Plugin timed out after 10 seconds
```

Im dem Fall, dass der Service nicht läuft oder eine Firewall den Verkehr blockiert, würden Sie folgende Fehlermeldung erhalten. Der Rückgabestatus ist dann ebenfalls CRITICAL:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_tcp -H 10.85.58.40 -p 64738
Connection refused
```

Diskussion

Bei den generischen Plugins check_tcp und check_udp handelt es sich um sehr einfache Plugins, die sich allerdings sehr vielseitig verwenden lassen. Sie können diese Plugins einsetzen, wenn für einen zu testenden Dienst kein passendes Plugin verfügbar ist.

Wenn der zu testende Dienst das TCP-Transportprotokoll nutzt, können Sie testen, ob der spezifizierte Zielport aktiv ist. Damit wissen Sie allerdings zunächst nicht, ob es sich dabei auch um den gewünschten Dienst handelt. Hier müssten Sie die zu sendende Zeichenkette und eine gewünschte Zeichenkette, die in der Anwort des Servers enthalten sein soll, entsprechend anpassen. Im Falle von UDP müssen Sie in jedem Fall beide Zeichenkette angeben, da sonst gar keine Verbindung aufgebaut wird.

Wir haben Ihnen im Folgenden zwei weitere Beispiele für entsprechende Prüfungen zusammengestellt. Sie können mit check_tcp etwa einen Inter Relay Chat (IRC-Server) überwachen. Eine entsprechende Prüfung könnte dann wie folgt aussehen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_tcp -H 130.133.8.2 -p 6667 -E -s
"PASS *\nNICK ANick\nUSER AName 8 * :MeinName\r\n\n" -e "ANick" -d 1 -m 1000
TCP OK - 0.020 second response time on port 6667
                    [:fu-berlin.de 020 * :Please wait while we process your connection.
:fu-berlin.de 001 ANick :Welcome to the Internet Relay Network ANick!~AName@hcksp0.de
:fu-berlin.de 002 ANick :Your host is fu-berlin.de, running version 2.11.2p2
:fu-berlin.de 003 ANick :This server was created Wed Dec 8 2010 at 17:45:14 CET
:fu-berlin.de 004 ANick fu-berlin.de 2.11.2p2 aoOirw abeiIklmnoOpqrRstv
:fu-berlin.de 005 ANick RFC2812 PREFIX=(ov)@+ CHANTYPES=#&!+ MODES=3 CHANLIMIT=#&!+:21
NICKLEN=15 TOPICLEN=255 KICKLEN=255 MAXLIST=beIR:64 CHANNELLEN=50 IDCHAN=!:5
CHANMODES=beIR,k,l,imnpstaqr :are supported by this server
:fu-berlin.de 005 ANick PENALTY FNC EXCEPTS=e INVEX=I CASEMAPPING=ascii NETWORK=IRCnet
:are supported by this server
:fu-berlin.de 042 ANick 276BBLVMU :your unique ID
:fu-berlin.de 251 ANick :There are 63814 users and 6 services on 29 servers
:fu-berlin.de 252 ANick 99 :operators online
:fu-berlin.de 253 ANick 5 :unknown connections
:fu-berlin.de 254 ANick 34458 :channels formed
:fu-berlin.de 255 ANick :]|time=0.019811s;;;0.000000;10.000000
```

Abschließend noch ein Beispiel für das UDP-Protokoll. Mit folgendem Servicecheck können Sie einen DNS-Server überwachen. Sie können beispielsweise testen, ob eine bestimmte IP-Addresse wie gewünscht aufgelöst wird:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_udp -H 10.168.178.1 -p 53 -s
"XY www.oreilly.de" -e "XY" -v
Using service UDP
Port: 53
flags: 0x2
Send string: XY www.oreilly.de
server_expect_count: 1
 0: XY
received 17 bytes from host
#-raw-recv-------#
XY?tww
#-raw-recv-------#
looking for [XY] anywhere in [XY?tww]
found it
UDP OK
- 0.003 second response time on port 53 [XY?tww]|time=0.003363s;;;0.000000;10.000000
```

Allerdings empfehlen wir Ihnen hierzu, das Plugin `check_dns` (siehe Rezept 6.5, *Das DNS überwachen (check_dns)*) zu bevorzugen.

Siehe auch

- Überwachung eines E-Mail-Servers: Rezept 6.14, *Überwachung der E-Mail-Auslieferungsdienste (check_pop, check_imap)* und Rezept 6.13, *Überwachung des E-Mail-Versanddienstes (check_smtp)*
- Rezept zur Überwachung eines FTP-Servers: Rezept 6.10, *Überwachung eines FTP-Servers (check_ftp)*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zur Überwachung von DNS-Servern: Rezept 6.5, *Das DNS überwachen (check_dns)*
- Hilfeseiten zu dem Nagiosplugin `check_ntp_time`: *http://nagiosplugins.org/man/check_ntp_time*
- Hilfeseiten zu dem Nagiosplugin `check_ntp_peer`: *http://nagiosplugins.org/man/check_ntp_peer*

6.4 Überwachung eines Jabber-Servers (check_jabber)

Problem

Sie möchten einen Jabber-Server überwachen, also eine Maschine mit einem Chat-Server der das XMPP (eXtensible Messaging and Presence Protocol) verwendet.

Lösung

Für eine rudimentäre Überwachung ist in den Standardplugins `check_jabber` als Abwandlung von `check_tcp` (siehe Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*) enthalten. Wir zeigen Ihnen im Folgenden, wie Sie dieses Plugin einbinden und verwenden können.

Befehlsdefinition und notwendige Optionen

Um das Plugin in ein Monitoring-System mit Nagios/Icinga einzubinden, erstellen Sie zunächst eine entsprechende Befehlsdefinition unter Nutzung entsprechender Makros (siehe zu Befehlsdefinitionen und Makros Rezept 4.3, *Befehle hinzufügen (command)*). Sie können hierzu die folgende, minimale Definition verwenden:

```
define command {
       command_name                    check_jabber
       command_line                    $USER1$/check_jabber -H $HOSTADDRESS$ \
                                       $ARG1$
}
```

Der einzige vorgesehene Parameter ist auch der einzige Pflichtparameter:

- `-H / --hostname`

  Mit diesem Parameter geben Sie an, welche Maschine abgefragt werden soll. Sie können dabei entweder eine IP-Adresse oder einen voll qualifizierten Domain-Namen verwenden.

Weitere Optionen

Als weitere Optionen können Sie die bei `check_tcp` beschriebenen verwenden, da dieses Plugin ein symbolischer Link auf `check_tcp` ist.

Dienstdefinition

Um die voranstehende Befehlsdefinition für die (bereits definierte) Maschine `conf` unter Nutzung der Service-Vorlage `template-service-generic` (siehe Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*) einzubinden, können Sie die folgende Dienstdefinition verwenden (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*):

```
define service {
       service_description             Jabber
       use                             template-service-generic
       check_command                   check_jabber
       host_name                       conf
}
```

Die Namen für Vorlage und Zielmaschine müssen Sie dabei gegebenenfalls Ihrer Umgebung entsprechend anpassen.

Test, Rückgabewerte und Graphen

Die Durchführung und Rückgabewerte des Plugins können Sie auf der Kommandozeile wie folgt testen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_jabber -H 10.85.58.40
JABBER OK - 0.002 second response time on port 5222|time=0.001964s;;;0.000000;10.000000
```

Das Plugin gibt durch das Pipe-Symbol »|« separierte Performancedaten zurück, so dass ein Grapher die Daten verarbeiten kann. Wenn Sie PNP4Nagios nutzen, erhalten Sie deshalb automatisch einen Graphen wie in Abbildung 6-5 .

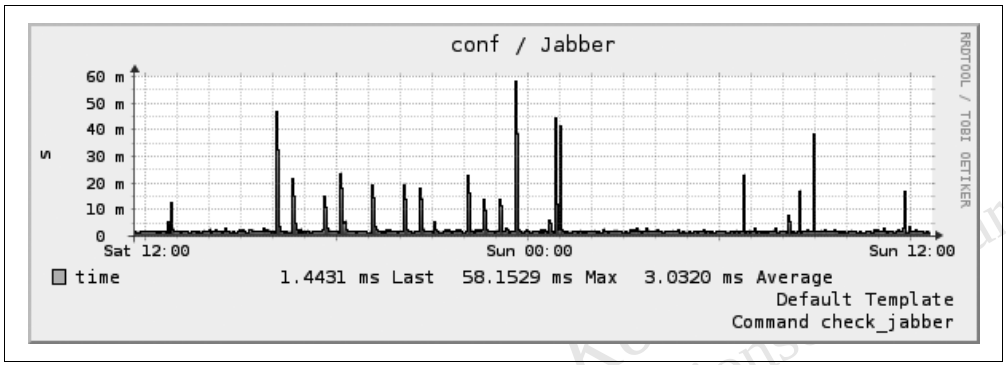


Abbildung 6-5: PNP4Nagios-Graph für check_jabber

Diskussion

Ein Jabber-Server nutzt verschiedene Dienste für unterschiedliche Formen der Kommunikation. Der durch dieses Plugin getestete Dienst auf Port TCP/5222 wird typischerweise für die Kommunikation zwischen Client und Servern genutzt

Server-zu-Server-Kommunikation

Wenn Sie den Server nicht nur lokal verwenden, wird er für die Kommunikation mit anderen Jabber-Servern darüber hinaus auf Port TCP/5269 lauschen. Diese Kommunikation überwacht das Plugin nicht. Sie können auch nicht einfach den entsprechenden Port hinzufügen, um diesen Dienst zu überwachen, da dieser über ein anderes Protokoll kommuniziert. Um auch die Verfügbarkeit dieses Ports zu überwachen, können Sie das Plugin `check_tcp` mit einem entsprechenden Parameter für den Port einsetzen (siehe Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*). Der folgende Aufruf dieses Plugins führt einen entsprechenden rudimentären Test durch:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_tcp -H 10.85.58.40 -p 5269
TCP OK - 0.001 second response time on port 5269|time=0.000784s;;;0.000000;10.000000
```

Dabei prüfen Sie dann allerdings nur, dass an diesem Port ein Serverprozess läuft, nicht aber, was für ein Service dahinter steckt. Wenn Sie testen möchten, ob anderen Servern eine Sitzung angeboten wird, können Sie dies mit einem erweiterten Test über optionale

Parameter zu `check_tcp` realisieren. Sie können dazu eine Sitzung anfordern und in der Antwort die Zeichenkette `id=` erwarten, um den Erfolg zu testen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_tcp -H 10.85.58.40 -p 5269 -s
"<stream:stream xmlns='jabber:server' xmlns:stream='http://etherx.jabber.org/streams'
to='10.85.58.40' version='1.0'>" -e "id="
TCP OK - 0.002 second response time on port 5269 [<?xml version='1.0'?>
<stream:stream xmlns:stream='http://etherx.jabber.org/streams' xmlns='jabber:server'
xmlns:db='jabber:server:dialback' id='123806705' version='1.0'>]|time=0.002065s;;;
0.000000;10.000000
```

Bei einer Einbindung in das Monitoring können Sie dabei zusätzlich den Parameter `-j` verwenden, um die Ausgabe kompakt zu halten. Das folgende Kommando zeigt dies anhand des voranstehenden Beispiels:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_tcp -H 10.85.58.40 -p 5269 -s
"<stream:stream xmlns='jabber:server' xmlns:stream='http://etherx.jabber.org/streams'
to='10.85.58.40' version='1.0'>" -e "id=" -j
TCP OK - 0.002 second response time on port 5269|time=0.002109s;;;0.000000;10.000000
```

Sofern Sie weitere Dienste über Jabber anbieten (wie etwa Konferenzräume), können Sie sich mit `check_tcp` entsprechende Prüfungen erstellen. Allerdings müssen Sie dazu gegebenenfalls die Protokolle kennen, um passende Abfragen erstellen zu können.

Webfrontend am Beispiel *ejabberd*

Wenn die Serverinstallation mit einem Web-Frontend zur Administration ausgestattet ist, wie etwa bei `ejabberd` auf Port 5280, können Sie dieses gegebenenfalls mit dem Plugin `check_http` überwachen (siehe Rezept 6.11, *Überwachung eines Webservers (check_http)*). Der folgende Aufruf zeigt Ihnen ein Beispiel für den Test der Admin-Seite eines ejabberd-Servers:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_http -H 10.85.58.40 -p 5280 -u /admin -a
'ejabberd-admin@10.85.58.40:PASSWD'
HTTP OK: HTTP/1.1 200 OK - 1723 bytes in 0.036 second response time
|time=0.036120s;;;0.000000 size=1723B;;;0
```

Da die Administrationsseite nur über eine Authentifizierung mittels `http-auth` möglich ist, müssen Sie dabei den Benutzernamen (hier `ejabberd-admin@10.85.58.40`) und das Passwort (hier: `PASSWD`) mit dem Parameter `-a` angeben.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Überwachung beliebiger TCP/UDP-Ports: Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*

- Rezept zur Vereinfachung der Konfiguration mit Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Rezept zur Überwachung eines Webservers: Rezept 6.11, *Überwachung eines Webservers (check_http)*

6.5 Das DNS überwachen (check_dns)

Problem

Sie möchten einen DNS-Server überwachen. Die Funktionalität des DNS (Domain Name System) soll durch einen Test geprüft werden, indem ein Domänenname erfolgreich in eine IP-Adresse aufgelöst wird.

Lösung

Im Paket der Standard- beziehungsweise Nagiosplugins ist zu diesem Zweck das Plugin `check_dns` enthalten, welches den Befehl `nslookup` verwendet. Das Plugin ermöglicht Ihnen, unterschiedliche Abfragen an das DNS abzusetzen und auszuwerten. Wir zeigen Ihnen im Folgenden, wie Sie dieses Plugin einbinden und verwenden können.

Befehlsdefinition und notwendige Optionen

Für eine Einbindung benötigen Sie zunächst eine Befehlsdefinition (siehe hierzu Rezept 4.3, *Befehle hinzufügen (command)*). Im DNS sind unterschiedliche Arten von Abfragen möglich. Bei der Standard-Abfrage werden textbasierte Domänen in nummerische Adressen aufgelöst. Häufig wird auch die umgekehrte Abfrage genutzt (der sogenannte Reverse-Lookup), bei dem entsprechend zu nummerischen Adressen Domäne-Namen gesucht werden. Wir stellen Ihnen zu diesen Abfragen jeweils Befehlsdefinitionen zur Verfügung:

```
define command {
       command_name                    check_dns
       command_line                    $USER1$/check_dns -H $HOSTNAME$ $ARG1$
}
define command {
       command_name                    check_dns_reverse
       command_line                    $USER1$/check_dns -H $HOSTADDRESS$ $ARG1$
}
```

Beachten Sie, dass einmal das Makro für den Namen des Gerätes (`$HOSTNAME$`) und einmal die Adresse des Gerätes (`$HOSTADDRESS$`) angegeben ist. Die einzige Pflichtoption, die Sie für dieses Plugin zur Verfügung stellen müssen, ist die in den Definitionen vorgesehene:

- `-H` / `--hostname`

  Mit dieser Option geben Sie den Namen oder die Adresse an, der beziehungsweise die aufgelöst werden soll. Achten Sie bei textbasierten Adressen darauf hier den vollständigen Namen, also den sogenannten fully qualified domain name (FQDN), zu verwenden.

Die Benennung als `hostname` ist bei dieser Option eigentlich etwas unglücklich, da es hier bei dem Plugin `check_dns` einen bedeutenden Unterschied macht, ob Sie nun den Hostnamen oder die IP-Adresse angeben. Bei den meisten anderen Plugins können Sie typischerweise statt des Namens alternativ auch die Adresse angeben, ohne dass die eigentliche Prüfung sich technisch unterscheidet.

Weitere Optionen

Neben dieser notwendigen Pflichtoption können Sie das Plugin mit den folgenden Optionen weiter an Ihre Bedürfnisse anpassen:

- `-s` / `--server`

  Wenn Sie gezielt einen bestimmten DNS-Server abfragen wollen, können Sie ihn mit dieser Option spezifizieren. Ansonsten wird das Plugin den in Ihrem System verwendeten Server (häufig über DHCP zugeteilt) verwenden.

- `-a` / `--expected-address`

  Mit dieser Option können Sie den erwarteten Rückgabewert angeben. Das Plugin wird dann einen Fehler melden, falls es einen davon abweichenden Wert erhält. Sie können dabei mehrere IP-Adressen oder Domänen-Namen in einer durch Kommata separierten Liste angeben. Beachten Sie, dass Namen dabei auf einen `.` enden müssen, den Sie also entsprechend hinzufügen müssen.

- `-A` / `--expect-authority`

  DNS-Server geben bei Anfragen zurück, ob Sie für den jeweiligen Namen zuständig sind oder nicht. Mit dieser Option weisen Sie das Plugin an, diese Zuständigkeit zu erwarten. Entsprechend wird Ihnen dann eine Fehlermeldung zurückgegeben, wenn diese nicht gegeben ist – selbst wenn die Auflösung dennoch funktioniert.

- `-w` / `--warning` und `-c` / `--critical`

  Neben den inhaltlichen Auswertungen können Sie das Plugin auch veranlassen, die Antwortzeiten auszuwerten. Geben Sie dazu entsprechende Schwellenwerte in Sekunden an. Sie können hierbei Nachkommastellen verwenden, also etwa `0.001`.

- `-t` / `--timeout`

  Neben den Schwellenwerten für die Antwortzeiten können Sie das Plugin auch veranlassen, die Anfrage nach einem vom Vorgabewert 10 Sekunden abweichenden Zeitraum abzubrechen. Geben Sie dazu mit diesem Parameter die Anzahl der Sekunden an.

Diese Optionen erlauben Ihnen eine weitgehende Kontrolle darüber, wie das Plugin Abfragen durchführt und auswertet. Wir gehen im Rahmen der Diskussion im Nachfolgenden noch geanuer auf entsprechende Prüfszenarien ein.

Dienstdefinition

Um die obige Befehlsdefinitionen zu verwenden, müssen Sie sie über eine Dienstdefinition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) mit entsprechenden Geräten verknüpfen. Um beide Definitionen für eine definierte Maschine `hcksp0.de` einzubinden, können Sie etwa die folgenden Definitionen verwenden:

```
define service {
       service_description                    DNS
       use                                    template-service
       check_command                          check_dns
       host_name                              hcksp0.de
}
define service {
       service_description                    DNS-Reverse
       use                                    template-service
       check_command                          check_dns_reverse
       host_name                              hcksp0.de
}
```

Sie prüfen damit, ob die in der Definition des Gerätes (siehe Rezept 4.1, *Maschinen einbinden (host)*) hinterlegten Daten (Name und Adresse) im DNS-System aufgelöst werden.

Test, Rückgabewerte und Graphen

Vor der Einbindung beziehungsweise wenn Sie Probleme mit der Einbindung haben, sollten Sie das Plugin von der Kommandozeile aus testen. Für die voranstehende Definition sind folgende Aufrufe äquivalent:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:/# /usr/local/icinga/libexec/check_dns -H monilog.de
DNS OK: 0.028 seconds response time. monilog.info returns 212.111.226.161|
time=0.027717s;;;0.000000
icinga@moni:/# /usr/local/icinga/libexec/check_dns -H 212.111.226.161
DNS OK: 0.030 seconds response time. 212.111.226.161 returns web1.smt-hybrid.de.
|time=0.029712s;;;0.000000
```

Das Plugin gibt dabei jeweils durch das Pipe-Symbol »|« separiert Performancedaten zu den Antwortzeiten zurück, so dass Sie bei Verwendung von PNP4Nagios automatisch Graphen zu diesen Prüfungen erhalten. Abbildung 6-6 zeigt Ihnen ein Beispiel für einen solchen Graphen.

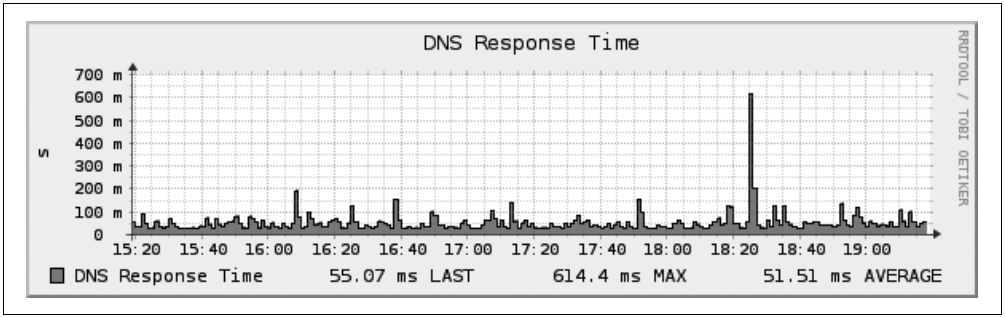


Abbildung 6-6: PNP4Nagios-Graph für DNS-Abfragezeiten

Diskussion

Die hier gezeigte Einbindung ist minimalistisch, da Sie lediglich überprüft, ob die Daten überhaupt aufgelöst werden. Zumindest bei einfachen Konfigurationen, bei denen ein Name auf genau eine Adresse aufgelöst wird, können Sie die Prüfung sehr einfach um eine Korrektheitsprüfung erweitern. Setzen Sie hierzu den Parameter `-a` ein, um auch zu prüfen, dass der korrekte Wert zurückgegeben wird:

```
define service {
       service_description                    DNS+
       use                                    template-service
       check_command                          check_dns!-a $HOSTADDRESS$
       host_name                              hcksp0.de
}
define service {
       service_description                    DNS-Reverse+
       use                                    template-service
       check_command                          check_dns_reverse!-a $HOSTNAME$.
       host_name                              hcksp0.de
}
```

Beachten Sie, wie hier jeweils das Gegenstück (Name beziehungsweise Adresse) für die Option `-a` verwendet wird. Durch diese weitergehende Prüfung können Sie Angriffe durch sogenanntes DNS-Spoofing erkennen. Bei diesem Angriff werden DNS-Anfragen auf einen manipulierten Server umgeleitet, der die Anfragen dann im Sinne des Angreifers auflöst. Wenn Geräte mehrere Namen und/oder Adressen haben, funktioniert diese Prüfung allerdings nicht ganz so einfach. Sie müssten die zusätzlichen Daten dann entweder manuell für jeden Test einzeln pflegen oder über benutzerdefinierte Variablen (siehe Rezept 4.10, *Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen*) hinterlegen und verwenden.

Zusätzlich zu diesen erweiterten Tests könnten Sie die Prüfungen auch noch mit der Option `-s` durchführen, um gezielt den gewünschten DNS-Server abzufragen. Auf diese Weise prüfen Sie dann ohne `-s` auf DNS-Angriffe aus Sicht Ihrer Netzteilnehmer und mit `-s` die Funktionsfähigkeit des eigenen DNS-Servers. Aufgrund der hohen Bedeutung des

DNS (typischerweise läuft nur wenig Kommunikation direkt über IP-Adressen) ist der Aufwand für derart erweiterte Prüfungen durchaus gerechtfertigt.

check_dig: Zusätzlich zu dem Plugin `check_dns` enthalten die Nagiosplugins ebenfalls das Plugin check_dig, welches das Unix-Kommando `dig` nutzt. Hier ist es zusätzlich möglich, auch eine Portnummer, den Typ des abzufragenden DNS-Eintrags oder auch beliebige andere Parameter an das Kommando `dig` zu übergeben.

Siehe auch

- Rezept zur Einbindung von Kommandos: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zur Einbindung von Diensten: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Einbindung von Geräten: Rezept 4.1, *Maschinen einbinden (host)*
- Rezept zur Nutzung von benutzerdefinierten Variablen: Rezept 4.10, *Erweiterung von Konfigurationsobjekten über benutzerdefinierte Variablen*

6.6 DHCP überwachen (check_dhcp)

Problem

Sie möchten einen DHCP-Server überwachen. Die Funktionalität von DHCP (Dynamic Host Configuration Protocol) soll über eine erfolgreiche Abfrage einer IP-Adresse getestet werden.

Lösung

Für diesen Zweck ist in den Standard-Plugins `check_dhcp` enthalten. Wir zeigen Ihnen im Folgenden, wie Sie dieses einbinden und nutzen können.

Ausführung mit administrativen Rechten: Das Plugin `check_dhcp` benötigt bei der Ausführung administrative Rechte. Diese sollten bereits bei der Installation entsprechend gesetzt worden sein. Sie können dies im Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten* nachlesen, um den betreffenden Mechanismus zu verstehen und gegebenenfalls anzupassen.

Befehlsdefinition und notwendige Optionen

Bei der Installation wurde für Sie typischerweise bereits eine Befehlsdefinition erstellt (siehe auch Rezept 4.3, *Befehle hinzufügen (command)*). Bei unserer Referenzinstallation war dies die folgende:

```
define command{
       command_name                    check_dhcp
       command_line                    $USER1$/check_dhcp $ARG1$
}
```

Dabei wird der Pfad zum Plugin über das Makro `$USER1$` gesetzt und sämtliche Parameter werden über das Makro `$ARG1$` übergeben.

Diese Definition weicht von den anderen Definitionen ab, da hier die zu prüfende Maschine gar nicht spezifiziert wird. Ganz ohne weitere Parameter prüft das Plugin dann die Funktion der automatischen Verteilung über die sogenannte Broadcast-Adresse, erkennt einen verfügbaren DHCP-Server also automatisch.

Weitere Optionen

Über die folgende Optionen können Sie die Funktionsweise des Plugins anpassen:

- `-t / --timeout`

  Über diese Option können Sie die Wartezeit in Sekunden angeben, nach der die Anfrage als erfolglos abgebrochen wird.

- `-s / --serverip`

  Spezifizieren Sie hier die Adresse Ihres DHCP-Servers, damit das Plugin prüfen kann, dass der gewünschte Server (und nicht nur irgendeiner) reagiert.

- `-i / --interface`

  Mit dieser Option können Sie die Netzwerkschnittstelle festlegen, über die die Anfrage versendet werden soll. Diese Option ist meist nur bei mehreren Schnittstellen erforderlich. Als Vorgabewert wird `eth0` verwendet.

- `-m / --mac`

  Mit dieser Option können Sie eine MAC-Adresse übergeben, die bei der Anfrage an den DHCP-Server verwendet werden soll. Im Normalfall wird das Plugin die MAC-Adresse der entsprechenden Netzwerkschnittstelle verwenden. Durch Angabe dieser Option können Sie stattdessen etwa eine nur für diesen Test eingerichtet Auflösung prüfen.

- `-r / --requestedip`

  Falls Ihr DHCP-Server so konfiguriert ist, dass er für bestimmte MAC-Adressen vorab definierte IP-Adressen anstatt zufälliger zurückgibt, können Sie hier die erwartete IP-Adresse angeben. Das Plugin wird dann testen, ob der Server auch die gewünschte Adresse zurückgegeben hat.

Dienstdefinition

Zur Einbindung der voranstehenden, minimalistischen Befehlsdefinition für eine bereits definierte Maschine können Sie die folgende Dienstdefinition (siehe auch Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) verwenden:

```
define service{
       service_description                     DHCP
       use                                     template-service-generic
       check_command                           check_dhcp
       host_name                               localhost
}
```

Dabei wird für die Maschine `localhost` ein Dienst mit dem Namen `DHCP` erstellt, der basierend auf der Vorlage `template-service-generic` die eben angeführte Befehlsdefinition referenziert.

Test, Rückgabewerte und Graphen

Die Funktion des Plugins können Sie von der Kommandozeile wie folgt testen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_dhcp
OK: Received 1 DHCPOFFER(s), max lease time = 3600 sec.
```

Der Rückgabewert des Plugins enthält dabei keine Performance-Daten, so dass Sie zu diesem Service zunächst keine Graphen zeichnen können.

Diskussion

Über die vorgeschlagene Standardeinbindung des Plugins können Sie auf einfache Weise testen, dass ein DHCP-Server vorhanden und in der Lage ist, eine IP-Adresse zu vergeben. Dabei wird normalerweise eine `DHCPDISCOVER`-Nachricht an die Broadcast-Adresse gesendet, die der Server mit einer `DHCPOFFER`-Nachricht beantwortet, die eine IP-Adresse als Zuweisung enthält. Nur wenn auf diese eine `DHCPREQUEST`-Nachricht folgte, die das Plugin aber gerade nicht sendet, würde die Zuweisung auf dem Server auch aktiv und von diesem mit einer `DHCPACKNOWLEDGE`-Nachricht beantwortet.

Timeouts: Der Ablauf einer DHCP-Abfrage kann länger dauern, als der Vorgabewert von einer Sekunde für den Timeout hier vorsieht. Wenn das Plugin bei Ihnen also Fehler meldet, obwohl Ihr Netzwerk zu funktionieren scheint, setzen Sie den Timeout für die Prüfung um einige Sekunden hoch, indem Sie den Parameter `-t` mit der Anzahl der zu wartenden Sekunden übergeben, also etwa `-t 3` für 3 Sekunden.

Im Folgenden stellen wir Ihnen ein paar mögliche Anpassungen für Prüfungen vor, mit denen Sie gezielt Ihre Umgebung testen können.

Spezifischen Server testen

In der Praxis können Probleme mit DHCP dadurch entstehen, dass es ungewollt mehrere DHCP-Server gibt, die miteinander konkurrieren. Da das Plugin ohne weitere Optionen jedoch nur das Vorhandensein irgendeines DHCP-Servers testet, würde es einen solchen Fehler gar nicht abfangen können. Geben Sie dem Plugin deshalb möglichst die Adresse Ihres DHCP-Servers an, damit ein solcher Fall in Ihrer Umgebung vom Plugin erkannt werden kann – als Abwandlung zur voranstehenden Ausführung also etwa so:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_dhcp -s 10.85.58.1
OK: Received 1 DHCPOFFER(s), 1 of 1 requested servers responded, max lease time = 3600
sec.
```

Beachten Sie die Veränderung im Rückgabewert: Das Plugin gibt jetzt explizit an, dass es der gewünschte Server war, der geantwortet hat.

Zuweisung statischer Adressen

Sofern Sie über Ihren DHCP-Server statische Adressen verteilen lassen, sollten Sie gegebenenfalls auch diese Zuweisung testen lassen, als Erweiterung des voranstehenden Beispiels also etwa so:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_dhcp -s 10.85.58.1 -m 00:16:36:fb:7a:f6
-r 10.85.58.207
OK: Received 1 DHCPOFFER(s), 1 of 1 requested servers responded, requested address
(10.85.58.207) was offered, max lease time = 3600 sec.
```

Beachten Sie erneut, wie sich der Rückgabewert ändert. Jetzt gibt das Plugin zusätzlich an, dass nicht irgendeine, sondern die gewünschte Adresse zurückgegeben wurde.

Prüfung mehrerer Teilnetze

Falls Ihre Umgebung aus unterschiedlichen (virtuellen) Teilnetzen besteht, in denen jeweils andere DHCP-Server-Konfigurationen aktiv sind, sollten Sie diese einzeln testen. Jedes Teilnetz, mit dem Ihr Monitoring-Server über ein Interface verbunden ist (siehe hierzu Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*), können Sie testen, indem Sie dem Plugin das entsprechende Interface angeben:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_dhcp -i eth0 -m 00:16:36:fb:7a:f6
OK: Received 1 DHCPOFFER(s), max lease time = 3600 sec.
```

Erstellen Sie entsprechend für jedes Subnetz eine Service-Definition mit Angabe des Interfaces und den für das jeweilige Netzwerk angepassten Prüfbedingungen.

Siehe auch

- Rezept zur Ausführung von Plugins mit administrativen Rechten: Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*

- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Planung der Netzwerkanbindung der Monitoring-Servers: Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*

6.7 Erstellung generischer SNMP-Abfragen (check_snmp)

Problem

Sie möchten einen bestimmten SNMP-Wert auslesen.

Lösung

Im Standard-Set der Plugins ist für diesen Zweck das generische check_snmp enthalten. Im Folgenden zeigen wir Ihnen, wie sie dieses einrichten und verwenden. Die Einrichtung entsprechender Server haben wir in dem Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* beziehungsweise Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten* beschrieben.

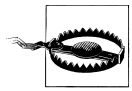
Befehlsdefinition und notwendige Optionen

Zur Einbindung können Sie die folgenden, generischen Definitionen verwenden, bei denen alle Pflichtparameter als Makros vorgesehen sind, unterschieden nach Version des genutzten SNMP-Servers (zu Befehlsdefinitionen und Makros siehe Rezept 4.3, *Befehle hinzufügen (command)*). Für SNMP Version 3 wäre die Definition folgende:

```
# check_snmp v3
define command {
       command_name                    check_snmp3
       command_line                    $USER1$/check_snmp -H $HOSTADDRESS$ \
                                       -P 3 -L authPriv -U "icinga" -a SHA \
                                       -A $USER12$ -x AES -X $USER12$ \
                                       -o $ARG1$ $ARG2$
}
```

Für Geräte, die lediglich SNMP Version 2c oder 1 unterstützen, können Sie eine der folgende Definitionen verwenden:

```
# check_snmp v1
define command {
       command_name                    check_snmp1
       command_line                    $USER1$/check_snmp -H $HOSTADDRESS$ \
                                       -P 1 -C $USER11$ -o $ARG1$ $ARG2$
}
# check_snmp v2c
define command {
       command_name                    check_snmp2
       command_line                    $USER1$/check_snmp -H $HOSTADDRESS$ \
                                       -P 2c -C $USER11$ -o $ARG1$ $ARG2$
}
```



SNMP-Version: Entsprechend sollten Sie bei jedem Gerät jeweils die bei diesem höchste unterstützte Version verwenden. Sofern Sie Version 1 oder 2c verwenden müssen, weil Ihre Geräte Version 3 nicht unterstützten, sollten Sie weiter auf eine Sicherung der Verbindung achten. Jeder, der hier den Verkehr belauschen kann, kann auch Ihre SNMP-Passwörter mitlesen.

Der Pfad zum Plugin wird dabei durch das Makro `$USER1$` gesetzt. Die hierbei für beide Versionen vorgesehenen Parameter sind folgende:

- `-H`
  Mit dieser Option geben Sie an, an welche Netzwerkadresse die Anfrage geschickt werden soll. Es ist vorgesehen, dass dieser Wert über das Makro `$HOSTADDRESS$` gesetzt wird.
- `-o`
  Hier geben Sie an, welche(r) Object Identifier (OID) abgefragt werden soll(en). Mehrere OIDs sind dabei durch Kommata oder Leerzeichen zu separieren. Wir haben hier vorgesehen, dass Sie den oder die OIDs in entsprechenden Service-Definitionen (siehe im Nachfolgenden) angeben und diese dann über das Makro `$ARG1$` übernommen werden.
- `-P`
  Mit diesem Parameter legen Sie die zu nutzende SNMP-Version fest. Wir haben neben der von uns empfohlenen Version 3 auch die Versionen 2c und 1 behandelt, da es bei älteren Geräten häufig vorkommt, dass Version 3 nicht unterstützt wird. Beachten Sie, dass sich die Authentifizierung in Abhängigkeit von der genutzten Version unterscheidet.

Authentifizierung für SNMP-Versionen 1 und 2

Bei den SNMP-Versionen 1 und 2 erfolgt die Authentifizierung über die sogenannte Community. Dabei handelt es sich um ein Passwort, dass unverschlüsselt übertragen wird. Die entsprechende Option ist folgende:

- `-C`
  Verwenden Sie diese Option, um eine Zeichenkette für die Verwendung mit SNMP Version 1 oder 2 als Community festzulegen. Im Beispiel haben wir vorgesehen, dass ein als Benutzermakro `$USER11$` (zu Benutzermakros siehe Rezept 4.3, *Befehle hinzufügen (command)*) hinterlegtes Passwort genutzt wird. Durch diese Vorgehensweise vereinfachen Sie nicht nur die Hinterlegung des Passworts, die sonst mehrfach erfolgen muss, und vermeiden gegebenenfalls Änderungen an diesem, sondern verhindern auch, dass es im Klartext in Protokolldateien auftaucht.

Authentifizierung für SNMP-Version 3

Bei Version 3 von SNMP setzt sich die Authentifizierung aus Benutzernamen und Passwort zusammen. Dabei sind für die Verschlüsselung verschiedene Protokolle möglich, weshalb das zu verwendende Protokoll ebenfalls angegeben werden muss. Die folgenden Optionen sind hierfür vorgesehen:

- `-L`

  Mit dieser Option geben Sie an, ob Sie Authentifizierung und/oder Verschlüsselung der Verbindung verwenden möchten. Die möglichen Werte und Ihre Funktionen sind

  – `noAuthNoPriv`

    für keine Authentifizierung und keine Verschlüsselung,

  – `authNoPriv`

    für Authentifizierung aber keine Verschlüsselung und

  – `authPriv`

    für Authentifizierung und Verschlüsselung.

Wir haben im voranstehenden Beispiel bei der Befehlsdefinition für SNMP Version 3 `authPriv` vorgesehen. Bei der Verwendung der Authentifizierung benötigen Sie des Weiteren die folgenden Optionen:

- `-a`

  Mit dieser Option legen Sie das Verschlüsselungsverfahren für die Authentifizierung fest. Im Beispiel haben wir hier SHA vorgesehen, welches als sicherer als das ebenfalls mögliche, ältere MD5 gilt.

- `-U`

  Hier geben Sie den Benutzernamen an, der entsprechend auf Ihrem Server eingerichtet sein muss (zur Servereinrichtung siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* beziehungsweise Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*). Im Beispiel verwenden wir hier *icinga*.

- `-A`

  Mit dieser Option müssen Sie das entsprechende Passwort für den Benutzer übergeben. Wir haben im Beispiel vorgesehen, dass dies über das Benutzer-Makro `$USER12$` erfolgt (zu Benutzermakros siehe Rezept 4.3, *Befehle hinzufügen (command)*). Durch diese Vorgehensweise vereinfachen Sie nicht nur die Hinterlegung des Passworts, die sonst mehrfach erfolgen muss, und vermeiden gegebenenfalls Änderungen an diesem, sondern verhindern auch, dass es im Klartext in Protokolldateien auftaucht.

Wenn Sie darüber hinaus die übertragenen Daten verschlüsseln möchten (Option `-L` mit `authPriv`), müssen Sie außerdem noch die folgenden Optionen nachtragen:

- `-x`

  Mit dieser Option legen Sie das auf dem Server eingestellte Verschlüsselungsverfahren fest. Die möglichen Werte sind hier DES und AES.

- `-X`

  Mit dieser Option übergeben Sie das Passwort für die Verschlüsselung, welches ebenfalls mit dem auf dem Server hinterlegten übereinstimmen muss. Wir haben im Beispiel wieder vorgesehen, dass dies über das Benutzer-Makro `$USER12$` erfolgt (zu Benutzermakros siehe Rezept 4.3, *Befehle hinzufügen (command)*).

Tests für Zahlenwerte

Wenn die von Ihnen abgefragte OID einen Zahlenwert darstellt, können Sie die folgenden Optionen verwenden, um das Ergebnis zu verarbeiten:

- `-w und -c`

  Mit diesen Optionen geben Sie die Werte-Bereiche an, die für die Status `WARNING` respektive `CRITICAL` genutzt werden sollen. Die Syntax der Spezifikation ist dabei zum Teil etwas gewöhnungsbedürftig (siehe hierzu *http://nagiosplug.sourceforge.net/developer-guidelines.html#THRESHOLDFORMAT*), aber in der Regel werden Sie vermutlich mit der Variante @start:end auskommen. Sie können die Bereiche auf Unendlich setzen, indem Sie für -~ bis 10 `"@~:10"` beziehungsweise für 10 bis ~ `"@10:"` verwenden. Sofern Sie mehrere Werte abfragen, können Sie entweder einen Bereich für alle oder eine durch Kommata separierte Liste von Bereichen mit je einem Bereich für jeden Wert verwenden.

- `--rate (--rate-multiplier)`

  Hiermit können Sie das Plugin anweisen, als Rückgabewert die Differenz zur vorherigen Messung zurückzugeben. Bei einem Zähler für übertragene Bytes erhalten Sie so (im Zusammenhang mit dem Abfrageintervall) etwa die Übertragungsrate. Das Plugin geht dabei zunächst von einem sekündlichen Abfrageintervall aus, das Sie aber mit der Option `--rate-multiplier` anpassen können (etwa `--rate-multiplier 60` für ein minütliches Abfrageintervall).

Tests für Textwerte

Wenn die von Ihnen abgefragte OID eine Zeichenkette zurückliefert, können Sie mit den folgenden Optionen die Verarbeitung beeinflussen (wenn Sie mehrere OIDs abfragen, gilt der Test für alle Ergebnisse):

- `-s`

  Mit dieser Option können Sie eine Zeichenkette angeben, die (exakt so) zurückgegeben werden muss, damit der Status `OK` zurückgegeben wird.

- `-r` beziehungsweise `-R`

  Mit diesen Optionen können Sie einen erweiterten regulären Ausdruck verwenden, der zutreffen muss, damit der Status `OK` zurückgegeben wird. Mit `-R` schalten Sie dabei den Test auf Groß-/Kleinschreibung ab.

- `--invert-search`

  Als Ergänzung zu einer der Optionen `-s`, `-r` oder `-R` können Sie hiermit das Ergebnis umkehren. Damit wird bei einem Fund beziehungsweise zutreffen dann der Status `CRITICAL` statt `OK` zurückgegeben.

Weitere Optionen

Als weitere Optionen möchten wir Ihnen die folgenden vorstellen (für eine vollständige Übersicht rufen Sie das Plugin bitte auf der Kommandozeile mit der Option `--help` auf):

- `-p`

  Mit dieser Option können Sie den Port festlegen, falls dieser bei Ihrem Server von dem Standardport 161 abweicht.

- `-e`

  Mit dieser Option können Sie veranlassen, dass das Plugin bei einem Fehlschlag zunächst erneute Versuche unternimmt. In aller Regel können Sie ein solches Verhalten aber besser über die entsprechenden Parameter in Ihrem Nagios/Icinga erzielen (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*), da Sie fehlgeschlagene Versuche dann nochvollziehen können.

- `-t`

  Mit dieser Option können Sie einen vom Standardwert 10 Sekunden abweichenden Timeout in Sekunden festlegen.

- `-d`

  Sofern der zur OID gehörende Rückgabewert ein Trennzeichen enthält, können Sie mit dieser Option statt des SNMP-Standards `=` ein abweichendes Trennzeichen angeben und verwenden.

- `-l`

  Mit dieser Option können Sie der Rückgabe eine Zeichenkette voranstellen, also beispielsweise mittels `-l "Benutzer"` aus 10 ein `Benutzer 10` machen.

- `-u`

  Mit dieser Option können Sie dem Rückgabewert eine Zeichenkette anhängen, also beispielsweise mittels `-u "kBit"` aus `10` ein `10 kBit` machen.

- `-D`

  Wenn Sie mehrere OIDs abfragen, können Sie mit dieser Option ein Trennzeichen für den Rückgabewert angeben. Damit können Sie etwa bei zwei Zahlenwerten 10 und 20 mit Hilfe von `-D " bis "` die Zeichenkette `10 bis 20` bilden lassen.

Dienstdefinition

Basierend auf der oben vorgeschlagenen Befehlsdefinition ist das folgende Beispiel eine mögliche Anwendung:

```
define service{
       service_description                    check_kvm
       use                                    template-srv-hcksp
       check_command                          check_snmp3!.1.3.6.1.2.1.1.1.0
       hostgroup_name                         hg-hcksp
}
```

Die hier abgefragte OID entspricht der Systembeschreibung (technisch `SNMPv2-MIB::sysDescr.0`). Da wir hier keine weiteren Optionen angegeben haben, würde dieser Service unabhängig vom Rückgabewert den Status `OK` aufweisen, solange die Abfrage nur erfolgreich ist.

Test, Rückgabewerte und Graphen

Den Befehl der voranstehenden Dienstdefinition können Sie zunächst auf der Kommandozeile testen. Dabei müssen Sie die sonst durch Makros ersetzten Optionen manuell angeben (hier für das erwähnte Beispiel auf Basis von SNMP Version 3):

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_snmp -P 3 -L authPriv -U "icinga" -a SHA
-A "PASSPHRASE" -x AES -X "PASSPHRASE" -H localhost -o .1.3.6.1.2.1.1.1.0
SNMP OK - "Linux moni 2.6.32-5-amd64 #1 SMP Mon Jan 16 16:22:28 UTC 2012 x86_64" |
```

An dem "|" können Sie erkennen, dass das Plugin prinzipiell Performancedaten zurückgibt. Bei Zeichenketten wie hier im Beispiel gibt es allerdings keine Performancedaten. Für ein Beispiel mit Zahlenwerten, das auch Quelle des folgenden Beispiel-Graphen Abbildung 6-7 ist, schauen Sie sich bitte Rezept 8.3, *Überwachung von Hypervisoren und virtuellen Maschinen* an.

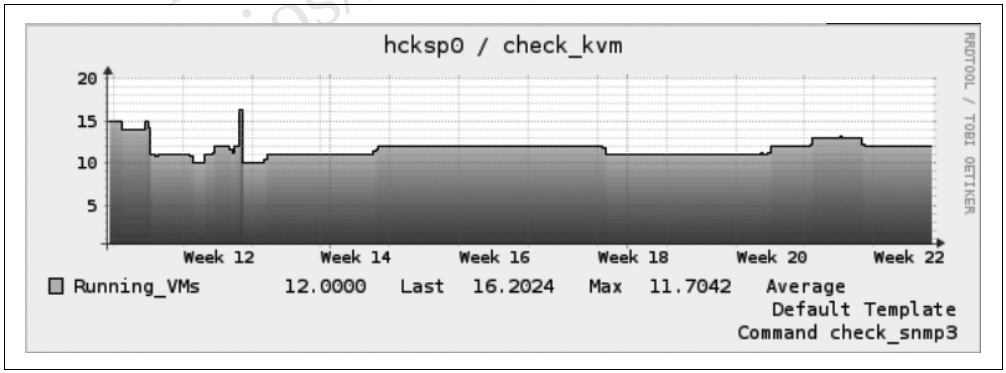


Abbildung 6-7: PNP4Nagios-Graph eines numerischen SNMP-Rückgabewertes

Bezeichner für Graphen: Wenn Sie im Graphen bei SNMP-Abfragen anstelle der nummerischen OID einen sprechenden Bezeichner haben möchten, können Sie diesen ohne spezifische Anpassungen an den Graphen-Definitionen über die Option `-l` angeben. Dieser dann vom Plugin zurückgegebene Bezeichner wird von den Standard-Graphen nämlich automatisch als Bezeichner erkannt und verwendet.

Wenn Sie sich entsprechende Graphen erstellen lassen, erhalten Sie dazu auch die statistischen Werte wie Durchschnitt und Maximum, die das Plugin hier nicht zurückgeben kann.

Diskussion

Mit `check_snmp` können Sie auf relativ einfache Weise beliebige Werte über SNMP abfragen. Die Vielfalt an Optionen ist dabei zunächst verwirrend, sorgt aber dafür, dass Sie im Zweifel kein gesondertes Plugin benötigen, sondern sich mit diesem Plugin behelfen können. Als Beispiel sei hier auf das Rezept für das Monitoring von virtuellen Maschinen verwiesen, aus dem auch der Graph stammt (siehe Rezept 8.3, *Überwachung von Hypervisoren und virtuellen Maschinen*). Dieses Rezept besteht für die zu überwachenden Maschine lediglich in einer Anpassung der SNMP-Konfiguration im Zusammenspiel mit der Nutzung dieses Plugins auf der Monitoring-Maschine.

Ein weiteres Beispiel, welches Ihnen vielleicht von Nutzen sein kann, ist die einfache Abfrage von in SNMP eingebundenen, benutzerdefinierten Kommandos. In Rezept 2.5, *Plugins unter Linux über SNMP ausführen* zeigen wir Ihnen am Beispiel von `check_apt` (siehe siehe Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*), wie Sie Plugins über SNMP einbinden können. Damit liegen Ihnen die Rückgabewerte der Plugins über SNMP vor. Für eine Einbindung in Nagios/Icinga müssen Sie diese jetzt noch abfragen. Als einfache Variante können Sie nun `check_snmp` verwenden, um die zurückgegebene Statuszeile nach einer geeigneten Zeichenketten zu durchsuchen. Die Statuszeile eines wie eben in Rezept 2.5, *Plugins unter Linux über SNMP ausführen* beschrieben als `check_apt` eingebundenen Plugins steht unter der OID `NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."check_apt"` bereit. Da `check_apt` beim Status `OK` eine Zeichenkette beginnend mit `APT OK:` zurückgibt, können Sie das Plugin wie folgt mit `check_snmp` abfragen:

```
icinga@moni:/usr/local/icinga/etc/moni-conf.d$ /usr/local/icinga/libexec/check_snmp
-P 3 -L authPriv -U icinga -a sha -A "PASSPHRASE" -x AES -X "PASSPHRASE" -H 127.0.0.1
-o .1.3.6.1.4.1.8072.1.3.2.3.1.1.9.99.104.101.99.107.95.97.112.116 -r "APT OK:.*"
SNMP OK - "APT OK: 0 packages available for upgrade (0 critical updates). " |
```

Einbinden können Sie diesen Test mit

```
define service{
       service_description                    check_snmp_apt
       use                                    template-srv
       check_command                          check_snmp3!\
.1.3.6.1.4.1.8072.1.3.2.3.1.1.9.99.104.101.99.107.95.97.112.116! -r "APT OK:.*"
       hostgroup_name                         snmpd
}
```

Passen Sie hierbei gegebenenfalls wieder Ihre Vorlage (hier `template-srv`) und die Maschinenzuweisung (hier über die Maschinengruppe `snmpd`) an.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Format der Schwellenwerte *http://nagiosplug.sourceforge.net/developer-guidelines.html#THRESHOLDFORMAT*
- Rezept zur Überwachung von Hypervisoren und virtuellen Maschinen: Rezept 8.3, *Überwachung von Hypervisoren und virtuellen Maschinen*
- Rezept zur Remote-Ausführung von lokalen Plugins über SNMP: Rezept 2.5, *Plugins unter Linux über SNMP ausführen*
- Rezept zur Überwachung von Aktualisierungen: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*

6.8 Schnittstellen über SNMP allgemein prüfen (check_ifstatus)

Problem

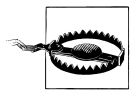
Sie möchten den Status der Netzwerkschnittstellen eines Gerätes überwachen, etwa um auf herausgerutschte Kabel zu prüfen.

Lösung

Im Paket der Standardplugins ist zu diesem Zweck `check_ifstatus` enthalten, das Ihnen eine entsprechende Prüfung über SNMP (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* beziehungsweise Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*) ermöglicht. Dabei wird der Status der Schnittstellen abgefragt und eine Übersicht zurückgegeben. Wir zeigen Ihnen im Folgenden, wie Sie dieses Plugin einsetzen und einbinden können.

Befehlsdefinition und notwendige Optionen

Um das Plugin `check_ifstatus` einzubinden, benötigen Sie zunächst eine Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*). Wir stellen Ihnen hier für die unterschiedlichen Versionen von SNMP Alternativen vor:



SNMP-Version: Sie sollten bei jedem Gerät die höchste unterstützte Version verwenden. Sofern Sie Version 1 oder 2c verwenden müssen, weil Ihre Geräte Version 3 nicht unterstützten (etwa Windows), müssen Sie damit rechnen, dass Angreifer nicht nur den Datenverkehr sondern auch die verwendeten Passwörter mitlesen können.

```
define command {
       command_name                    check_ifstatus_v1
       command_line                    $USER1$/check_ifstatus -H $HOSTADDRESS$ \
                                       -v 1 -C $USER11$ $ARG1$
}
```

```
define command {
       command_name                    check_ifstatus_v2
       command_line                    $USER1$/check_ifstatus -H $HOSTADDRESS$ \
                                       -v 2 -C $USER11$ $ARG1$
}

define command {
       command_name                    check_ifstatus_v3
       command_line                    $USER1$/check_ifstatus -H $HOSTADDRESS$ \
                                       -v 3 -L authPriv -U "icinga" -a SHA1 \
                                       -A $USER12$ -P AES -X $USER12$ $ARG1$
}
```

AES-Verschlüsselung: Das Plugin basiert auf Perl, für das eine eigene Bibliothek für die Verschlüsselung über AES vorhanden und auch notwendig ist. Wenn Sie eine Fehlermeldung wie

```
Required module Crypt/Rijndael.pm not found
```

erhalten, müssen Sie dieses noch installieren. Sie erreichen dies mit dem Debian-Paketmanager wie folgt:

```
root@moni:~# apt-get install libcrypt-rijndael-perl
```

Nach der Installation verschwindet die Fehlermeldung, und die Verschlüsselung funktioniert wie hier beschrieben.

Wählen Sie für sich eine der Definitionen danach aus, welche SNMP Version Ihre Geräte unterstützen. Als Pfad und Zieladresse nutzen die Definitionen wie üblich (siehe Rezept 4.3, *Befehle hinzufügen (command)*) die Makros `$USER1$` und `$HOSTADDRESS$`. Für SNMP Versionen 1 und 2c wird die Passphrase (der sogenannte Community-String) über das Makro $USER11$ gesetzt, während bei Version 3 das Makro `$USER12$` für die hinterlegte Passphrase zur Authentifizierung und gegebenenfalls Verschlüsselung zum Einsatz kommt (wir verwenden hier im Beispiel die gleiche Passphrase für die Authentifizierung und Verschlüsselung).

Die in den Definitionen vorgesehenen Pflichtparameter sind folgende:

- `-H` / `--hostname`

  Legen Sie mit diesem Parameter den Hostnamen oder IP-Adresse des zu überwachenden Systems fest.

- `-v` / `--snmp_version`

  Mit dieser Parameter spezifizieren Sie die zu nutzende SNMP-Version. Version 2 (für Version 2c) führt gegenüber 1 größere Zahlenräume ein, Version 3 führt gegenüber 2c eine erweiterte Benutzerverwaltung und Verschlüsselung ein.

In Abhängigkeit von der genutzten SNMP-Version müssen Sie dann weiter die jeweiligen Zugangsdaten setzen. Für Versionen 1 und 2 ist dies schlicht folgende:

- `-C` / `--community`

  Geben Sie hier das »community-string« genannte Passwort an.

Für Version 3 müssen Sie hingegen diese Optionen verwenden:

- `-L` / `--seclevel`

  Mit dieser Option geben Sie den genutzten Sicherheitslevel entsprechend der serverseitigen Einrichtung an. Optionen sind `noAuthNoPriv` für einen ungeschützten Zugang, `authNoPriv` für eine verschlüsselte Authentifizierung, aber unverschlüsselte Übertragung der Daten, oder `authPriv` für Verschlüsselung der Authentifizierung und der Datenübertragung.

- `-U` / `--secname`

  Hiermit geben Sie den zu verwendenden Benutzernamen an, so wie Sie ihn bei der Konfiguration des SNMP-Servers eingerichtet haben.

- `-a` / `--authproto`

  Geben Sie hier gegebenenfalls (bei Verwendung von `authNoPriv` oder `authPriv` oben) das für die Authentifizierung zu nutzende Protokoll (`MD5` oder `SHA1`) an.

- `-A` / `--authpass`

  Mit dieser Option übergeben Sie das zu nutzende Passwort, welches zu dem angegebenen Benutzernamen passen muss.

- `-P` / `--privproto`

  Wenn Sie die Datenübertragung verschlüsseln (`authPriv` oben), geben Sie hier das für die Verschlüsselung genutzte Protokoll an (`DES` oder `AES`).

- `-X` / `--privpass`

  Für eine verschlüsselte Datenübertragung müssen Sie dann außerdem mit dieser Option die hierfür zu nutzende Passphrase angeben.

Wenn Sie die weiter vorne angeführte Definition verwenden, müssen Sie die verwendeten Verfahren und Protokolle gegebenenfalls an Ihre Umgebung anpassen. Weitere Optionen können Sie über die Service-Definition angeben. Diese werden dann über das Makro `$ARG1$` weitergeleitet.

Weitere Optionen

Über weitere Optionen können Sie das Plugin an Ihre Bedürfnisse anpassen. Einige wichtige möchten wir Ihnen an dieser Stelle vorstellen:

- `-p`

  Wenn Sie einen vom Standard-Port `161` abweichenden Port für Ihren SNMP-Server verwenden, müssen Sie diesen hier angeben.

- `-u` / `--unused_ports`

  Wenn Sie einzelne Schnittstellen von der Prüfung ausnehmen möchten, etwa weil diese nicht verwendet werden, können Sie mit dieser Option eine durch Kommata separierte Liste von Ordnungsnummern hierzu angeben.

- `-x` / `--exclude`

  Wenn Sie bestimmte Schnittstellentypen von der Prüfung ausschließen wollen, können Sie sie mit dieser Option durch eine durch Kommata separierte Liste von IANA-Schnittstellentypen spezifizieren. Eine Liste der Typen und der dazugehörigen Nummern finden Sie unter *https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib*. Dabei steht `6` beispielsweise für Ethernet-Schnittstellen und `24` für lokale Loopback-Schnittstellen.
- `-t` / `--timeout`

  Wenn die Abfragen für den Standard-Timeout (15 Sekunden) bei Ihnen zu lange dauern, können Sie mit dieser Option einen kürzeren Timeout (Angabe in Sekunden) definieren.

Insbesondere durch die Anpassung über den Ausschluss von spezifischen Schnittstellen und/oder Schnittstellentypen können Sie den Nutzwert dieses Plugins gegebenenfalls steigern. Selbst bei Servern mit nur einer Schnittstelle können Sie über `-x 24` etwa das Loopback-Interface ausschließen und so nur die tatsächlichen Netzwerk-Schnittstellen abfragen.

Dienstdefinition

Zur Einbindung einer der voranstehenden Definitionen müssen Sie diese dann noch über eine Dienstdefinition mit einer oder mehreren Maschinen verknüpfen (siehe hierzu Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*). Wenn Sie beispielsweise SNMP Version 3 mit Verschlüsselung der Authentifizierung und Datenübertragung auf die lokale Maschien anwenden möchten, funktioniert dies folgendermaßen:

```
define service {
       service_description                    IfStatus
       use                                    template-service-generic
       check_command                          check_ifstatus_v3
       host_name                              localhost
}
```

Beachten Sie dabei, dass dies (für alle Versionen) eine entsprechende SNMP-Konfiguration der Maschine voraussetzt (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* beziehungsweise Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*).

Test, Rückgabewerte und Graphen

Vor der Einbindung oder bei Problemen sollten Sie die Ausführung des Plugins von der Kommandozeile testen. Für die beschriebene Einbindung der lokalen Maschine funktioniert dies etwa so:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ifstatus -H 127.0.0.1 -v 3 -L authPriv
-U icinga -a SHA1 -A "Passphrase" -P AES -X "PASSPHRASE"
OK: host '127.0.0.1', interfaces up: 2, down: 0, dormant: 0, excluded: 0,
unused: 0 |up=2,down=0,dormant=0,excluded=0,unused=0
```

Dabei müssen Sie den Benutzernamen und die Passphrasen gegebenenfalls an Ihre Umgebung anpassen. Als weiteres Beispiel im Folgenden der Test auf einer Remote-Maschine und unter Ausschluss lokaler Loopback-Schnittstellen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ifstatus -H 10.85.58.1 -v 3 -L authPriv
-U icinga -a SHA1 -A "PASSPHRASE" -P AES -X "PASSPHRASE" -x 24
OK: host '10.85.58.1', interfaces up: 14, down: 0, dormant: 0, excluded: 1,
unused: 0 |up=14,down=0,dormant=0,excluded=1,unused=0
```

Beachten Sie, dass das Plugin Ihnen hier über `excluded: 1` angibt, dass hier eine Schnittstelle ausgeschlossen wurde.

Diskussion

Das Plugin ist relativ simpel, da es nur die Schnittstellen selbst zählt und keine weiteren Daten dazu auswertet. Entsprechend einfach lässt es sich einzubinden, zumindest wenn Sie mit den Parametern für SNMP vertraut sind. Lassen Sie sich dabei bitte nicht von der Anzahl an notwendigen Parametern für SNMPv3 abschrecken. Sobald Sie die Einrichtung einmal vorgenommen und verstanden haben, ist die Einbindung unproblematisch und schnell vollzogen – und der Sicherheitsgewinn durch Verschlüsselung ist erheblich.

Funktional ermöglicht Ihnen dieses Plugin, die Anzahl der aktiven Schnittstellen zu überwachen. Dies ist natürlich insbesondere bei Maschinen mit mehreren Schnittstellen interessant. Um gezielt einzelne Schnittstellen zu überwachen, können Sie das Plugin `check_ifoperstatus` verwenden (siehe Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*).

Siehe auch

- Rezept zur Einrichtung von SNMP auf Servern: Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* beziehungsweise Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Liste der IANA-Schnittstellentypen: *https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zum Plugin `check_ifoperstatus`: Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*

6.9 Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)

Problem

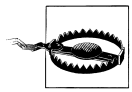
Sie möchten gezielt die Funktion wichtiger Netzwerkschnittstellen überwachen.

Lösung

In den Standard-Plugins ist zu diesem Zweck das Plugin `check_ifoperstatus` enthalten, das entsprechende Abfragen über SNMP durchführt (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* beziehungsweise Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*). Dieses ermöglicht Ihnen, Schnittstellen für die Überwachung festzulegen und Fehlermeldungen bei Änderung des Status auszugeben. Wir zeigen Ihnen im Folgenden, wie Sie das Plugin einbinden und verwenden können.

Befehlsdefinition und notwendige Optionen

Für die Einbindung benötigen Sie zunächst eine entsprechende Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*). Wir stellen Ihnen hier Varianten für verschiedenen SNMP-Versionen zur Verfügung:



SNMP-Version: Sie sollten bei jedem Gerät die höchste unterstützte Version verwenden. Sofern Sie Version 1 oder 2c verwenden müssen, weil Ihre Geräte Version 3 nicht unterstützten (etwa Windows), müssen Sie damit rechnen, dass Angreifer nicht nur den Datenverkehr, sondern auch die verwendeten Passwörter mitlesen können.

```
define command {
       command_name                    check_ifoperstatus_v1
       command_line                    $USER1$/check_ifoperstatus \
                                       -H $HOSTADDRESS$ -v 1 -C $USER11$ \
                                       -d $ARG1$ $ARG2$
}
define command {
       command_name                    check_ifoperstatus_v2
       command_line                    $USER1$/check_ifoperstatus \
                                       -H $HOSTADDRESS$ -v 2 -C $USER11$ \
                                       -d $ARG1$ $ARG2$
}

define command {
       command_name                    check_ifoperstatus_v3
       command_line                    $USER1$/check_ifoperstatus \
                                       -H $HOSTADDRESS$ -v 3 -L authPriv \
                                       -a SHA1 -U "icinga" -A $USER12$ -P AES \
                                       -X $USER12$ -d $ARG1$ $ARG2$
}
```

AES-Verschlüsselung: Das Plugin basiert auf Perl, für das eine eigene Bibliothek für die Verschlüsselung über AES vorhanden und auch notwendig ist. Wenn Sie eine Fehlermeldung wie

```
Required module Crypt/Rijndael.pm not found
```

erhalten, müssen Sie dieses noch installieren. Dies funktioniert folgendermaßen:

```
root@moni:~# apt-get install libcrypt-rijndael-perl
```

Nach der Installation verschwindet die Fehlermeldung und die Verschlüsselung funktioniert wie hier beschrieben.

Als notwendige Optionen müssen Sie dabei zunächst die Parameter für die Verbindung angeben:

- `-H` / `--hostname`

  Legen Sie mit diesem Parameter den Hostnamen oder IP-Adresse des zu überwachenden Systems fest.

- `-v` / `--snmp_version`

  Mit dieser Parameter spezifizieren Sie die zu nutzende SNMP-Version. Version 2 (für Version 2c) führt gegenüber 1 größere Zahlenräume ein, Version 3 führt gegenüber 2c eine erweiterte Benutzerverwaltung und Verschlüsselung ein.

In Abhängigkeit von der genutzten SNMP-Version müssen Sie dann weiter die jeweiligen Zugangsdaten setzen. Für Versionen 1 und 2 ist dies schlicht folgende Option:

- `-C` / `--community`

  Geben Sie hier das "community-string" genannte Passwort an.

Für Version 3 müssen Sie hingegen die folgenden Optionen verwenden:

- `-L` / `--seclevel`

  Mit dieser Option geben Sie den genutzten Sicherheitslevel entsprechend der serverseitigen Einrichtung an. Optionen sind `noAuthNoPriv` für einen ungeschützten Zugang, `authNoPriv` für Verschlüsselung bei der Authentifizierung, aber unverschlüsslte Übertragung der Daten, oder `authPriv` für Verschlüsselung der Authentifizierung und der Datenübertragung.

- `-U` / `--secname`

  Hiermit geben Sie den zu verwendenden Benutzernamen an, so wie Sie ihn bei der Konfiguration des SNMP-Servers eingerichtet haben.

- `-a` / `--authproto`

  Geben Sie hier gegebenenfalls (bei Verwendung von `authNoPriv` oder `authPriv` weiter vorne) das für die Authentifizierung zu nutzende Protokoll (`MD5` oder `SHA1`) an.

- `-A` / `--authpass`

  Mit dieser Option übergeben Sie das zu nutzende Passwort, welches zu dem angegebenen Benutzernamen passen muss.

- `-P` / `--privproto`

  Wenn Sie die Datenübertragung verschlüsseln (`authPriv` weiter vorne), geben Sie hier das für die Verschlüsselung genutzte Protokoll an (`DES` oder `AES`).

- `-X` / `--privpass`

  Für eine verschlüsselte Datenübertragung müssen Sie dann weiter mit dieser Option die hierfür zu nutzende Passphrase angeben.

Neben diesen die Verbindung spezifizierenden Parametern müssen Sie dann des Weiteren angeben, welche Schnittstelle Sie überwachen möchten. Hierzu stehen Ihnen drei Varianten zur Verfügung:

- `-k` / `--key`

  Mit dieser Option können Sie die gewünschte Schnittstelle über die Ordnungsnummer angeben, also die wievielte Schnittstelle es ist.

- `-d` / `--descr`

  Anstelle einer Ordnungsnummer können Sie mit dieser Option einen Namen angeben, über den die Schnittstelle identifiziert wird.

- `-T` / `--type`

  Alternativ zu den beiden vorhergehenden Optionen können Sie die Schnittstelle auch über den entsprechenden Typ angeben. Eine Liste der Typen und der dazugehörigen Nummern finden Sie unter *https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib*. Dabei steht beispielsweise `6` für Ethernet-Schnittstellen und `24` für lokale Loopback-Schnittstellen.

Wir haben in den voranstehenden Definitionen die Variante `-d` vorgesehen. Diese ist deutlich weniger performant als die anderen. Dafür können Sie mit ihr zweifelsfrei eine Schnittstelle angeben – bei der Angabe über `-k` kann es anderenfalls durch eine interne Änderung der Reihenfolge bereits dazu kommen, dass Ihre Prüfung nicht mehr wie gewünscht funktioniert.

Weitere Optionen

Neben diesen Pflichtoptionen können Sie das Plugin mit weiteren Optionen an Ihre Bedürfnisse anpassen. Einige dieser stellen wir Ihnen im Folgenden vor:

- `-p` / `--port`

  Wenn Sie einen vom Standard-Port `161` abweichenden Port für Ihren SNMP-Server verwenden, müssen Sie ihn mit dieser Option angeben.

- `-w` / `--warn`

  Wenn die abgefragte Schnittstelle im Zustand `dormant`, also aus unbekannten Gründen inaktiv ist, wird das Plugin als Vorgabewert den Status `CRITICAL` zurückgeben. Mit dieser Option können Sie einen hiervon abweichenden Status für diesen Zustand festlegen. Mögliche Werte sind `i` (ignore), um den Zustand zu ignorieren, `w` für `WARNING` und `c` für `CRITICAL`.

- `-D` / `--admin-down`

  Wenn Schnittstellen administrativ inaktiv gesetzt werden, befinden sie sich im sogenannten `admin-down`-Zustand. Für diesen wird das Plugin als Vorgabewert den Status `WARNING` zurückgeben. Mit dieser Option können Sie den zurückzugebenden Status manuell festlegen. Wie oben haben Sie dabei die Wahl zwischen `i` (ignore), um den Zustand zu ignorieren, `w` für `WARNING` und `c` für `CRITICAL`.

- `-t` / `--timeout`

  Wenn die Abfragen für den Standard-Timeout (15 Sekunden) bei Ihnen zu lange dauern, können Sie mit dieser Option einen abweichenden Timeout (Angabe in Sekunden) definieren.

Dienstdefinition

Um die voranstehende Kommandodefinition einzubinden, benötigen Sie dann des Weiteren eine entsprechende Dienstdefinition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*), die für die Einbindung der lokalen Maschine etwa so aussieht:

```
define service {
       service_description                    IfOperStatus
       use                                    template-service-generic
       check_command                          check_ifoperstatus_v3!eth0
       host_name                              localhost
}
```

Beachten Sie hierbei insbesondere, dass Sie die verwendete Vorlage (hier `template-service-generic`) an Ihre Umgebung anpassen müssen.

Test, Rückgabewerte und Graphen

Vor der Einbindung und bei Problemen sollten Sie das Plugin von der Kommandozeile aus testen. Wenn Sie etwa die voranstehende Einbindung testen möchten, funktioniert das folgendermaßen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:/usr/local/icinga/libexec$ /usr/local/icinga/libexec/check_ifoperstatus
-v 3 -d eth0 -H localhost -U icinga -a SHA1 -A "PASSPHRASE" -L authPriv -P AES
-X "PASSPHRASE"
OK: Interface eth0 (index 2) is up.
```

Das Plugin gibt keine Performancedaten zurück, was in der Natur der Abfrage liegt. Entsprechend gibt es zu diesem Plugin auch keine Graphen.

Diskussion

Während Sie mit `check_ifstatus` (siehe Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)*) alle Schnittstellen einer Maschine prüfen, können Sie das hier beschriebene `check_ifoperstatus` einsetzen, um gezielt einzelne Schnittstellen abzufragen. Wenn Sie die Schnittstellen bei Wartungs- beziehungsweise Änderungsarbeiten vorher administrativ deaktivieren, können Sie mit der Option `-D` dafür sorgen, dass dann keine Fehlermeldungen generiert werden. So können Sie sich in diesen Fällen die Hinterlegung von Ausfallzeiten über die Oberfläche sparen (siehe hierzu Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*).

Siehe auch

- Rezept zur Einrichtung von SNMP auf Servern: Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* beziehungsweise Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Liste der IANA-Schnittstellentypen: *https://www.iana.org/assignments/ianaiftype-mib/ianaiftype-mib*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zum Plugin `check_ifstatus`: Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)*
- Rezepte zur Nutzung der Oberflächen: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*

6.10 Überwachung eines FTP-Servers (check_ftp)

Problem

Sie möchten eine FTP-Server überwachen. Die Erreichbarkeit soll über das Protokoll FTP (File Transfer Protocol) geprüft werden.

Lösung

Im Paket der Standardplugins ist zu diesem Zwecke `check_ftp` als Variation von `check_tcp` (siehe Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*) enthalten. Dieses ermöglicht Ihnen, den entsprechenden Service-Port auf Erreichbarkeit zu überprüfen und dabei die Antwortzeiten auszuwerten. Wir zeigen Ihnen im Folgenden, wie Sie dieses Plugin nutzen können.

Befehlsdefinition und Optionen

Bei der Installation sollte bereits eine Definition für dieses Plugin erstellt worden sein. Bei unserer Referenzinstallation sah diese wie folgt aus:

```
# 'check_ftp' command definition
define command{
       command_name                    check_ftp
       command_line                    $USER1$/check_ftp -H $HOSTADDRESS$ $ARG1$
 }
```

Als Pflichtargument ist hier die zu prüfende Maschine vorgesehen, gewünschte weitere Parameter können Sie über die Service-Definition nutzen. Wie die Verbindung aufgebaut wird (gesonderte Ports, IP-Version usw.) können Sie gegebenenfalls weiter über die in Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)* zum Plugin `check_tcp` beschriebenen Parameter beeinflussen. Ebenso stehen Ihnen die dort beschriebenen Optionen für zusätzliche Prüfungen, beispielsweise für die Suche nach einer Zeichenkette, zur Verfügung.

Dienstdefinition

Um die Kommandodefinition als Service einzubinden (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*), können Sie eine Definition entsprechend der folgenden verwenden:

```
define service{
       service_description             FTP
       use                             template-service-generic
       check_command                   check_ftp!-j
       host_name                       FTP-Server
}
```

Dabei wird das Kommando als Service `FTP` für eine bereits definierte Maschine (siehe Rezept 4.1, *Maschinen einbinden (host)*) `FTP-Server` ausgeführt. Zusätzlich haben wir hier über den Parameter `-j` festgelegt, dass die Antworten des Servers nicht in der Ausgabe des Plugins wiedergegeben werden sollen. Die hier üblicherweise gesendete Willkommensnachricht, die vielen Zeilen umfasst, wird sonst in der Detailansicht des Services angezeigt.

Test, Rückgabewerte und Graphen

Die normale Funktion des Plugins können Sie auf der Kommandozeile wie folgt testen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen .

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ftp -H ftp.mozilla.org
FTP OK - 0.340 second response time on port 21 [220-
220- ftp.mozilla.org / archive.mozilla.org - files are in /pub/mozilla.org
220-
220- Notice: This server is the only place to obtain nightly builds and needs to
220- remain available to developers and testers. High bandwidth servers that
220- contain the public release files are available at ftp://releases.mozilla.org/
220- If you need to link to a public release, please link to the release server,
220- not here. Thanks!
220-
220- Attempts to download high traffic release files from this server will get a
220- "550 Permission denied." response.
220]|time=0.339714s;;;0.000000;10.000000
```

Durch den in der voranstehenden Definition verwendeten Parameter `-j` ändert sich die Ausgabe dabei wie folgt:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ftp -H ftp.mozilla.org -j
FTP OK - 0.340 second response time on port 21|time=0.340228s;;;0.000000;10.000000
```

Unabhängig davon liefert das Plugin die Antwortzeiten als Performancedaten. Bei PNP4Nagios wird durch die Standardvorlage dabei ein Graph wie in Abbildung 6-8 erzeugt.

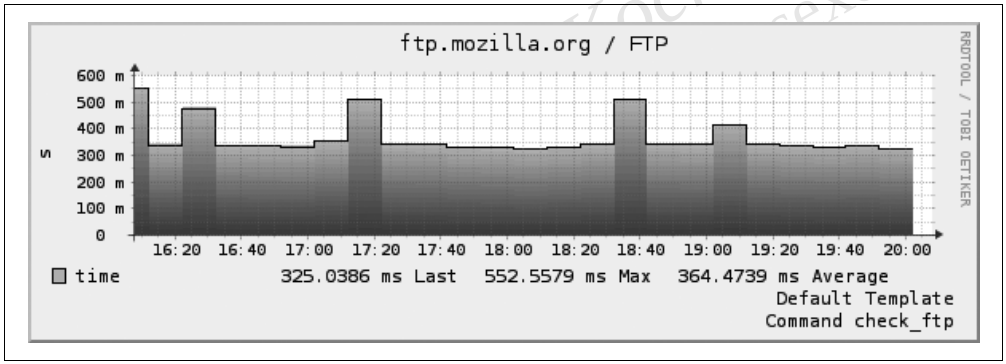


Abbildung 6-8: PNP4Nagios-Graph für Antwortzeiten bei FTP

Die Auflösung des Graphen hängt dabei wie immer vom genutzten Test-Intervall ab.

Diskussion

Über das Plugin `check_ftp` können Sie die Erreichbarkeit eines FTP-Servers auf einfache Weise überprüfen. Tatsächlich wird dabei das Plugin `check_tcp` genutzt, wobei der Port automatisch auf `21` und der erwartete Rückgabewert auf den Statuscode `220` gesetzt werden. Eine erweiterte Funktionalität wie den Test auf erfolgreiche Anmeldungen oder das Herunterladen von Dateien bietet das Plugin dabei jedoch nicht. Wenn Sie entsprechende Tests realisieren möchten, können Sie dies entweder über selbstgeschriebene Scripte oder möglicherweise über von anderen geschriebene Plugins erreichen (siehe dazu Rezepte in Kapitel 7, *Erstellung eigener und Einbindung externer Plugins*).

Siehe auch

- Rezept zur Überwachung beliebiger TCP/UDP-Ports: Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Kapitel zur Erstellung eigener und Einbindung externer Plugins: Kapitel 7, *Erstellung eigener und Einbindung externer Plugins*

6.11 Überwachung eines Webservers (check_http)

Problem

Sie möchten einen Webserver überwachen. Dazu wollen Sie dessen Erreichbarkeit über die Protokolle HTTP (Hypertext Transfer Protocol) und/oder HTTPS (HTTP Secure) testen.

Lösung

Zu diesem Zweck ist in den Standard-Plugins `check_http` enthalten. Wir zeigen Ihnen im Folgenden, wie Sie dieses Plugin einbinden und nutzen können.

Befehlsdefinition und notwendige Optionen

Typischerweise wurde Ihnen mit der Installation bereits eine Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) für dieses Plugin mitgeliefert, die bei uns wie folgt aussieht:

```
# 'check_http' command definition
define command{
       command_name                    check_http
       command_line                    $USER1$/check_http -I $HOSTADDRESS$ \
                                       $ARG1$
}
```

Als Pflichtparameter müssen Sie lediglich den Rechner spezifizieren, an den die HTTP-Anfrage geschickt werden soll. Den in dieser Definition vorgesehen Parameter `-I` sowie die außerdem für diesen Zweck nutzbaren Parameter sind folgende:

- `-I /  --IP-address`

  Mit dieser Option legen Sie die abzufragende Maschine fest. Sie können dabei entweder eine IP-Adresse oder aber den voll qualifizierten Domain-Namen verwenden. Die IP-Adresse ist dabei die bevorzugte Form, sofern Sie hiermit den eigentlichen Webdienst (die technische Auslieferung der hinterlegten Seite) testen wollen und die Namensauflösung der Domäne separat prüfen.

- `-H / --hostname`

  Wenn Ihr Server anhand von virtuellen Zuteilungen, den sogenannten virtual hosts (vhosts), unterschiedliche Seiten ausliefert, können Sie über diesen Parameter die entsprechende Kopfzeile für die Anfrage setzen. Geben Sie dazu hier den entsprechenden Namen inklusive des dazugehörigen Ports an, also etwa *http://www.andere-domain.de/*.

- `-S / --ssl`

  Diese Option weist das Plugin an, die Verbindung verschlüsselt mittels SSL (Secure Sockets Layer) herzustellen. Der Standardport wird dadurch auf 443 gesetzt.

- `-p / --port`

  Mit dieser Option können Sie einen Port für die Anfrage festlegen. Dies wird beispielsweise dann erforderlich, wenn die Anfrage umgeleitet wird und deshalb nicht der eigentliche Port (Standardport 80 beziehungsweise 443 bei verschlüsselten Verbindungen) verwendet werden soll.

- `-4` oder `-6`

  Mit dieser Option legen Sie die zu nutzende Version des IP-Protokolls fest. Als Standard ist Version 4 vorgesehen.

Je nach Beschaffenheit Ihrer Infrastruktur können Sie sich entsprechend angepasste Vorlagen schaffen (Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*).

Weitere Optionen

Neben den bereits vorgestellten Optionen, die sich nur auf die technische Art der Anfrage beziehen, steht Ihnen eine Fülle von weiteren Optionen zur Verfügung, mit denen sich die angefragte Seite und die Ergebnisverarbeitung spezifizieren lassen. Zunächst möchten wir Ihnen weitere Parameter vorstellen, die die Art der Anfrage beeinflussen:

- `-C / --certificate`

  Wenn Sie SSL nutzen, können Sie mit dieser Option auf eine reine Zertifikatsprüfung umschalten. Dabei wird kein Inhalt abgefragt, sondern lediglich die Gültigkeitsdauer des Zertifikates geprüft. Übergeben Sie dazu als Wert die Anzahl an Tagen die das Zertifikat noch gültig sein muss, um den Test zu bestehen. Also beispielsweise `-C 100`, um eine Warnung zu erhalten, sobald das Zertifikat in weniger als 100 Tagen abläuft.

- `-u / --url`

  Mit dieser Option können Sie den URL angeben, der abgerufen werden soll. Auf diese Weise können Sie etwa gezielt spezifische Seiten anstelle der Hauptseite abrufen und testen.

- `-a / --authorization`

  Mit dieser Option können Sie einen Benutzernamen und ein Password für die HTTP-Authentifizierung in der Form `name:passwort` übergeben, sofern dies erforderlich ist.

- `-t / --timeout`

  Mit dieser Option können Sie einen vom Vorgabewert 10 Sekunden abweichenden Timeout angeben, nach dem die Abfrage abgebrochen und als erfolglos gewertet wird.

Es gibt daneben noch weitere Optionen, die die Verbindung betreffen, etwa um spezifische Kopfzeilen zu setzen. Auf diese werden wir hier nicht weiter eingehen.

Neben diesen die Anfrage beeinflussenden Optionen steht Ihnen eine Auswahl weiterer Optionen zur Verfügung, um die Ergebnisverarbeitung zu beeinflussen:

- `-e / --expect`

  Mit dieser Option können Sie eine oder mehrere Zeichenketten angeben, von denen zumindest eine in der ersten Zeile der Antwort (der Statuszeile) enthalten sein soll. Normalerweise prüft das Plugin hier auf die Zeichenkette »HTTP/1.«, die auf eine technisch erfolgreiche Antwort eines Webservers hinweist. Das Plugin kennt die HTTP-Status und wertet diese aus (siehe *https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol*). Wenn Sie diese Option setzen, wird das Plugin nur noch auf die von Ihnen spezifizierten Zeichenketten prüfen.

- `-s / --string`

  Mit dieser Option können Sie den Inhalt der zurückgelieferten Seite auf das Vorhandensein einer Zeichenkette prüfen lassen.

- `-l / --linespan`

  Mit dieser Option können Sie für die Nutzung von regulären Ausdrücken angeben, dass Zeilenumbrüche ignoriert werden sollen (siehe auch Optionen `-r` und `-R`). Auf diese Weise können Sie unabhängig von möglicherweise dynamischen Umbrüchen auf das gewünschte Ergebnis testen.

- `-r / --regex` beziehungsweise `--ereg`

  Mit dieser Option können Sie den Inhalt der zurückgelieferten Seite mit Hilfe eines regulären Ausdrucks überprüfen, der in der Seite enthalten sein muss (siehe hierzu auch Option `-i` im Nachfolgenden). Im Gegensatz zur folgenden Option wird dabei auf die Groß-/Kleinschreibung geachtet.

- `-R / --eregi`

  Mit dieser Option können Sie einen regulären Ausdruck angeben, der in der zurückgelieferten Seite enthalten sein muss (siehe hierzu auch Option -i im Nachfolgenden). Bei dem Test wird im Gegensatz zur vorhergehenden Option nicht auf die Groß-/Kleinschreibung geachtet.

- -i / --invert-regex

  Bei Nutzung einer der Optionen zu regulären Ausdrücken (siehe Optionen -r oder -R), können Sie mit dieser Option festlegen, dass ein Auffinden des Ausdrucks nicht zum Status `OK` sondern stattdessen zum Status `CRITICAL` führt. Auf diese Weise können Sie einfach testen lassen, dass bestimmte Warn- oder Fehlerhinweise nicht auf einer Seite vorkommen.

- `-M / --max-age`

  Wenn Sie überprüfen möchten, wie aktuell eine Seite ist, können Sie mit dieser Option ein maximales Alter festlegen. Ein Zahlenwert wird dabei als Angabe in Sekunden interpretiert, daneben sind auch Angaben in Minuten `m`, Stunden `h` oder Tagen `d` möglich, also etwa `1d` für einen Tag.

- `-f / --onredirect`

  Mit dieser Option können Sie festlegen, welchen Status das Plugin zurückgeben soll, falls die Anfrage mittels eines sogenannten HTTP-Redirect weitergeleitet wurde. Mögliche Werte sind hier zunächst `OK`, `WARNING` und `CRITICAL`, mit denen Sie direkt den jeweiligen Status setzen können. Darüber hinaus können Sie die Werte `FOLLOW`, `STICKY` und `STICKYPORT` verwenden, um der Umleitung zu folgen und den Status abhängig von der nach der Weiterleitung zurückgegebenen Seite setzen zu lassen, wobei `STICKY` und `STICKYPORT` festlegen, dass der Server und der Serviceport gleich bleiben müssen.

- `-m / --pagesize`

  Mit dieser Option können Sie die Größe der zurückgelieferten Seite überprüfen. Übergeben Sie hierzu entweder nur die Mindestgröße in Bytes oder einen durch eine Mindestgröße und eine Maximalgröße festgelegten Bereich, wobei diese Größen durch einen Doppelpunkt separiert werden, also etwa `10:1000` für mindestens 10 Byte und maximal 1000 Byte.

- `-w / --warning` und `-c / --critical`

  Setzen Sie mit diesen Optionen Zeiten in Millisekunden, um den Status in Abhängigkeit von der Antwortzeit der Abfrage zu setzen.

Bei diesem Plugin können Sie also ganz verschiedene Aspekte eines Webservers testen. Im Rahmen der nachfolgenden Diskussion zeigen wir Ihnen entsprechende Varianten auf.

Dienstdefinition

Da Nagios/Icinga selbst ein Web-Frontend verwendet, wurde bei der Installation bereits eine entsprechende Dienst-Definition erstellt. Bei unserer Referenzinstallation war dies die folgende, auf der voranstehenden Befehlsdefinition aufbauende Deklaration:

```
define service{
       use                                  local-service ; Name of service template \
                                            to use
       host_name                            localhost
       service_description                  HTTP
       check_command                        check_http
       notifications_enabled                0
 }
```

Dabei wurde die Vorlage `local-service` verwendet, um den Test auf der lokalen Maschine auszuführen. Zusätzlich wurden hierbei die Benachrichtigungen über die Option `notifications_enabled` (siehe hierzu Rezept 4.2, *Dienste: Befehle mit Maschinen*

verknüpfen (service)) deaktiviert. Der auf diese Weise durchgeführte Test wird immer `OK` zurückgeben, solange der Webserver auf die Anfrage erfolgreich antwortet. Dies ist der Minimaltest, den ein Webserver normalerweise immer bestehen sollte.

Test, Rückgabewerte und Graphen

Sie können die Durchführung des Plugins mit den genannten Definitionen testen, indem Sie das entsprechende Kommando auf der Kommandozeile ausführen, ohne weitere Parameter also etwa wie folgt:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_http -I 127.0.0.1
HTTP OK: HTTP/1.1 200 OK - 453 bytes in 0.000 second response time
|time=0.000389s;;;0.000000 size=453B;;;0
```

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

Die Rückgabe des Plugins umfasst jeweils durch ein Pipe-Symbol »|« separierte Performancedaten, die für eine graphische Auswertung verwendet werden können. Wenn Sie PNP4Nagios nutzen, wurde Ihnen mit der Installation bereits eine entsprechende Graphen-Definition bereitgestellt, die aus den Daten zwei Graphen erzeugt, die die Antwortzeit (siehe Abbildung 6-9) beziehungsweise -größe (siehe Abbildung 6-10) im zeitlichen Verlauf darstellen. Beachten Sie bitte, dass die hier abgebildeten Graphen auf einem Prüfintervall von einer Minute basieren. Wenn Sie größere Intervalle verwenden, werden Ihre Graphen gegebenenfalls gröber aufgelöst.

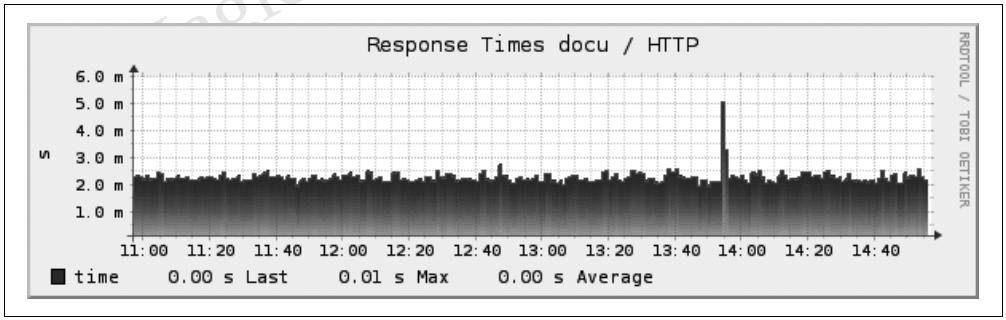


Abbildung 6-9: PNP4Nagios-Graph für HTTP-Antwortzeiten

Sofern der Test bei Ihnen wie gewünscht funktioniert, können Sie die entsprechenden Benachrichtigungen aktivieren. Entfernen Sie hierzu die Zeile mit der Option `notifications_enabled` aus der voranstehenden Service-Definition. Dann kommt der in der verwendeten Vorlage gesetzte Wert für diese Option zum Tragen und die Benachrichtigungen werden verarbeitet.

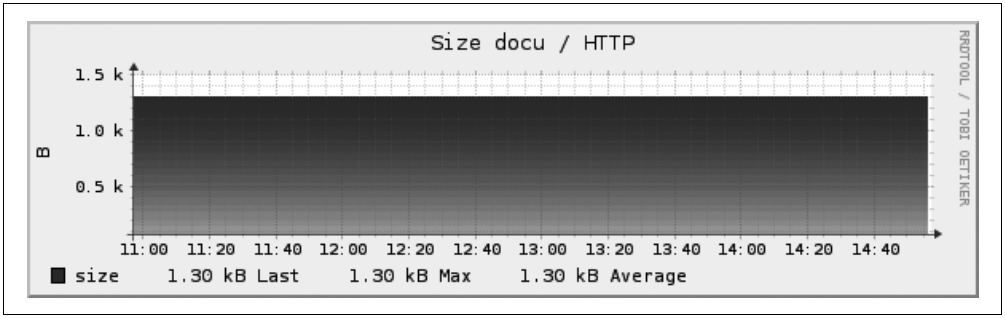


Abbildung 6-10: PNP4Nagios-Graph für HTTP-Antwortgröße

Diskussion

Die vielfältigen Optionen ermöglichen Ihnen eine umfangreiche Anpassung dieses Plugins, sowohl bezüglich der Abfrage (SSL-Verschlüsselung, Anmeldung, Port, Timeout) als auch bezüglich der Auswertung der Ergebnisse (Status, Art, Größe, Zeit, Zertifikat). Bei wichtigen Webservern können Sie entsprechend Ihrer Bedürfnisse mehrere Dienste auf Basis dieses Plugins verwenden.

Unseres Erachtens nach wichtige Variationen stellen wir Ihnen in dem Rezept 8.7, *Webserver überwachen* vor. Sie können diese entsprechend für die auf dem Webserver laufenden Webseiten einrichten. In jedem Fall sollten Sie die im Nachfolgenden angeführten Tests durchführen.

Performance

Der automatisch eingerichtete Test überprüft nur, ob überhaupt eine Seite ausgeliefert wird. Bei produktiv eingesetzten Webservern sollten Sie diesen Test zusätzlich mit Warnschwellen für die Antwortzeiten versehen, damit Sie mögliche Performance-Engpässe erkennen. Fügen Sie Ihrer Dienstdefinition dazu einfach die beiden Optionen mit der Anzahl an Millisekunden hinzu:

```
check_command                    check_http!--warning 50 --critical 100
```

Unterschiedliche Inhalte eines Servers

Wenn ein Server mehrere Dienste über eine Domain zur Verfügung stellt, können Sie diese separat abfragen. Nutzen Sie hierzu Definitionen mit dem Parameter `--url`, um jeweils entsprechende Dienste einzurichten:

```
check_command                    check_http!--url "/book"
```

Sie können den konkreten Inhalt dann bei Bedarf wie beschrieben überprüfen lassen, etwa dahingehend, ob bestimmte Wörter enthalten sind oder die Seiten aktuell generiert wurden.

Siehe auch

- Rezept zur Vereinfachung der Konfiguration mit Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zur Überwachung eines Webservers: Rezept 8.7, *Webserver überwachen*

6.12 Überwachung eines SSH-Dienstes (check_ssh)

Problem

Sie möchten einen SSH-Server überwachen. Die Erreichbarkeit des Servers soll über SSH (Secure SHell) getestet werden.

Lösung

Im Standardpaket der Plugins ist zu diesem Zwecke `check_ssh` enthalten. Wir zeigen Ihnen im Folgenden, wie Sie dieses einsetzen können.

Befehlsdefinition und notwendige Optionen

Über die Installation der Plugins wird Ihnen typischerweise bereits eine Befehlsdefinition zur Verfügung gestellt. Bei unserer Referenzinstallation war dies die folgende:

```
# 'check_ssh' command definition
define command{
       command_name                    check_ssh
       command_line                    $USER1$/check_ssh $ARG1$ $HOSTADDRESS$
 }
```

Dabei wird der Pfad zum Plugin über das Makro `$USER1$` bereitgestellt und als notwendiger Parameter der zu prüfende Host als Makro `$HOSTADDRESS$` übergeben. Der Einheitlichkeit halber können Sie der Angabe der Zielmaschine dabei auch das sonst übliche `-H` voranstellen, es ist allerdings bei diesem Plugin nicht zwingend erforderlich. Weitere, optionale Parameter können in der Service-Definition über das Makro `$ARG1$` angegeben werden (siehe Rezept zu Kommandodefinitionen und Makros Rezept 4.3, *Befehle hinzufügen (command)*).

Weitere Optionen

Für das Plugin stehen weitere Parameter zur Verfügung, unter anderem die Folgenden (für eine vollständige Übersicht rufen Sie das Plugin bitte auf der Kommandozeile mit dem Parameter `--help` auf):

- `-p / --port`

  Mit dieser Option können Sie dem Plugin den Port mitteilen, falls dieser bei Ihrem Server vom Standardport 22 abweicht.

- `-4 / --use-ipv4` oder `-6 / --use-ipv6`

  Mit dieser Option können Sie die zu nutzende Version des Internet-Protokolls angeben.

- `-t / --timeout`

  Wenn Sie einen vom Vorgabewert 10 Sekunden abweichenden Timeout wünschen, können Sie diesen mit dieser Option angeben (in Sekunden).

- `-r / --remote-version`

  Mit dieser Option können Sie eine Zeichenkette vorgeben, die mit der vom Server zurückgegebenen Zeichenkette verglichen wird.

Dienstdefinition

Die im Voranstehenden angeführte Befehlsdefinition wurde für die Monitoring-Maschine selbst dann bereits eingebunden. Bei unserer Referenzinstallation war die entsprechende Dienstdefinition folgende:

```
define service {
       use                             local-service
       host_name                       localhost
       service_description             SSH
       check_command                   check_ssh
       notifications_enabled           0
}
```


Beachten Sie, dass dabei über die Option `notifications_enabled` (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) Benachrichtigungen deaktiviert sind. Sofern auf Ihrer Monitoring-Maschine ein SSH-Server läuft und das Plugin wie gewünscht funktioniert, können und sollten Sie diese Option gegebenenfalls entfernen.

Test, Rückgabewerte und Graphen

Um das Plugin zu testen, können Sie es wie folgt auf der Kommandozeile ausführen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ssh -H 127.0.0.1
SSH OK - OpenSSH_5.5p1 Debian-6+squeeze2 (protocol 2.0)
```

Beachten Sie, dass das Plugin keine Performancedaten zurückgeliefert. Entsprechend stehen hier keine Graphen zur Verfügung.

Diskussion

Das Plugin `check_ssh` überprüft die tatsächliche Funktion eines SSH-Servers. Sie sollten zusätzlich die Möglichkeit der Versionsüberprüfung nutzen. Dies gilt insbesondere dann, wenn Sie bei Versionsänderungen in jedem Fall involviert sind, da Sie SSH-Sicherheitsupdates verfolgen und/oder den SSH-Server selbst kompilieren. Wenn Sie die gewünschte Version zentral im Monitoring-System einpflegen, werden Ihnen notwendige Updates automatisch als Warnungen angezeigt.

Beachten Sie dabei gegebenenfalls, dass dem beim eben gezeigten Aufruf zurückgegebenen String die Protokollversion in Klammern angehängt wurde. Für die Angabe der zu testenden Zeichenkette müssen Sie also entsprechend nur den Teil davor verwenden. In unserem Falle sähe dies folgendermaßen aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ssh -H 127.0.0.1 -r
"OpenSSH_5.5p1 Debian-6+squeeze2"
SSH OK - OpenSSH_5.5p1 Debian-6+squeeze2 (protocol 2.0)
```

Da das Plugin keine Performancedaten zurückgibt, können Sie sich zunächst keinen Graphen dazu zeichnen lassen. In der Regel wird dies auch nicht nötig sein, da Sie die Netzwerk-Erreichbarkeit vermutlich bereits über das Plugin `check_ping` (siehe Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*) testen. Die dabei ermittelten Performancedaten werden in Standardumgebungen typischerweise denen der Erreichbarkeit des SSH-Ports entsprechen. Wenn Sie jedoch gesondert einen Graphen zu den Antwortzeiten am entsprechenden Port möchten, können Sie hierzu zusätzlich das Plugin `check_tcp` (siehe Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*) wie folgt verwenden:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_tcp -H 127.0.0.1 -p 22
TCP OK - 0.000 second response time on port 22|time=0.000239s;;;0.000000;10.000000
```

Bei einer entsprechenden Einbindung als Service und Nutzung eines Graphers erhalten Sie dann auch einen entsprechenden Graphen, wie im Rezept beschrieben.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Überwachung der Erreichbarkeit mit Ping: Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*
- Rezept zur Überwachung beliebiger TCP/UDP-Ports: Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*

6.13 Überwachung des E-Mail-Versanddienstes (check_smtp)

Problem

Sie möchten einen E-Mail-Server überwachen. Die Funktionalität des Versanddienstes soll über das SMTP-Protokoll (Simple Mail Transfer Protocol) beziehungsweise SMTPS (Simple Mail Transfer Protocol Security) geprüft werden.

Lösung

Für das beim Versand von E-Mails genutzte SMTP (Simple Mail Transfer Protocol) beziehungsweise die mit TLS abgesicherte Variante SMTPS ist im Satz der Standardplugins check_smtp enthalten. Wir beschreiben Ihnen im Folgenden, wie Sie das Plugin testen, einrichten und verwenden können.

Befehlsdefinition und notwendige Optionen

Mit der Installation wurde typischerweise bereits eine Befehlsdefinition erstellt. In unserem Referenzsystem war dies Folgende:

```
# 'check_smtp' command definition
define command{
       command_name                    check_smtp
       command_line                    $USER1$/check_smtp -H $HOSTADDRESS$ \
                                       $ARG1$

}
```

Als Pflichtoption ist dabei nur die folgende Option vorgesehen:

- `-H / --host`

  Mit dieser Option geben Sie die Adresse der zu prüfenden Maschine an.

Damit Sie in der Service-Definition bei Bedarf weitere Optionen setzen können, haben wir hier `$ARG1$` vorgesehen (siehe Rezept 4.3, *Befehle hinzufügen (command)*).

Weitere Optionen

Als weitere die Verbindung betreffende Optionen stehen Ihnen dann zunächst außerdem die folgenden Optionen zur Verfügung:

- `-p / --port`

  Mit diesem Parameter können Sie einen vom Vorgabewert `25` abweichenden Port angeben.
- `-4 / --use-ipv4 oder -6 / --use-ipv6`

  Mit dieser Option können Sie festlegen, welche Version des IP genutzt werden soll.
- `-D / --certificate`

  Mit dieser Option können Sie die Plugins bei Nutzung verschlüsselter Verbindungen auf eine Zertifikatsprüfung umstellen. Übergeben Sie hier die Anzahl der Tage, die das Zertifikat noch gültig sein muss, damit der Test als bestanden gilt.

- `-w / --warning` und `-c / --critical`

  Mit diesen Optionen legen Sie bei Prüfung der E-Mail-Funktion Schwellenwerte für die Status `WARNING` und `CRITICAL` bezüglich der Antwortzeit fest.

- `-t / --timeout`

  Wenn die Antworten eines Servers zu lange dauern, um von den Plugins noch erfasst zu werden, können Sie mit dieser Option eine längere Zeitspanne in Sekunden angeben, die die Plugins auf eine Antwort warten sollen (Vorgabewert: 10).

Wie genau über SMTP geprüft werden soll, können Sie darüber hinaus mit den folgenden, zusätzlichen Parametern vorgeben:

- `-S / --starttls`

  Mit dieser Option geben Sie an, dass TLS für die Verschlüsselung der Verbindung genutzt werden soll. Beachten Sie, dass das Plugin SSL-Verschlüsselung nicht unterstützt.

- `-C / --command`

  Wenn Sie spezifische Befehle auf dem SMTP-Server ausführen möchten, können Sie sie mit dieser Option jeweils als Zeichenketten angeben.

- `-R / --response`

  Für jeden Befehl, den Sie mittels der Option **`-C`** festgelegt haben, können Sie mit dieser Option die erwartete Antwort als Zeichenkette festlegen.

- `-F / --fqdn`

  Mit dieser Option können Sie einen FQDN (Fully Qualified Domain Name) angeben, der in der HELO-Nachricht zum Server geschickt werden soll.

- `-A / --authtype`

  Das SMTP-Plugin unterstützt den Test der Authentifizierung. Um diesen Test zu nutzen, müssen Sie mit dieser Option `LOGIN` als Verfahren festlegen, welches das derzeit einzige unterstützte Verfahren ist.

- `-U / --authuser`

  Wenn Sie den Anmeldevorgang testen möchten, geben Sie hier den zu verwendenden Benutzernamen an.

- `-P / --authpass`

  Hiermit stellen Sie gegebenenfalls das mit dem Benutzernamen zu verwendende Passwort zur Verfügung.

Dienstdefinition

Um die im Voranstehenden angeführte Befehlsdefinitionen einzubinden, müssen Sie entsprechende DienstDienstdefinitionen (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) für bereits angelegte Maschinen (siehe Rezept 4.1, *Maschinen einbinden (host)*) erstellen. Eine Einbindung für den Test mit Verschlüsselung können Sie dabei mit der folgenden Definition erreichen:

```
define service {
       service_description                    SMTP
       use                                    template-service-generic
       check_command                          check_smtp!-p 587 -S
       host_name                              smtp.googlemail.com
}
```

Bitte beachten Sie, dass wir hier eine Vorlage `template-service` referenzieren und die Prüfung einem bereits konfigurierten Gerät `smtp.googlemail.com` zuweisen. Passen Sie diese Angaben gegebenenfalls an Ihre Bedürfnisse an.

Test, Rückgabewerte und Graphen

Die Funktionen der Plugins können Sie vor dem Einbinden von der Kommandozeile aus testen, für die weiter vorne angeführte einfache Version der Einbindung zum Beispiel folgendermaßen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_smtp -H smtp.googlemail.com -p 587 -S
SMTP OK - 0.159 sec. response time|time=0.159199s;;;0.000000
```

Die Plugins geben beim Test der E-Mail-Funktionalität (im Gegensatz zum Test von Zertifikaten bei Verschlüsselung) immer Performancedaten zurück. Wenn Sie PNP4Nagios einsetzen, erhalten Sie dadurch automatisch Graphen. Abbildung 6-11 zeigt den Graphen mit den Antwortzeiten, den Sie beim Plugin `check_smtp` gegebenenfalls automatisch erhalten.

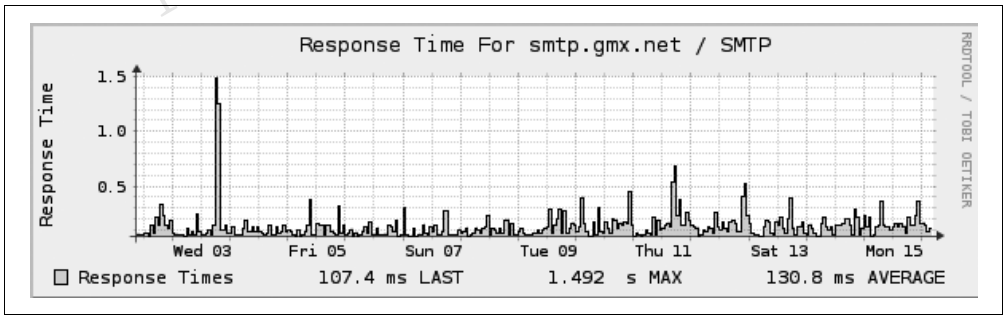


Abbildung 6-11: PNP4Nagios Bild zu check_smtp

Diskussion

Das sozusagen mitgelieferte Plugin `check_smtp` ermöglicht Ihnen, unkompliziert einfache Tests für die Auslieferungsfunktionen von E-Mail-Servern zu erstellen. Den Empfang von

E-Mails haben wir im Rezept 6.14, *Überwachung der E-Mail-Auslieferungsdienste (check_pop, check_imap)* abgehandelt. Weitere Beispiele für den Einsatz der zur Verfügung stehenden Optionen finden Sie im Rezept 8.6, *E-Mail-Server überwachen*. Diese Tests sind bei der heutigen Bedeutung von E-Mail wichtig, selbst wenn Sie keinen eigenen E-Mail-Server betreiben. Wir empfehlen Ihnen in diesem Fall, die entsprechenden Dienste bei Ihrem Provider zu überwachen.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Einbindung einer Maschine: Rezept 4.1, *Maschinen einbinden (host)*
- Überwachung der E-Mail-Auslieferungsdienste: Rezept 6.14, *Überwachung der E-Mail-Auslieferungsdienste (check_pop, check_imap)*
- Überwachung eines E-Mail-Servers: Rezept 8.6, *E-Mail-Server überwachen*

6.14 Überwachung der E-Mail-Auslieferungsdienste (check_pop, check_imap)

Problem

Sie möchten einen E-Mail-Server überwachen. Die Erreichbarkeit der Auslieferungsdienste soll über die Protokolle IMAP (Internet Message Access Protocol) beziehungsweise IMAPS (Internet Message Access Protocol Secure) und/oder POP3 (Post Office Protocol Version 3) beziehungsweise POP3S (Post Office Protocol Version 3 Secure) überprüft werden.

Lösung

In den Standardplugins sind hierzu die Plugins check_imap, `check_simap`, `check_pop` und `check_spop` enthalten, deren Einbindung und Nutzung wir Ihnen im Folgenden erläutern. Bei diesen Plugins handelt es sich um Variationen des Plugins `check_tcp` (siehe Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*). Sie enthalten Modifikationen für die jeweiligen Protokolle.

Befehlsdefinition und notwendige Optionen

Bei der Installation der Plugins werden normalerweise bereits Befehlsdefinitionen für diese Plugins eingerichtet. Bei unserer Referenzinstallation waren dies die Folgenden:

```
# 'check_imap' command definition
define command {
```

```
        command_name                    check_imap
        command_line                    $USER1$/check_imap -H $HOSTADDRESS$ \
                                        $ARG1$
}

# 'check_pop' command definition
define command {
        command_name                    check_pop
        command_line                    $USER1$/check_pop -H $HOSTADDRESS$ $ARG1$
}
```

Diese identisch aufgebauten Definitionen (zu Befehlsdefinitionen und Makros siehe Rezept 4.3, *Befehle hinzufügen (command)*) verwenden zunächst das Makro `$USER1$`, um den Pfad zu den Plugins festzulegen. Über das Makro `$HOSTADDRESS$` übergeben sie die Adresse der Zielmaschine. Weitere, in den Dienstdefinitionen angegebene Optionen werden über das Makro `$ARG1$` übernommen.

Dabei wird der jeweils einzige Pflichtparameter eingeplant:

- `-H / --host`
  Mit dieser Option geben Sie die Adresse der zu prüfenden Maschine an.

Weitere Optionen

Sie können die Plugins mit einer Anzahl weiterer Optionen an Ihre Bedürfnisse anpassen. Einige wichtige dieser Optionen möchten wir Ihnen im Folgenden vorstellen. Bei allen Plugins sind dabei die folgenden Parameter möglich:

- `-p / --port`
  Mit diesem Parameter können Sie einen vom Vorgabewert (für IMAP `143` und für POP3: `110`) abweichenden Serviceport angeben.
- `-4 / --use-ipv4` oder `-6 / --use-ipv6`
  Mit dieser Option können Sie festlegen, welche IP-Protokollversion genutzt werden soll.
- `-D / --certificate`
  Mit dieser Option können Sie die Plugins bei Nutzung verschlüsselter Verbindungen auf eine Zertifikatsprüfung umstellen. Übergeben Sie hier die Anzahl der Tage, die das Zertifikat noch gültig sein muss, damit der Test als bestanden gilt.
- `-w / --warning` und `-c / --critical`
  Mit diesen Optionen legen Sie bei Prüfung der E-Mail-Funktion Schwellenwerte für die Status `WARNING` und `CRITICAL` bezüglich der Antwortzeit fest.
- `-t / --timeout`
  Wenn die Antworten eines Servers zu lange dauern, um von den Plugins noch erfasst zu werden, können Sie mit dieser Option eine längere Zeitspanne in Sekunden angeben, die die Plugins auf eine Antwort warten sollen (Vorgabewert: 10).
- `-S / --ssl`
  Mit dieser Option definieren Sie, dass SSL für die Verschlüsselung der Verbindung verwendet werden soll. Beachten Sie, dass Verschlüsselung mittels TLS für diese Plugins nicht unterstützt wird.

Dienstdefinition

Um die voranstehenden Befehlsdefinitionen einzubinden, müssen Sie entsprechende Dienstdefinitionen (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) für bereits angelegte Maschinen (siehe Rezept 4.1, *Maschinen einbinden (host)*) erstellen. Eine Einbindung mit Parametern für Verschlüsselung und eine entsprechende Anpassung der Ports können Sie dabei mit Definitionen wie den folgenden erreichen:

```
define service {
       service_description                    IMAP
       use                                    template-service
       check_command                          check_imap!-p 993 -S
       host_name                              smtp.googlemail.com
}

define service {
       service_description                    POP
       use                                    template-service
       check_command                          check_pop!-p 995 -S
       host_name                              smtp.googlemail.com
}
```

Beachten Sie, dass wir hier eine Vorlage `template-service` referenzieren und die Prüfungen einer bereits konfigurierten Maschine mit dem Namen `smtp.googlemail.com` zuweisen. Passen Sie diese Werte gegebenenfalls an Ihre Umgebung an.

Test, Rückgabewerte und Graphen

Die Funktionen der Plugins können Sie vor dem Einbinden von der Kommandozeile aus testen, für die weiter vorne angeführte, einfache Version der Einbindung von POP3 und IMAP zum Beispiel wie folgt:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_pop -H pop.googlemail.com -p 995 -S
POP OK - 0.054 second response time on port 995 [+OK Gpop ready for requests from
176.9.84.37 mv7pf22372746wib.7]|time=0.054232s;;;0.000000;10.000000

icinga@moni:~$ /usr/local/icinga/libexec/check_imap -H imap.googlemail.com -p 993 -S
IMAP OK - 0.070 second response time on port 993 [* OK Gimap ready for requests from
176.9.84.37 t7if1614664bkh.25]|time=0.069783s;;;0.000000;10.000000
```

Die Plugins geben beim Test der E-Mail-Funktionalität (im Gegensatz zum Test des Zertifikates bei Verschlüsselung) immer Performancedaten zurück. Wenn Sie PNP4Nagios einsetzen, erhalten Sie dadurch automatisch Graphen. Abbildung 6-12 zeigt den Graphen mit den Antwortzeiten bei `check_pop`, den Sie gegebenenfalls automatisch erhalten.

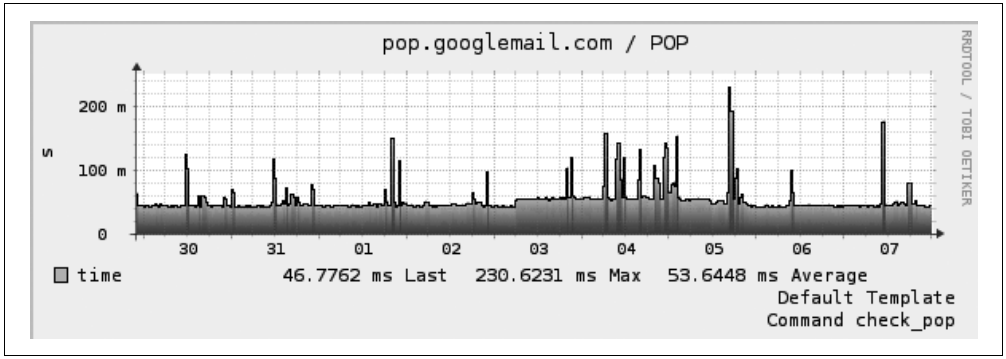


Abbildung 6-12: PNP4Nagios-Graph für die Antwortzeiten bei `check_smtp`

Diskussion

Die Plugins sind sehr nützlich, um schnell Tests von E-Mail-Servern unter Nutzung üblicher Protokolle zu implementieren. Sie können damit gegebenenfalls auch externe Server überwachen, um deren Erreichbarkeit zu dokumentieren und möglicherweise Problemmeldungen dazu besser nachvollziehen zu können. In Rezept 8.6, *E-Mail-Server überwachen* zeigen wir Ihnen weitere Beispiele zum Einsatz dieser Plugins.

Siehe auch

- Rezept zur Überwachung beliebiger TCP/UDP-Ports: Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Einbindung einer Maschine: Rezept 4.1, *Maschinen einbinden (host)*
- Überwachung eines E-Mail-Servers: Rezept 8.6, *E-Mail-Server überwachen*

6.15 Verzeichnisdienst LDAP überwachen (check_ldap)

Problem

Sie möchten einen LDAP-Server überwachen. Dazu soll der Verzeichnisdienst LDAP (Lightweight Directory Access Protocol) über eine zu spezifizierende Suchabfrage getestet werden.

Lösung

In den Standard-Plugins ist zu diesem Zweck `check_ldap` enthalten. Wir zeigen Ihnen im Folgenden, wie Sie dieses einbinden und nutzen können.

Befehlsdefinition und notwendige Optionen

Für eine Nutzung müssen Sie das Plugin zunächst mit einer Befehlsdefinition einbinden. Sie können hierzu die folgende, Definition nutzen (wegen der Option `-3` hier also für LDAP Version 3):

```
# check_ldap
define command {
       command_namd                    check_ldap3
       command_line                    $USER1$/check_ldap -3 -H $HOSTADDRESS$ \
                                       -b $ARG1$
}
```

Sie geben damit die Nutzung der Pflichtparameter vor und nutzen Makros (zu Befehlsdefinitionen und Makros siehe Rezept 4.3, *Befehle hinzufügen (command)*) für den Pfad und die Optionen. Die notwendigen Optionen sind folgende:

- `-H / --hostname`

  Mit diesem Parameter legen Sie die Adresse der abzufragenden Maschine fest. In der Definition wird diese über das Makro `$HOSTADDRESS$` zur Laufzeit aus der verknüpften Maschinendefinintion gesetzt werden.

- `-b / --base`

  Hier geben Sie das oberste Element (Base Domain Name) des zu nutzenden LDAP-Verzeichnisses an. Dies kann eine Zeichenkette wie beispielsweise `b="ou=people,dc=oreillys,dc=de"` sein.

Weitere Optionen

Die Arbeitsweise des Plugins können Sie darüber hinaus mit den folgenden Optionen an Ihre Bedürfnisse anpassen:

- `-S / --ssl` beziehungsweise `-T / --starttls`

  Mit diesen Optionen können Sie die zu verwendende Verschlüsselung angeben, wobei `-S` für SSL (LDAPS aus LDAPv2) und `-T` für STARTTLS steht. Bei der Nutzung von `-S` wird der Port automatisch auf 636 gesetzt.

- `-p / --port`

  Sofern Sie nicht den Standardport `389` verwenden, können Sie mit dieser Option manuell angeben, an welchen Port sich das Plugin verbinden soll.

- `-4 / --use-ipv4` oder `-6 / --use-ipv6`

  Mit diesen Optionen können Sie manuell die zu nutzende Version des IP-Protokolls festlegen.

- `-2 / --ver2` oder `-3 / --ver3`

  Mit dieser Option legen Sie die zu nutzende LDAP-Version fest. Der Vorgabewert ist Version 2. In OpenLDAP ist dies standardmäßig LDAPv3.

- `-D / --bind`

  An dieser Stelle können Sie eine X.500-Verzeichnisobjekt angeben, welches an das LDAP-Verzeichnis gebunden ist (Bind-DN). Dieses benötigen Sie für das Testen

einer authentifizierten Verbindung (sonst ist der Test anonym). Das Verzeichnisobjekt könnte beispielsweise so aussehen: `-D "uid=john,ou=people,dc=hcksp,dc=de"`.

- `-P / --pass`

  Wenn Ihr LDAP-Server ein Passwort verlangt, können Sie das zu nutzende Passwort hier angeben. Diese Option ist nur in Verbindung mit dem Parameter -D sinnvoll und könnte kombiniert dann wie folgt aussehen: `-D "uid=john,ou=people,dc=hcksp, dc=de" -P "PASSPHRASE"`.

- `-w / --warning` und `-c / --critical`

  Wenn Sie die Antwortzeiten vom Plugin gegen Schwellenwerte testen lassen möchten, geben Sie sie mit dieser Option in Sekunden an.

- `-t / --timeout`

  Wenn Sie den Timeout auf einen vom Vorgabewert 10 Sekunden abweichenden Wert setzen möchten, so nutzen Sie dazu diese Option.

Dienstdefinition

Um die weiter vorne angeführte Befehlsdefinition zu verwenden, benötigen Sie dann außerdem eine entsprechende Dienstdefinition (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) für eine bereits angelegte Maschine (siehe Rezept 4.1, *Maschinen einbinden (host)*). Für eine einfache Abfrage basierend auf der voranstehenden Kommandodefinition könnte diese folgendermaßen aussehen:

```
define service{
       service_description                     LDAPv3
       use                                     template-service-vms
       check_command                           check_ldap3!ou=people,dc=hcksp0,dc=de
       host_name                               eldub
}
```

Beachten Sie, dass wir hier eine Vorlage `template-service` verwenden und die Prüfung einer entsprechend konfigurierten Maschine `eldub` zuweisen. Passen Sie diese Werte gegebenenfalls Ihrer Umgebung an.

Test, Rückgabewerte und Graphen

Bevor Sie einen entsprechenden Dienst einbinden, sollten Sie die Ausführung des Plugins von der Kommandozeile testen.

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ldap -3 -H 10.85.58.44
-b "ou=people,dc=hcksp0,dc=de"
LDAP OK - 0.002 seconds response time|time=0.001721s;;;0.000000
```

Das Plugin gibt dabei Performancedaten zu den ermittelten Abfragezeiten zurück. Wenn Sie PNP4Nagios verwenden, erhalten Sie deshalb automatisch entsprechende Graphen, wie beispielsweise den in Abbildung 6-13.

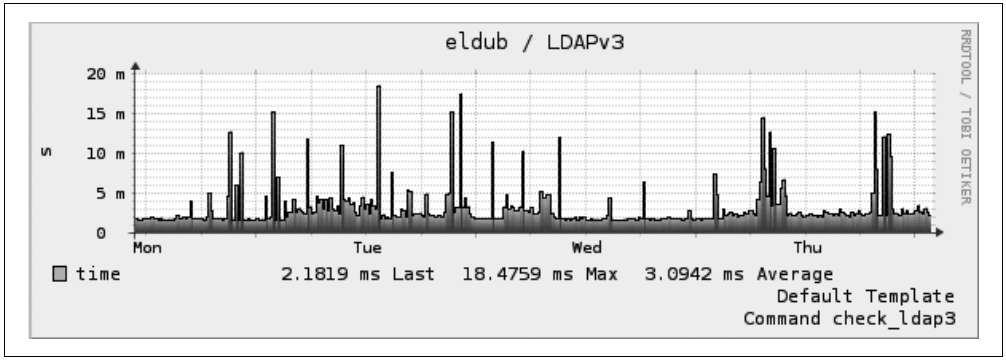


Abbildung 6-13: PNP4Nagios Graph für check_ldap

Diskussion

Das Plugin `check_ldap` ermöglicht Ihnen die einfache Abfrage von LDAP-Servern, auch bei Nutzung von Verschlüsselung und Authentifizierung. Solche weitergehende Nutzung und die entsprechenden Optionen zeigen wir Ihnen im Rezept 8.9, *Zentrale Netzwerk-Dienste überwachen (DNS, DHCP, LDAP, AD)* anhand weiterer Beispiele.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Einbindung einer Maschine: Rezept 4.1, *Maschinen einbinden (host)*
- Rezept zur Überwachung eines Verzeichnisdienstes: Rezept 8.9, *Zentrale Netzwerk-Dienste überwachen (DNS, DHCP, LDAP, AD)*

6.16 SMB-Dateidienst überwachen (check_smb)

Problem

Sie wollen einen SMB-Dienst überwachen. Dabei wollen Sie die Verfügbarkeit der Dateifreigaben prüfen.

Lösung

Die Sammlung von Standard-Plugins enthält mit `check_smb` ein Plugin, das genau für diesen Zweck gedacht ist. Für das Plugin spielt es keine Rolle, ob es sich bei dem SMB-

Dienst um eine Dateifreigabe auf einem Windows-Server oder auf einem SAMBA-Server handelt. Wir zeigen Ihnen im Folgenden, wie Sie dieses einbinden und nutzen können. Das Plugin fragt über das Netzwerk eine spezifizierte SMB-Freigabe ab und gibt bei Erfolg den freien Speicherplatz der Freigabe zurück.

Plugin-Optionen

Das Plugin bietet Ihnen die folgenden Optionen für Anpassungen an Ihre Umgebung:

- `-H / --hostname`

  Hier geben Sie abzufragende Maschine an. Sie können anstelle von Internet-Adressen auch den NetBIOS-Namen verwenden.

- `-P / --port`

  Das Plugin verwendet als Vorgabewert die von `smbclient` verwendeten Ports (139 oder 445). Mit dieser Option können Sie einen abweichenden Port angeben und verwenden.

- `-s / --share`

  Hier geben Sie den Namen der Freigabe an, auf die zugegriffen werden soll.

- `-W / --workgroup`

  Die Arbeitsgruppe ist als Vorgabewert `WORKGROUP`. Mit dieser Option können Sie für den Test eine andere Arbeitsgruppe angeben und nutzen.

- `-u / --user`

  Das Plugin nutzt als Vorgabewert den Benutzer `guest`. Wenn Sie die Abfrage unter einem anderen Benutzer durchführen möchten, geben Sie mit dieser Option den entsprechenden Benutzernamen an.

- `-p / --password`

  Mit dieser Option geben Sie das Passwort für den verwendeten Benutzernamen an.

- `-w / --warning und -c / --critical`

  Das Plugin setzt die Status `WARNING` und `CRITICAL` basierend auf dem freien Speicherplatz der Freigabe. Es verwendet hierzu Vorgabewerte für die Warnschwellen von 85% für `WARNING` und 95% für `CRITICAL`. Sie können mit diesen Optionen abweichende Warnschwellen angeben. Wenn Sie statt % die Einheiten k (Kilo), M (Mega) oder G (Giga) verwenden, prüft das Plugin auf freien Speicherplatz (dann muss also `WARNING` > `CRITICAL` sein).

Damit bietet das Plugin Optionen, um die bei SAMBA typischerweise genutzten Parameter zu beeinflussen.

Kommandozeilen-Test

Bevor Sie das Plugn einbinden, testen Sie die von Ihnen gewählten Optionen zunächst von der Kommandozeile aus. Dazu rufen Sie das Plugin unter manueller Angabe aller Optionen und Werte auf:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
Myrd:~ # /usr/lib/nagios/plugins/check_disk_smb -H 192.168.0.2
--workgroup=GRUPPE --share=FREIGABE --user=BENUTZER --PASSWD=123
Domain=[GRUPPE] OS=[Unix] Server=[Samba 3.2.7-11.8.1-2426-SUSE-CODE11]
Disk ok - 273.5G (69%) free on \\192.168.0.2\FREIGABE
```

Wir haben hier eine Maschine unter der Adresse `192.168.0.2` abgefragt und dabei die Freigabe `FREIGABE` in der Arbeitsgruppe `GRUPPE` mit dem Benutzerkonto `BENUTZER` und dem PASSWD `123` verwendet. Denken Sie daran diese Werte an Ihre Umgebung anzupassen.

Kommando-Definition

Sofern der Aufruf von der Kommandozeile funktioniert hat, können Sie das Einbindung mit der Kommando-Definition beginnen:

```
# SMB-Freigabe
define command{
       command_name                          check_smb
       command_line                          $USER1$/check_disk_smb -H $HOSTADDRESS$\
--workgroup=$ARG1$ --user=$ARG2$ --password=$ARG3$ --share=$ARG4$ $ARG5$
}
```

Diese Definition verwendet das Makro `$USER1$` (siehe dazu das Rezept zu Kommandodefinitionen und Makros Rezept 4.3, *Befehle hinzufügen (command)*), um den Pfad zu dem Plugin einzusetzen und nutzt über `$HOSTADDRESS$` die Adresse, die in der später referenzierten Maschine angegeben ist. Über `$ARG1$` – `$ARG4$` werden die vier erforderlichen Parameter in der Service-Definition erwartet.

CRITICAL SMB anon access: Wenn Sie diese Fehlermeldung erhalten, dann ersetzen Sie das Makro `$HOSTADDRESS$` durch den voll qualifizierten Domänennamen beziehungsweise das Makro `$HOSTNAME$`, oder verwenden Sie gegebenenfalls den NetBIOS-Namen. Seit Windows Server 2008 wird hier die IP-Adresse nicht mehr akzeptiert.

Service-Definition

Aufbauend auf der obigen Kommando-Definition können Sie dann beginnen entsprechende Dienste zu erstellen. Legen Sie sich hierzu entsprechende Dienstdefinitionen an (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*):

```
define service{
        service_description                     SMB
        use                                     template-service
        check_command                           check_smb!gruppe!benutzer!123!freigabe
        host_name                               file-serv
}
```

Beachten Sie dabei, dass Sie die referenzierte Vorlage `template-service`, die Daten für die Freigabe sowie die Referenz auf die zu prüfenden Maschinen an Ihre Umgebung anpassen müssen. Sie benötigen dann weiter eine Service-Definition für jede Freigabe, die Sie prüfen möchten. Wenn Sie identische Freigaben und Benutzer auf mehreren Maschinen vorhalten, können Sie diese alle mit einer Definition testen, indem Sie die weiteren Maschinen durch Kommata separiert angeben.

Performance-Daten und Graphen

Das Plugin gibt keine Performance-Daten zurück, deshalb erhalten Sie zunächst auch keinen Graphen. Sie könnten sich bei Bedarf allerdings mit einem Wrapper-Script (siehe Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*) behelfen.

Diskussion

Das Plugin ermöglicht Ihnen wie beschrieben die Erreichbarkeit von SAMBA-Freigaben über das Netzwerk zu testen. Sie können auf diese Weise sicherstellen, dass eine Freigabe von dem Dienst wie gewünscht angeboten wird. Die zurückgegebenen Informationen zum Füllstand sind häufig nicht besonders interessant, da Sie diese in aller Regel direkt auf dem Server ermitteln werden.

Wenn Sie von ihrem überwachenden Rechner aus keinen direkten Zugang zum lokalen Netzwerk haben, in dem der Dienst angeboten wird, können Sie das Plugin so zunächst nicht verwenden. Sie können sich hier aber behelfen, indem Sie das Plugin per SNMP (siehe Rezept 2.5, *Plugins unter Linux über SNMP ausführen* und Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*), SSH (siehe Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*) oder NRPE (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*) auf einer Maschine in dem Netzwerk einbinden, von dem der Dienst erreichbar ist, um den Dienst dennoch zu überwachen.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Erstellung eines Wrapper-Scripts für die Erstellung von Performance-Daten: Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*

- Rezepte zur Remote-Ausführung von lokalen Plugins über SNMP: Rezept 2.5, *Plugins unter Linux über SNMP ausführen* und Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*
- Rezept zur Remote-Ausführung von lokalen Plugins über SSH: Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*
- Rezept zur Remote-Ausführung von lokalen Plugins über NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*

6.17 MySQL-Datenbanken überwachen (check_mysql)

Problem

Sie wollen einen MySQL-Server überwachen. Sie möchten die Funktionalität von MySQL (My Structured Query Language) mit Hilfe einer zu spezifizierenden Suchabfrage testen.

Lösung

Im Paket der Standard-Plugins sind zu diesem Zweck die Plugins `check_mysql` und `check_mysql_query` enthalten. Wir zeigen Ihnen im Folgenden, wie Sie diese Plugins einbinden und nutzen können.

Befehlsdefinition und notwendige Optionen

Für das Einbinden benötigen Sie zunächst eine entsprechende Kommando-Definition (siehe Rezept 4.3, *Befehle hinzufügen (command)*). Da Sie in der Regel einen Benutzernamen und ein Passwort verwenden sollten, empfehlen wir Ihnen die Hinterlegung des Passwortes als benutzerdefiniertes Makro. Fügen Sie Ihrer `resource.cfg` (bei unserer Referenzinstallation unter */usr/local/icinga/etc/resource.cfg*) hierfür einen Eintrag wie den folgenden hinzu:

```
[...]
# credentials for mysql on db-server
$USER10$="passwort"
[...]
```

Verwenden Sie als Kommando-Definition dann etwa die Folgende, um das so definierte Passwort direkt über das Kommando zu referenzieren:

```
# check mysql on host db
define command
{
       command_name                          check_db_mysql
       command_line                          $USER1$/check_mysql -u sql_monitor \
-p $USER10$ -H $HOSTADDRESS$ $ARG1$
}
```

```
define command
{
       command_name                            check_db_mysql_query
       command_line                            $USER1$/check_mysql_query \
-u sql_monitor -p $USER10$ -H $HOSTADDRESS$ -q $ARG1$ $ARG2$
}
```

Wir verwenden hierbei das MySQL-Benutzerkonto `sql_monitor`. Geben Sie an dieser Stelle den bei Ihnen zum hinterlegten Passwort passenden Benutzernamen an. Das Passwort selbst wird dann über das Benutzer-Makro eingesetzt. Passen Sie hier gegebenenfalls die Nummer des Makros an. Die zu überwachende Maschine wird jeweils über das Makro `$HOSTADDRESS$` automatisch übernommen. Bei der Definition für `check_mysql` werden weitere Optionen über `$ARG1$` verwendet, während bei `check_mysql_query` `$ARG1$` die zu nutzende Abfrage ist und weitere Optionen über `$ARG2$` übernommen werden (zu Makros siehe Rezept 4.3, *Befehle hinzufügen (command)*).

Auf diese Weise haben Sie für eine bestimmte Maschine generische Definitionen, die bereits die Zugangsdaten enthalten. Sie können darauf aufbauend dann leicht unterschiedliche Services definieren (siehe im Nachfolgenden).

Die beiden Plugins unterscheiden sich hinsichtlich der Optionen in genau einer Option, auf die wir nachfolgend eingehen werden. Die für beide Plugins gemeinsamen Optionen sind folgende:

- `-s / --socket`

  Um einen spezifischen Unix-Socket auf der lokalen Maschine anzugeben, können Sie diese Option verwenden. Sofern dieser Socket bei Ihnen nicht spezifisch angepasst ist, können Sie die Plugins an dieser Stelle auch ohne die Optionen `-s`, `-H` und `-P` aufrufen. Die Plugins verwenden dann automatisch die lokale Maschine und den Standard-Socket.

- `-H / --hostname`

  Um eine Maschine über das Netzwerk abzufragen, geben Sie mit dieser Option die abzufragende Maschine an. Diese Option überschreibt gegebenenfalls die Option `-s`.

- `-P / --port`

  Wenn Sie eine Abfrage über das Netzwerk durchführen, können Sie mit dieser Option einen vom Vorgabewert 3306 abweichenden Port angeben.

- `-u / --username`

  Mit dieser Option geben Sie den MySQL-Benutzernamen an, den Sie für die Abfrage verwenden möchten.

- `-p / --password`

  Wenn Sie einen Benutzernamen mit -u angeben, müssen Sie mit dieser Option auch ein Passwort angeben. Bei leeren Passwörtern übergeben Sie hier "".

Spezifische Optionen für *check_mysql*:

- `-d / --database`

  Mit dieser Option können Sie eine abzufragende Datenbank angeben. Ohne diese Angabe wird `check_mysql` den Datebankserver allgemein überprüfen.

- `-S / --check-slave`

  Wenn Sie den MySQL-Dienst mit Replikation verwenden, können Sie **`check_mysql`** mit dieser Option veranlassen, zu prüfen, ob der Replikations-Client läuft. Bei Nutzung dieser Option übergeben Sie dann zusätzlich die Warnwerte für die Verzögerung des Replikationsdienstes mit den folgenden beiden Optionen:

  – `-w / --warning`

    Anzahl von Sekunden, ab der der Status auf `WARNING` gesetzt werden soll.

  – `-c / --critical`

    Anzahl von Sekunden, ab der der Status auf `CRITICAL` gesetzt werden soll.

Spezifische Optionen für *check_mysql_query*:

- `-d / --database`

  Mit dieser Option können Sie die abzufragende Datenbank festlegen. Alternativ können Sie diese Option bei `check_mysql_query` weglassen und die Datenbank über die Abfrage in der Option **`-q`** spezifizieren.

- `-q / --query`

  Mit dieser Option übergeben Sie `check_mysql_query` die durchzuführende MySQL-Abfrage. Beachten Sie, dass die Abfrage einen nummerischen Wert zurückgeben muss. Bei Nutzung dieser Option können Sie dann zusätzlich Warnwerte mit den folgenden beiden Optionen angeben:

  – `-w / --warning`

    Wert, ab dem der Status auf `WARNING` gesetzt werden soll.

  – `-c / --critical`

    Wert, ab dem der Status auf `CRITICAL` gesetzt werden soll.

Dienstdefinition

Um die voranstehenden Kommandodefinitionen zu nutzen, können Sie nun Service-Definitionen (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) wie beispielsweise die folgenden verwenden:

```
define service{
       service_description                    MySQL
       use                                    template-service-vms
       check_command                          check_mysql_on_db
       host_name                              db
}
```

```
define service{
       service_description                    db-docu
       use                                    template-service-vms
       check_command                          check_mysql_on_db!-d drupal_docu
       host_name                              db
}

define service{
       service_description                    db-docu-sessions
       use                                    template-service-vms
       check_command                          check_mysql_query_on_db!"SELECT \
COUNT(\*) FROM drupal_docu.sessions"
       host_name                              db
}
```

Das erste Beispiel fragt ab, ob der MySQL-Server auf der Maschine `db` überhaupt läuft. Das zweite Beispiel prüft ob darauf eine Datenbank `drupal_docu` vorhanden ist. Das dritte Beispiel führt eine Abfrage auf dieser Datenbank aus und ermittelt, wie viele Sitzungen in diesem System momentan laufen. Beachten Sie zunächst, dass Sie vermutlich die Namen für Maschine und Datenbank an Ihre Umgebung anpassen müssen.

Escapes: Beachten Sie bei `check_mysql_query` auch, dass Sie in der Konfiguration die Verwendung von Sonderzeichen wie hier * unter Nutzung von \ escapen müssen. Dies ist beim Test von der Kommandozeile nicht der Fall, was eine Quelle für mögliche Fehler bedeuten kann, die dann unter Umständen schwer zu identifizieren sind.

Test, Rückgabewerte und Graphen

Um die Prüfungen wie in den Service-Definitionen vorgeschlagen zu testen, sollten Sie entsprechende Aufrufe auf der Kommandozeile durchführen, bevor Sie die tatsächliche Einbindung und den erforderlichen Neustart des Monitoring-Systems vornehmen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec/check_mysql -H db -u sql_monitor -p "PASSWD"
Uptime: 4483800 Threads: 3 Questions: 32158954 Slow queries: 8 Opens: 53154 Flush tables:
 1 Open tables: 64 Queries per second avg: 7.172

icinga@moni:~$ /usr/local/icinga/libexec/check_mysql -H db -u sql_monitor -p "PASSWD"
-d drupal_docu
Uptime: 4483785 Threads: 5 Questions: 32158767 Slow queries: 8 Opens: 53154 Flush tables:
 1 Open tables: 64 Queries per second avg: 7.172

icinga@moni:~$ /usr/local/icinga/libexec/check_mysql_query -H db -u sql_monitor
-p "PASSWD" -q "SELECT COUNT(*) FROM drupal_book.sessions;"
QUERY OK: 'SELECT COUNT(*) FROM drupal_book.sessions;' returned 2.000000
```

Beachten Sie, dass die Plugins keine Performancedaten zurückgeben. Sie können sich gegebenenfalls wie in Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* beschrieben eine entsprechende Umwandlungsmöglichkeit bauen. Hier ein Beispiel für ein Wrapper-Script für Abfragen mit `check_mysql_query` auf einer bestimmten Maschine, für die die Zugangsdaten im Script selbst hinterlegt sind:

```
#!/bin/bash

# Auslesen der Parameter und setzen entsprechender Variablen HOST_NAME und QUERY
QUERY=""
for PARAM in "$@";
do
          if ( [ "$HOST_NAME" == "" ] );
          then
          HOST_NAME=$PARAM
          else
          QUERY="$QUERY $PARAM"
          fi
done

# Ausführen des Befehls und speichern der textuellen Rückgabe in der Variable LINE
LINE=`/usr/local/icinga/libexec/check_mysql_query -H $HOST_NAME -u sql_monitor
-p "PASSWD" -q "$QUERY"`

# Speicherung des nummerischen Rückgabewerts in der Variable RC
RC=$?

# Ermittlung des Zahlenwerts und Speicherung in der Variable COUNT
COUNT=`echo $LINE | egrep -o "[0-9]{1,9}\.[0-9]{1,9}$"`

# Ausgabe der Rückgabe ergänzt um Performancedaten
echo $LINE \| count=$COUNT

# Rückgabe des erhaltenen nummerischen Rückgabewertes
exit $RC
```

Wenn Sie viele Datenbankserver mit unterschiedlichen Zugangsdaten abfragen möchten, sollten Sie das Script eventuell erweitern, so dass Benutzername und Passwort ebenfalls als Argumente übergeben und ausgewertet werden. Um es als `wrapper check mysql query.sh` einzubinden, würden Sie dann abweichend vom voranstehenden Beispiel etwa die folgende Definition verwenden:

```
# wrapper für check_mysql_query
define command {
       command_name                    wrapper_check_mysql_query
       command_line                    $USER2$/wrapper_check_mysql_query.sh \
$HOSTADDRESS$ -q $ARG1$ $ARG2$
}
```

Zum Einbinden für zwei verschiedene Datenbanken können Sie dann etwa die folgenden Service-Definitionen verwenden:

```
define service{
       service_description             db-docu-sessions
       use                             template-service-vms
```

```
        check_command                   wrapper_check_mysql_query!"SELECT \
COUNT(\*) FROM drupal_docu.sessions"
        host_name                       db
}

define service{
        service_description             db-docu-nodes
        use                             template-service-vms
        check_command                   wrapper_check_mysql_query!"SELECT \
COUNT(\*) FROM drupal_docu.node"
        host_name                       db
}
```

Bevor Sie diese Einbindung vornehmen, sollten Sie sie natürlich auf der Kommandozeile etwa folgendermaßen testen:

```
icinga@moni:~$ /usr/local/icinga/libexec2/wrapper_check_mysql_query.sh db "SELECT COUNT(*)
FROM drupal_docu.node;"
QUERY OK: ' SELECT COUNT(*) FROM drupal_docu.node;' returned 124.000000 | count=124.000000
```

Damit erhalten Sie dann auch für Ihre MySQL-Abfragen Graphen. Abbildung 6-14 zeigt Ihnen zur Veranschaulichung den Graphen zum angeführten Beispiel der aktiven Sessions.

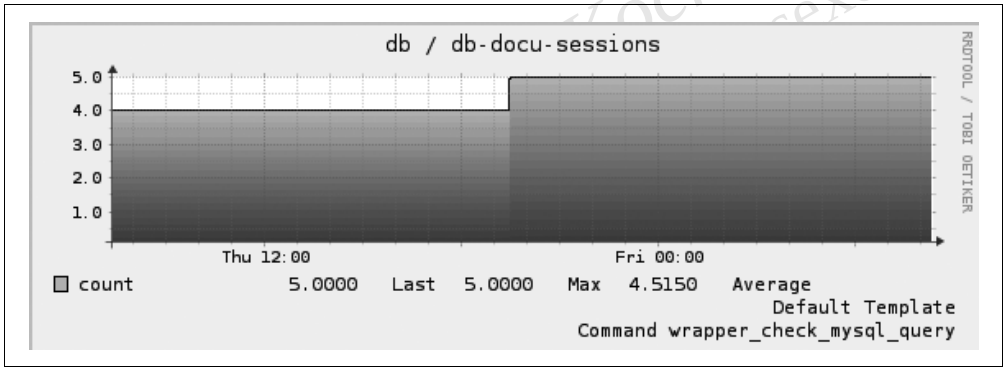


Abbildung 6-14: PNP4Nagios-Graph über Wrapper zu check_mysql_query

Diskussion

Das Plugin `check_mysql` ermöglicht Ihnen zunächst einen einfachen Check, ob der Datenbankserver unter MySQL überhaupt läuft beziehungsweise ob eine bestimmte Datenbank darin vorhanden ist. Mit Hilfe von `check_mysql_query` können Sie darüber hinaus gezielt Abfragen auf nummerische Werte durchführen und auswerten, wozu jedoch Kenntnisse über den Aufbau der Datenbanken erforderlich sind. Ein externes Plugin, mit dem Sie umfangreiche Tests an einer Datenbank durchführen können, ist das Plugin `check_mysql_health` (siehe *http://labs.consol.de/lang/de/nagios/check_mysql_health/*).

Oracle & Postgres: Um Datenbanken von Oracle zu überwachen, können Sie das externe Plugin `check_oracle_health` (siehe *http://labs.consol.de/lang/de/nagios/check_oracle_health/*) verwenden. Für die Überwachung von Postgres können Sie das Plugin `check_postgres` (siehe *http://bucardo.org/wiki/Check_postgres*) einsetzen.

Wir zeigen Ihnen in Rezept 8.8, *Datenbankserver überwachen* weitere Beispiele für entsprechende Überprüfungen. Um auf Basis dieser Plugins Graphen zu generieren, müssten Sie sich wie in Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* erläutert, entsprechende Konvertierungsskripte schreiben, da die Plugins von Hause aus leider keine Performancedaten ausgeben.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zu Dienstdefinitionen: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezepte zum Erstellen eines Wrapper-Scripts: Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*
- Überwachung eines Datenbankservers: Rezept 8.8, *Datenbankserver überwachen*

KAPITEL 7

Erstellung eigener und Einbindung externer Plugins

In Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen* und Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen* haben wir Ihnen die Standard-Plugins von Nagios/Icinga vorgestellt. Diese ermöglichen Ihnen bereits, eine große Anzahl von Diensten zu testen. Typischerweise werden Sie allerdings feststellen, dass Sie einige Tests lieber etwas anders durchführen möchten oder zusätzliche Tests benötigen. In diesem Kapitel stellen wir Ihnen Möglichkeiten vor, mit denen sich der bereits beschriebene Funktionsumfang erweitern lässt.

Nagios/Icinga bindet die Plugins über ein definiertes API (Application Programming Interface) ein. Diese Schnittstelle wird von allen Plugins genutzt und steht Ihnen für Erweiterungen zur Verfügung. In Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)* stellen wir Ihnen dieses API aufgrund seiner Bedeutung zunächst detailliert vor, um darauf aufbauend auf dessen Nutzung einzugehen.

Als erstes Beispiel für die Anwendung stellen wir Ihnen zunächst ein eigenes Plugin in Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* in Form eines einfachen Skriptes vor. Mit diesem können Sie ein bereits bestehendes Plugin aufrufen und seine Ausgabe entsprechend des API erweitern, um seinen Nutzwert zu steigern. In einem weiteren Schritt zeigen wir Ihnen in Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*, wie Sie gleich mehrere Ergebnisse von vorhandenen Plugins abrufen und zu einem einzelnen zusammenfassen können.

Die Möglichkeit, Nagios/Icinga zu erweitern, haben auch viele andere Anwender genutzt. Die entsprechenden Ergebnisse werden Ihnen zum Teil über das Internet zur Verfügung gestellt. Es gibt hierzu neben einzelnen, verstreuten Webseiten auch Verzeichnisse, in denen Ihnen eine große Anzahl an Plugins für Erweiterungen zur Verfügung stehen:

- `monitoringexchange.org`

  Ein Verzeichnis, das spezifisch sowohl Nagios als auch Icinga adressiert, finden Sie unter *https://www.monitoringexchange.org/inventory/Check-Plugins*.

- `exchange.nagios.org`

  Alternativ können Sie auch auf das Nagios-eigene, aber auch für Icinga verwendbare Verzeichnis *http://exchange.nagios.org/directory/Plugins* zurückgreifen.

Bei den momentanen und in diesem Buch vorgestellten Versionen können Sie für Plugins beider Verzeichnisse verwenden. Um Ihnen den Prozess der Einbindung von solchen externen Plugins zu vermitteln, zeigen wir Ihnen in Rezept 7.4, *Einbinden des PNP4Nagios-Plugins (check_pnp_rrds)* zunächst, wie Sie das bei PNP4Nagios mitgelieferte Plugin zur Überwachung desselben einbinden. In Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins* zeigen wir Ihnen dann anhand einer Sammlung von Plugins für SNMP-Abfragen (die sogenannten Manubulon-Plugins), wie Sie selbst-beschaffte Plugins einbinden.

Insgesamt wird Ihnen dieses Kapitel also das Basiswissen vermitteln, um auf Tausende vorhandener Plugins zurückgreifen oder sich gegebenenfalls eigene Plugins erstellen zu können. Für die fortgeschrittene Nutzung ist dies elementar, da Sie trotz des erheblichen Umfangs der mitgelieferten Plugins vermutlich nicht alle Ihre Wünsche mit diesen abdecken können.

7.1 Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)

Problem

Sie möchten verstehen, wie an Nagios/Icinga Daten von Plugins übergeben werden. Dies ist die Grundlage, um später eigene Plugins erstellen zu können.

Lösung

Nagios/Icinga hat eine Programmierschnittstelle (Application Programming Interface beziehungsweise kurz API) zur Anbindung von Plugins. Prinzipiell kann jedes Programm mit entsprechenden Rückgaben als Plugin genutzt werden. Wir stellen Ihnen im Folgenden dieses API vor.

Status als Rückgabewert (Pflicht)

Der elementare Teil des API ist die Definition der numerischen Rückgabewerte eines Plugins als Status. Die Status haben eine leicht unterschiedliche Bedeutung, je nachdem, ob der Befehl für eine Maschine (`host-check`, siehe hierzu Rezept 4.1, *Maschinen einbinden (host)*) oder einen Dienst (`service-check`, siehe hierzu Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) durchgeführt wird. Die möglichen Status können Sie Tabelle 7-1 entnehmen.

Tabelle 7-1: Rückgabewert der API

Rückgabewert	Interpretation bei Maschine	Interpretation bei Service
0	UP	OK
1	DOWN	WARNING
2	UNREACHABLE	CRITICAL
3		UNKNOWN

Ein Programm, das Sie als Plugin in Nagios/Icinga verwenden möchten, muss diese Status-Codes geeignet setzen können und zurückgeben. Sie können sich hiervon einen Eindruck verschaffen, indem Sie das in den Standardplugins enthaltene `check_dummy##` mit unterschiedlichen Werten aufrufen. Dieses Plugin macht zunächst nichts anderes, als Ihnen eine dem übergebenen Code entsprechende Statusnachricht zurückzugeben, also bei den gültigen Statuscodes etwa Folgende:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_dummy 0
OK
icinga@moni:~$ /usr/local/icinga/libexec/check_dummy 1
WARNING
icinga@moni:~$ /usr/local/icinga/libexec/check_dummy 2
CRITICAL
icinga@moni:~$ /usr/local/icinga/libexec/check_dummy 3
UNKNOWN
```

Nagios/Icinga kann ausschließlich die genannten Rückgabewerte zuordnen, andere verursachen bei der Ausführung des Plugins einen Fehler vom Typ `out of bounds`. Wenn Sie beim Aufruf einen ungültigen Statuscode verwenden, wird das Plugin (wie Nagios/Icinga) daraus immer den Status `UNKNOWN` machen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_dummy -1
UNKNOWN: Status -1 is not a supported error state
```

Welche Funktion Ihr Plugin auch ausführt, es muss am Ende einen entsprechenden Rückgabecode für Nagios/Icinga generieren. Diesen verwendet Nagios/Icinga dann, um den grundsätzlichen Status anzuzeigen. In der Weboberfläche (siehe hierzu Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*) führt dieser Status dann zunächst zu den entsprechenden Kategorisierungen und Hervorhebungen. In Abhängigkeit dieser Status wird Nagios/Icinga dann gegebenenfalls auch Benachrichtigungen versenden (siehe hierzu Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*).

Text als beschreibende Rückgabe (Pflicht)

Neben dem Statuscode erwartet Nagios/Icinga eine kurze textuelle Rückgabe, die beschreiben sollte, auf welcher Grundlage Ihr Plugin den Status gesetzt hat. Dies ist der einzeilige Text, der Ihnen in der Weboberfläche in den Übersichten angezeigt wird, wie beispielsweise in Abbildung 7-1 ganz rechts.

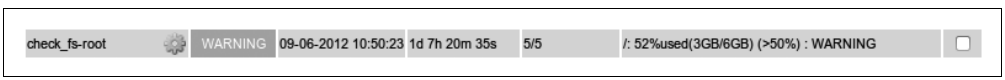


Abbildung 7-1: Bildschirmfoto: Beispiel der Darstellung eines Services in der Service-Übersicht

Auch diese Funktion können Sie sich zunächst am Beispiel des Plugins `check_dummy` verdeutlichen, indem Sie ihm zusätzlich zu einem Statuscode einen Text übergeben:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_dummy 0 text
OK: text
```

In einem Plugin sollte hier jeweils das Ergebnis der entsprechenden Prüfung zusammenfasst werden. Dies sind dann auch schon alle Pflichtbestandteile, um die API verwenden zu können. Die weiteren optionalen Möglichkeiten zur Datenübergabe werden wir Ihnen im Folgenden vorstellen.

Performancedaten für Grapher (optional)

Sofern Ihr Plugin Zahlenwerte verarbeitet, können und sollten Sie überlegen, zusätzlich sogenannte Performancedaten auszugeben. Als Trennzeichen zum Plugin-Rückgabetext wird dsa Pipe-Symbol »|« verwendet. Performancedaten beschreiben eine strukturierte Angabe eines Zahlenwertes sowie möglicherweise dazugehörige Metadaten und bestehen aus folgenden Angaben (die ersten beiden Angaben sind obligatorisch):

- einer Bezeichnung (engl. Label)
- dem Zahlenwert
- der Maßeinheit (UOM – Unit of Measurement) des Wertes
- dem Schwellenwert für den Status `WARNING`
- dem Schwellenwert für den Status `CRITICAL`
- dem Minimalwert des Wertes
- dem Maximalwert des Wertes

Das Format ist dabei wie folgt festgelegt:

```
'label'=value[UOM];[warn];[crit];[min];[max]
```

Die optionalen Werte sind hier in eckigen Klammern einschlossen. Sie können diese im Zweifel weglassen.

Bezeichner-Länge: Die ersten 19 Zeichen der Bezeichnung sollten eindeutig sein und in dieser Folge in keinem anderen Bezeichner vorkommen. Dies hängt mit einer Limitierung des in graphischen Auswertungenhäufig verwendeten RRDtools zusammen (siehe hierzu etwa *http://nagiosplug.sourceforge.net/developer-guidelines.html*).

In der einfachsten Form bestehen Performancedaten also aus dem Bezeichnuer und dem Zahlenwert:

```
'label'=value
```

Dieser kann dann um die weiteren Werte erweitert werden. Beachten Sie dabei, dass alle Werte die gleiche Maßeinheit verwenden müssen. Das voranstehende Beispiel könnte also wie in den folgenden Beispielen erweitert werden:

```
'label'=value %
'label'=value %;70;90
'label'=value %;70;90;0;100
```

In der Oberfläche werden Ihnen diese detaillierteren Daten in allen Übersichten zunächst nicht angezeigt. Dafür müssen Sie die Seite mit den Service-Details aufrufen, auf der Ihnen dann auch die Performance-Ausgabe angezeigt wird. Abbildung 7-2 zeigt Ihnen ein entsprechendes Beispiel.

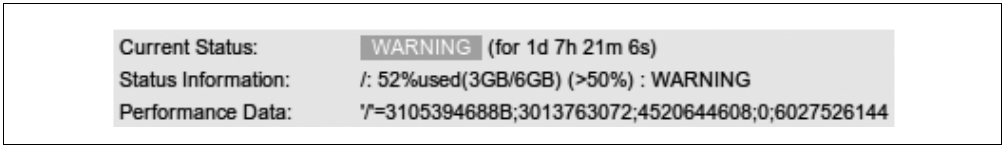


Abbildung 7-2: Bildschirmfoto der Detailansicht eines Services

Erweiterte Ausgaben mit Text/Perfomance-Daten (optional)

Zusätzlich zu den erforderlichen Werten, Rückgabecodes und Kurzbeschreibung können Sie optional noch weitere Zeilen im gleichen Format (also möglicherweise mit Performancedaten) übergeben. In der Weboberfläche werden diese weiteren Zeilen dann – genau wie die Performancedaten – nur in der Detail-Ansicht angezeigt, wie in Abbildung 7-3 dargestellt.

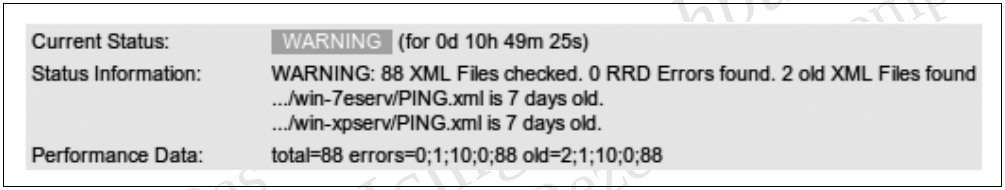


Abbildung 7-3: Bildschirmfoto: Beispiel einer mehrzeiligen Rückgabe in der Detailansicht eines Services

In diesem Beispiel hat das Plugin insgesamt drei Zeilen Text ausgegeben: Die Pflichtangabe und entsprechend zwei weitere Zeilen. Performancedaten aus mehreren Zeilen werden dabei zusammengefasst.

Diskussion

Die Programmierschnittstelle definiert ein relativ einfaches Format, an das sich alle Plugins halten müssen. Für Sie ist dieses Format insbesondere dann von Bedeutung, wenn Sie eigene Plugins schreiben, was wir Ihnen in den folgenden Rezepten erläutern werden. Das Verständnis hilft Ihnen aber häufig auch bei der Diagnose von Problemen mit der Benutzung von Plugins. Unserer Erfahrung nach funktionieren die mitgelieferten Standard-Plugins sehr stabil. Bei der Nutzung von zusätzlichen Plugins sind wir hingegen häufig auf Probleme gestoßen.

Die Entwickler-Richtlinien, die Sie unter *http://nagiosplug.sourceforge.net/developer-guidelines.html* abrufen können, enthalten auch Anweisungen dazu, wie für Plugins Schwellenwerte definiert werden sollen (*http://nagiosplug.sourceforge.net/developer-guidelines.html#THRESHOLDFORMAT*). Diese sind etwas gewöhnungsbedürftig, deshalb sollten Sie sich zumindest oberflächlich mit diesen vertraut machen. Sie werden von vielen Plug-

ins genutzt. Wenn Sie das Verhalten eines Plugins bezüglich der Schwellenwerte nicht verstehen, prüfen Sie, ob das Plugin gemäß der betreffenden Spezifikation arbeitet und ob die Rückgabewerte korrekt umgesetzt wurden.

Siehe auch

- Rezepte zur Einbindung von Maschinen und Diensten: Rezept 4.1, *Maschinen einbinden (host)* und Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezepte zu den Oberflächen: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*
- Rezept zur Anpassung von Benachrichtigungen: Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*
- Seiten mit Entwicklerrichtlinien vom Nagios-Projekt: *http://nagiosplug.sourceforge.net/developer-guidelines.html*
- Dokumentationsseiten zur API: *http://nagiosplug.sourceforge.net/developer-guidelines.html#PLUGOUTPUT* beziehungsweise *http://docs.icinga.org/latest/de/pluginapi. html*
- Beschreibung wie Warnschwellen in Plugins spezifiziert werden sollen: *http://nagiosplug.sourceforge.net/developer-guidelines.html#THRESHOLDFORMAT*
- Rezept zur Einbindung weiterer Plugins am Beispiel der Manubulon SNMP-Plugins: Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*

7.2 Ein einfaches Beispiel eines benutzerdefinierten Plugins

Problem

Sie möchten zu den Daten eines bestimmten Plugins Graphen generieren, aber das Plugin liefert keine Performancedaten und deshalb werden keine Graphen erstellt.

Lösung

Sie können Sich in diesen Fällen mit einem einfachen, eigenen Plugin behelfen. Dieses muss dann

1. die Prüfungsabfrage von Nagios/Icinga an das eigentliche Plugin weiterleiten,
2. die Rückgabe von diesem entgegennehmen und
3. daraus einen Rückgabewert für Nagios/Icinga generieren, der Performancedaten (siehe Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*) in der von Ihnen gewünschten Art und Weise enthält, und diese an Nagios/Icinga zurückgeben.

Damit können Sie dann statt des ursprünglichen Plugins direkt das von Ihnen erstellte Plugin aufrufen. Dieses wird dann das eigentliche Plugin aufrufen und die von Ihnen gewünschte Ausgabe an Nagios/Icinga zurückgeben.

Wir zeigen Ihnen im Folgenden am Beispiel des Plugins check_procs (siehe Rezept 5.3, *Prozesse überwachen (check_procs)*), wie Sie ein solches Wrapper-Script erstellen und einbinden. Das Plugin `check_procs` liefert bei einem Aufruf ohne Parameter die Zahl der laufenden Prozesse, wie in folgendem Aufruf:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_procs
PROCS OK: 91 processes
```

Diese Metrik ist eigentlich gut zum Graphen geeignet, aber das Plugin gibt eben keine Performancedaten aus. PNP4Nagios wird deshalb auch erst einmal keine Graphen zu diesem Plugin erzeugen können. Erstellen Sie in einem Ihrer Plugin-Verzeichnisse deshalb eine neue Datei, beispielsweise unter dem Namen *wrapper_check_procs.sh*, mit folgendem Inhalt (passen Sie dabei gegebenenfalls den Pfad zu `check_procs` an):

```
#!/bin/bash

# Ausführung des Plugins mit Übergabe aller Parameter und Speicherung der
# Ausgabe des Plugins in der Variable LINE
LINE=`/usr/local/icinga/libexec/check_procs $*`

# Speicherung des Rückgabe-Codes des Plugins in der Variable RC
RC=$?

# Speicherung der dritten Zeichenkette aus der zurückgegebenen
# Zeichenkette in der Variable COUNT
COUNT=`echo $LINE | awk '{print $3}'`

# Ausgabe der Zeichenkette aus der Variable LINE und Anhängen der Prozesszahl
# aus der Variablen PROCS als Performancedaten
echo $LINE \| procs=$COUNT

# Programm mit Status-Code aus der Variablen RC beenden
exit $RC
```

Das Script entstammt in ursprünglicher Form der Dokumentation von PNP4Nagios (*http://docs.pnp4nagios.org/pnp-0.6/wrapper?s[]=check&s[]=procs*), wurde von uns aber modifiziert und erweitert. Unseren Kommentaren zu den einzelnen Zeilen können Sie entnehmen, welche Funktion die jeweils darunter angefügte Zeile hat. Das Script verwendet die Kommandos `echo` und `awk`, zu denen Sie Hilfeseiten mit den Anweisungen `man echo` und `man awk` aufrufen können. Testen Sie dieses neue Plugin zunächst von der Kommandozeile, wie in folgendem Beispiel:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec2/wrapper_check_procs.sh
PROCS OK: 88 processes | procs=88
```

Vergleichen Sie die Ausgabe mit der Ausgabe des eigentlichen, im Voranstehenden angeführten Plugins. Wie im Rezept zur API (siehe Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*) beschrieben werden Sie die Erweiterung um Performancedaten erkennen. Um dieses Wrapper-Script nun in die laufende Konfiguration einzubinden, ändern Sie die bestehende Befehlsdefinition für `check_procs` (siehe Rezept 5.3, *Prozesse überwachen (check_procs)*) wie folgt (die alte Zeile ist hier auskommentiert angeführt):

```
# check_procs
define command{
       command_name                    check_procs
     # command_line                    $USER1$/check_procs $ARG1$
       command_line                    $USER2$/wrapper_check_procs.sh $ARG1$
}
```

Beachten Sie dabei, dass wir das Verzeichnis für die Plugins hier mit dem Makro `$USER2$` ersetzt haben, und passen Sie dies gegebenenfalls an Ihre Umgebung an. Jetzt müssen Sie lediglich noch Ihr Monitoringsystem neu starten, damit die geänderte Konfiguration aktiv wird. Nachdem genügend Messwerte vorhanden sind, werden Sie in PNP4Nagios einen Graphen wie den in Abbildung 7-4 dargestellten finden.

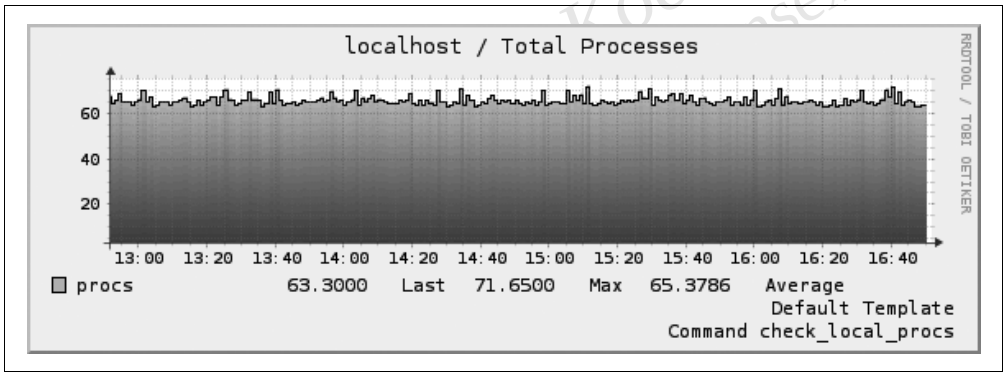


Abbildung 7-4: PNP4Nagios-Graph aus dem beschriebenen Wrapper-Script

Beachten Sie, dass die Auflösung des Graphen vom Prüfintervall abhängig ist. Dem dargestellten Graphen liegen Daten von Prüfungen in einem minütlichen Intervall zugrunde. Wenn das Prüfintervall bei Ihnen größer ist, wird Ihr Graph gegebenenfalls weniger fein aufgelöst.

Diskussion

Wenn Sie bei Plugins Performancedaten nachrüsten möchten, ist ein Wrapper-Script wie das beschriebene oft eine einfache Lösung. Handelt es sich bei dem Plugin selbst um ein Script, dann können Sie natürlich alternativ das Script direkt anpassen (und dann am

besten das geänderte Script gleich der übrigen Nagios/Icinga-Community zur Verfügung stellen). Allerdings kann dies je nach Aufbau des jeweiligen Scripts durchaus etwas Einarbeitung erfordern. Bei binären Plugins müssten Sie die jeweilige Programmiersprache beherrschen, den Quelltext ändern und das Plugin dann neu kompilieren. Dann ist der Änderungsaufwand also entsprechend höher. Deshalb bietet sich gerade in einem solchen Fall der Weg über ein Wrapper-Script an.

Nachteil dieser Lösung ist natürlich der, dass bei jedem Test ein zusätzliches Script aufgerufen werden muss. Auch wenn es sich um ein vergleichsweise schlankes Script handelt, verbraucht doch jeder Aufruf Ressourcen. Entsprechend sollten Sie bei stark belasteten Systemen dann gegebenenfalls doch in Erwägung ziehen, das Orginalscript beziehungsweise den Quelltext entsprechen anzupassen.

Siehe auch

- Rezept zur API: Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*
- Rezept zu `check_procs`: Rezept 5.3, *Prozesse überwachen (check_procs)*
- Das ursprüngliche Plugin aus der PNP4Nagios-Dokumentation *http://docs.pnp4nagios.org/pnp-0.6/wrapper?s[]=check&s[]=procs*
- Hilfe-Seiten zu den Programmen echo und awk: `man echo` und `man awk`

7.3 Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen

Problem

Sie möchten unterschiedliche Test durchführen und die Prüfungsergebnisse zusammenfassen. Nagios/Icinga soll dann das zusammengefasste Ergebnis verarbeiten.

Lösung

Die Zusammenfassung unterschiedlicher Tests erfordert eine Logik. Nagios/Icinga ist nicht darauf ausgelegt, eine solche zu implementieren. Entsprechend müssen Sie die notwendigen Verarbeitungsschritte extern durchführen und Nagios/Icinga das bereits zusammengefasste Ergebnis übergeben.

In Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* haben wir Ihnen bereits gezeigt, wie Sie ein einzelnes Ergebnis entsprechend Ihrer Anforderungen auslesen, verändern und dann an Nagios/Icinga weiterleiten können. Dabei rufen Sie ein Plugin auf, werten das Ergebnis aus und geben dann ein modifiziertes beziehungsweise erweitertes Ergebnis an Nagios/Icinga zurück. Wir zeigen Ihnen im Folgenden eine Erweiterung dieses Verfahrens, bei dem Sie mehrere Ergebnisse abrufen, die Ergebnisse zusammenführen und dann zurückgeben.

Als Beispiel dient dabei die Abfrage von über SNMP (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*) eingebundenen Plugins (siehe Rezept 2.5, *Plugins unter Linux über SNMP ausführen*), bei denen das Ergebnis in mehreren unterschiedlichen OIDs (Object Identifiern) gespeichert wird. Einzelne OIDs könnten Sie mit `check_snmp` (siehe Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*) bequem abfragen. Um aber über SNMP eingebundene Plugins vollständig auszuwerten, müssen Sie mehrere Werte abfragen und zusammenführen. Insgesamt stehen Ihnen bei über SNMP eingebundenen Plugins diverse OIDs in der `NET-SNMP-EXTEND-MIB` zur Verfügung. Erforderlich für eine Auswertung von Plugin-Ausgaben (siehe Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*) sind dabei insbesondere das nummerische Ergebnis, also der Status selbst, sowie die textuelle Ausgabe des ausgeführten Plugins. Diese stehen Ihnen unter den OIDs:

- `NET-SNMP-EXTEND-MIB::nsExtendResult` (nummerisch: `.1.3.6.1.4.1.8072.1.3.2.3.1.4`) für das nummerische Ergebnis und
- `NET-SNMP-EXTEND-MIB::nsExtendOutput1Line` (nummerisch: `.1.3.6.1.4.1.8072.1.3.2.3.1.1`) für die textuelle Rückgabe

zur Verfügung. Diese werden entsprechend für jedes über SNMP ausgeführte Plugin geführt. Um anzugeben, welches Ergebnis Sie abrufen möchten, müssen Sie den OIDs deshalb noch den Namen des Plugins anhängen, so wie Sie ihn in der Konfigurationsdatei des SNMP-Servers angegeben haben (siehe Rezept 2.5, *Plugins unter Linux über SNMP ausführen*). Für ein unter dem Namen `check_apt` eingebundenes Plugin würden die OIDs also beispielsweise folgendermaßen aussehen:

- `NET-SNMP-EXTEND-MIB::nsExtendResult."check_apt"` (numerisch: `.1.3.6.1.4.1.8072.1.3.2.3.1.4.9.99.104.101.99.107.95.97.112.116`) und
- `NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."check_apt"` (numerisch: `.1.3.6.1.4.1.8072.1.3.2.3.1.1.9.99.104.101.99.107.95.97.112.116`).

Auch wenn Plugins theoretisch mehrere Zeilen zurückgeben können (siehe hierzu Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*), berücksichtigen wir hier der Einfachheit halber nur die erste Zeile der Plugin-Ausgabe. In Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* wird lediglich die textuelle Beschreibung ausgewertet und anhand dieser auf den zurückgelieferten Status geschlossen. Um nun aber direkt den vom Plugin gelieferten Status auszuwerten, müssen wir dem Wrapper eine entsprechende Abfrage für diese zusätzliche OID hinzufügen. Um sicherzustellen, dass beide Abfragen erfolgreich waren, haben wir darüber hinaus eine (einfache) Logik implementiert, um Fehler abzufangen:

```
#!/bin/bash
# Parameter
HOST_NAME=$1
PLUGIN_NAME=$2

# Einstellungen
LOCAL_PLUGIN_PATH="/usr/local/icinga/libexec"

if ( [ PLUGIN_NAME == "" ] );
then
       echo "ERROR: no plugin specified, exiting."
       exit 3
fi

OID_RESULT_NUM="NET-SNMP-EXTEND-MIB::nsExtendResult.\"$PLUGIN_NAME\""
OID_RESULT_LINE1="NET-SNMP-EXTEND-MIB::nsExtendOutput1Line.\"$PLUGIN_NAME\""

# Ausführung des Plugins für den OID mit dem nummerischen Rückgabewert
# und Speicherung in REMOTE_RC1
REMOTE_RC1=`$LOCAL_PLUGIN_PATH/check_snmp -H $HOST_NAME -o $OID_RESULT_NUM -P 3
-L authPriv -U icinga -a SHA -A "PASSPHRASE" -x AES -X "PASSPHRASE"`

# Speicherung des Rückgabe-Codes des Plugins in der Variable RC
RC=$?

# Fehlerhandlung
if ( [ $RC != 0 ] ) ;
then
       echo "ERROR: 1st query via SNMP failed!"
       exit 3
fi

# Ausführung des Plugins für den OID mit der textuellen Beschreibung
# und Speicherung in REMOTE_RC2
REMOTE_RC2=`$LOCAL_PLUGIN_PATH/check_snmp -H $HOST_NAME -o $OID_RESULT_LINE1 -P 3
-L authPriv -U icinga -a SHA -A "PASSPHRASE" -x AES -X "PASSPHRASE"`

# Speicherung des Rückgabe-Codes des Plugins in der Variable RC
RC=$?

# Fehlerhandlung
if ( [ $RC != 0 ] ) ;
then
       echo "ERROR: 2nd query via SNMP failed!"
       exit 3
fi

# Vorbereitung der Rückgabewerte
RESULT_NUM=`echo $REMOTE_RC1 | cut -d" " -f4`
RESULT_TEXT=`echo $REMOTE_RC2 | cut -d"-" -f2-`
# Textuelle Ausgabe
echo $RESULT_TEXT
# nummerischer Rückgabewert und Beendigung
exit $RESULT_NUM
```

Das Script erwartet als Übergabeparameter die abzufragende Maschine sowie den Namen, unter dem Sie das eigentlich abzufragende Plugin in SNMP eingebunden haben. Als Beispiel haben wir Ihnen in Rezept 2.5, *Plugins unter Linux über SNMP ausführen* die Einbindung von `check_apt` gezeigt und dabei als Namen eben check_apt verwendet, worauf wir im Folgenden aufbauen.

Hinterlegung von Zugangsdaten: Wir haben die Zugangsdaten hier direkt im Skript hinterlegt. Bei Änderungen der Zugangsdaten müssen Sie also gegebenenfalls daran denken, die entsprechenden Anpassungen in dem voranstehenden Script vorzunehmen.

Test

Wenn Sie das Script abgespeichert haben, sollten Sie es zunächst von der Kommandozeile testen:

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/icinga/libexec2/wrapper_check_snmp_plugin.sh 10.85.58.42
check_apt
APT OK: 0 packages available for upgrade (0 critical updates). |
```


Sie sehen hier zunächst nur die textuelle Ausgabe, Performancedaten werden keine generiert. Mit dem folgendem Aufruf können Sie sich direkt im Anschluss den numerischen Rückgabewert, den sogenannten `exit-code`, des zuletzt durchgeführten Befehls anzeigen lassen. Wenn Sie diesen Befehl also direkt nach dem manuellen Aufruf des Plugins ausführen, entspricht dieser Wert dem numerischen Status (siehe Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*) für Nagios/Icinga:

```
icinga@moni:~$ echo $?
0
```

Damit hätten Sie also separat beide Daten, den textuellen und den numerischen Rückgabewert, über SNMP ausgelesen, für Nagios/Icinga aufbereitet und übergeben.

Einbindung

Sofern das Script wie gewünscht funktioniert, können Sie es dann wie folgt einbinden. Legen Sie sich zunächst eine entsprechende Befehlsdefinition (siehe Rezept 4.3, *Befehle hinzufügen (command)*) an:

```
define command {
       command_name                    wrapper_check_snmp_plugin
       command_line                    $USER2$/wrapper_check_snmp_plugin.sh \
                                       $HOSTADDRESS$ $ARG1$

}
```

Darauf aufbauend können Sie dann Dienst-Definitionen (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) für entsprechend eingebundene Plugins erstellen. Nachfolgend ist eine Definition für das als Beispiel verwendete `check_apt` (siehe Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)* und Rezept 2.5, *Plugins unter Linux über SNMP ausführen*) angeführt:

```
define service {
       service_description                    check_snmp_apt
       use                                    template-srv
       command                                wrapper_check_snmp_plugin.sh!check_apt
       hostgroups                             hg-snmpd-apt
}
```

Passen Sie hierbei gegebenenfalls die zu nutzende Vorlage (hier `template-srv`) und die Zuweisung (hier über die Maschinengruppe `hg-snmpd-apt`) an Ihre Anforderungen an. Nach der Einbindung wird die Abfrage entsprechend durchgeführt werden.

Diskussion

Die Vorgehensweise, die wir Ihnen in diesem Rezept aufgezeigt haben, kann für Sie in mehrfacher Hinsicht nützlich sein. Zum einen können Sie entsprechende Scripte für Plugins jeglicher Art schreiben. Zum anderen können Sie die Abfrage der hier als Beispiel genutzten Einbindung von Plugins über SNMP verwenden, um auf unkomplizierte Art und Weise Plugins auf anderen Maschinen einzubinden. Beispiele für diese Vorgehensweise stellen wir Ihnen in Rezept 8.1, *Überwachung von Unix-/Linux-Servern* vor.

Siehe auch

- Rezept zur Erstellung eines Wrapper-Scripts: Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*
- Rezepte zu Befehls- und Dienstdefinitionen: Rezept 4.3, *Befehle hinzufügen (command)* und Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Einrichtung von SNMP unter Linux: Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*
- Rezept zur Ausführung von Plugins via SNMP unter Linux: Rezept 2.5, *Plugins unter Linux über SNMP ausführen*
- Rezept zur Plugin-Schnittstelle von Nagios/Icinga (API): Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*
- Rezept zur Erstellung generischer SNMP-Abfragen: Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*
- Rezept zur Überwachung von Aktualisierungen: Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*
- Rezept zur Überwachung von Linux-Servern: Rezept 8.1, *Überwachung von Unix-/Linux-Servern*

7.4 Einbinden des PNP4Nagios-Plugins (check_pnp_rrds)

Problem

Sie möchten die von PNP4Nagios generierten RRDs überwachen (RRD steht für Round Robin Database). Dabei sollen veraltete Datensätze gemeldet werden.

Lösung

Das Plugin `check_pnp_rrds` wird sozusagen mit PNP4Nagios mitgeliefert. Abhängig von Ihrer Distribution und Installationsart finden Sie es an den in Tabelle 7-2 aufgeführten Speicherorten.

Tabelle 7-2: Speicherort des Plugins check_pnp_rrds

Installationsart/Distribution	Ort
Quellcode-Installation	/usr/local/pnp4nagios/libexec/
Paket-Installation Debian/Ubuntu	/usr/lib/pnp4nagios/libexec/
Paket-Installation SLES|OpenSuSE	/usr/lib/nagios/plugins/
Paket-Installation 'RHEL|CentOS|Fedora	/usr/libexec/pnp4nagios/

Wir beziehen uns im Folgenden exemplarisch auf unsere Referenzinstallation, der Installation von Icinga mit dem PNP4Nagios-Addon aus Quellen unter Debian (siehe Rezept 1.4, *Icinga aus den Quellen installieren*).

Befehlsdefinition und notwendige Optionen

Um das Plugin einzubinden definieren Sie zunächst einen entsprechenden Befehl (siehe hierzu Rezept 4.3, *Befehle hinzufügen (command)*):

```
# check_pnp_rrds
define command{
       command_name                    check_pnp_rrds
       command_line                    /usr/local/pnp4nagios/libexec/ \
                                       check_pnp_rrds.pl
}
```

Beachten Sie hierbei, dass das Plugin zunächst unter einem von den anderen Plugins abweichendem Pfad abgelegt wurde. Diese Kommandodefinition verwendet nicht wie sonst üblich ein Makro für den Pfad, sondern enthält bereits den vollständigen Pfad, da unter diesem Pfad sich eben nur genau dieses Plugin befindet. Das Plugin verwendet die folgenden Parameter und Standardwerte als Pflichtoptionen:

- `-w` / `--warning` und `-c` / `--critical`

  Mit diesen Optionen legen Sie fest, bei wie vielen veralteten Dateien der Status auf `WARNING` (Vorgabewert `1`) beziehungsweise `CRITICAL` (Vorgabewert `10`) gesetzt werden soll.

- `-a` / `--fileage`

  Mit dieser Option legen Sie fest, ab welchen Alter in Tagen Dateien als veraltet berücksichtigt werden (Vorgabewert: 7).

- `-p` / `--rrdpath`

  Mit dieser Option teilen Sie dem Plugin mit, unter welchem Pfad es nach den PNP-Dateien suchen soll (Vorgabewert ist */usr/local/pnp4nagios/var/perfdata*).

- `-t` / `--timeout`

  Mit dieser Option legen Sie fest, nach welcher Laufzeit in Sekunden die Ausführung des Plugins als nicht erfolgreich abgebrochen wird (Vorgabewert: 10).

Da das Plugin für diese Parameter Vorgabewerte setzt, können Sie auch gar keine Optionen angeben.

Weitere Optionen

Als weiterer Parameter für Anpassungen an ihre Anforderungen steht Ihnen der Folgende zur Verfügung:

- `--ignore-hosts`

  Mit dieser Option können Sie über einen regulären Ausdruck Gerätenamen spezifizieren, die das Plugin ignorieren soll.

Dienstdefinition

Definieren Sie dann weiter einen entsprechenden Dienst (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*), der die Befehlsdefinition an die Monitoring-Maschine anbindet. Von einer nicht verteilten Umgebung ausgehend ist dies `localhost`:

```
define service{
       service_description                    PNP-RRDs
       use                                    template-service-generic
       check_command                          check_pnp_rrds
       host_name                              localhost
}
```

Dabei wird hier die Vorlage `template-service-generic` referenziert. Passen Sie den Namen gegebenenfalls an, um die von Ihnen gewünschte Vorlage zu verwenden. Damit haben Sie das Plugin dann in Nagios/Icinga eingebunden und nach dem nächsten Neustart werden entsprechende Prüfungen durchgeführt.

Test, Rückgabewerte und Graphen

Vor einer Einbindung sollten Sie das Plugin wie immer von der Kommandozeile testen.

Test mit Benutzerrechten: Achten Sie beim Testen darauf, dass Sie als der Benutzer angemeldet sind, unter dem auch Nagios/Icinga die Plugins aufruft. So merken Sie sofort, wenn es ein Problem mit den Rechten gibt. Diese können auftreten, wenn Sie etwa neue Plugins als Benutzer `root` installieren und vergessen, die Rechte anzupassen.

```
icinga@moni:~$ /usr/local/pnp4nagios/libexec/check_pnp_rrds.pl
OK: 88 XML Files checked. 0 RRD Errors found. 0 old XML Files found | total=88 errors=0;1
;10;0;88 old=0;1;10;0;88
```

Das Plugin gibt jeweils durch das Pipe-Symbol separiert Performancedaten (siehe hierzu Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*) zurück. Entsprechend erhalten Sie in PNP4Nagios automatisch einen Graphen, wie in Abbildung 7-5 dargestellt.

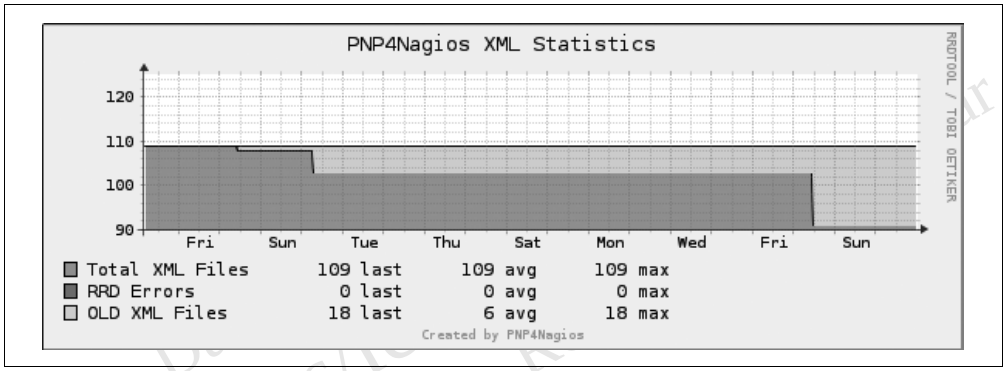


Abbildung 7-5: PNP4Nagios-Graph der RRD-Datenbanken

Diskussion

Das Plugin meldet Ihnen sowohl Fehler in den RRD-Dateien, als auch veraltete XML-Dateien. Wir erklären Ihnen diese Phänomene im Folgenden und erläutern, was Sie jeweils tun können.

RRD-Fehler treten auf, wenn sich die Performance-Ausgaben von Plugins ändern, nachdem bereits Messwerte vorliegen. Für die RRDTools in Kombination mit PNP4Nagios gibt es unseres Wissens keine einfache Möglichkeit, die neuen mit den alten Daten in Übereinstimmung zu bringen. Lösen können Sie dies, indem Sie die fraglichen Dateien löschen (siehe hierzu Rezept 9.3, *Alte Graphen und Daten aus PNP4Nagios entfernen*). Beachten Sie, dass dabei automatisch alle Alt-Daten verlorengehen.

Die als veraltetet gemeldeten XML-Dateien beziehen sich auf nicht mehr vorhandene Dienste. Jeder Dienst, dessen Plugin Performancedaten liefert, erzeugt automatisch eine XML-Datei, die diese Daten beschreibt, sowie eine entsprechende RRD-Datenbank mit den dazugehörigen Daten. Wenn Sie einen solchen Dienst löschen, sollten Sie auch die dazugehörigen Dateien entfernen (siehe Rezept 9.3, *Alte Graphen und Daten aus PNP4Nagios entfernen*).

Damit ist Ihnen das Plugin eine wichtige Hilfe bei der Arbeit mit Nagios/Icinga und PNP4Nagios. Der Standardwert für das zu prüfende Alter in Tagen ist dabei auf sieben Tage gesetzt. Unserer Erfahrung nach ist dies unnötig hoch. Damit Sie zum Zeitpunkt der Benachrichtigung leichter nachvollziehen können, welche Änderungen am System zum Ist-Stand geführt haben, empfehlen wir Ihnen einen kürzeren Benachrichtigungszeitraum von zwei Tagen. Damit können Arbeiten auch mal länger als einen Tag dauern, ohne dass dies zu einer Warnung führt, und gleichzeitig ist der Umfang der bei einer Warnung bereits fehlenden Daten gegebenenfalls noch nicht ganz so hoch.

Siehe auch

- Rezept zur Installation von Icinga aus den Quellen: Rezept 1.4, *Icinga aus den Quellen installieren*
- Rezepte zu Befehls- und Dienstdefinitionen: Rezept 4.3, *Befehle hinzufügen (command)* und Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Einführung in die von Nagios/Icinga verwendete API: Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*
- Rezept zur Entfernung von Alt-Daten in PNP4Nagios: Rezept 9.3, *Alte Graphen und Daten aus PNP4Nagios entfernen*

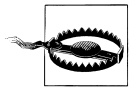
7.5 Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins

Problem

Sie möchten Daten über SNMP abfragen und hierzu externe Plugins einbinden.

Lösung

Wir zeigen Ihnen am Beispiel der Manubulon SNMP-Plugins (siehe *http://nagios.manubulon.com/*), wie Sie externe Plugins einbinden und worauf Sie dabei achten sollten. Es gibt eine große Anzahl an Plugins, die von anderen Benutzern erstellt und zur Verfügung gestellt werden. Die im Folgenden beschriebenen SNMP-Plugins dienen hier nur als Beispiel für die generelle Einbindung von externen Plugins.



Qualität externer Plugins: Die Nagiosplugins werden von einem gemeinsamen Team von Entwicklern gepflegt und haben eine hohe Qualität. Sie arbeiten konsistent, zuverlässig und schnell. Bei externen Plugins gibt es häufig keine vergleichbare Zusammenarbeit. Stattdessen wurden diese häufig von einem Benutzer erstellt und nur für seine Umgebung getestet. Hier ist deshalb schnell die Gefahr gegeben, dass die Plugins bei Ihnen nicht wie erwartet funktionieren. Testen Sie externe Plugins deshalb besonders sorgfältig. Es kann manchmal trotz vorhandener Lösungen besser sein, eine eigene Lösung (siehe vorangegangene Rezepte) zu erstellen, als ein bestehendes aber problembehaftetes Plugin zu verwenden.

Bei den Manubulon-Plugins handelt es sich ähnlich wie bei den Nagiosplugins um eine Sammlung von mehreren Plugins. Im Falle der Manubulon-Plugins sind diese alle auf SNMP-Abfragen ausgegelegt und bieten gegenüber der mit `check_snmp` möglichen Einzelabfrage eine jeweils spezifisch angepasste Behandlung der abgefragten Werte.

Aktualisierte Versionen: Die hier beschriebene Version der Manubulon-Plugins wurde seit vielen Jahren nicht mehr gepflegt und aktualisiert. Der Icinga-Entwickler Michael Friedrich hat sich dieses Problems angenommen und stellt unter *https://github.com/dnsmichi/manubulon-snmp* aktualisierte Versionen zur Verfügung. Wir beschreiben hier die Original-Versionen, bei denen es in Abhängigkeit von der genutzten Umgebungen zu Problemen oder unnötigen Fehlermeldungen kommen kann. Sie können deshalb alternativ diese aktualisierten Versionen verwenden. Dabei sollte sich nur die Quelle der Plugins ändern, während die jeweiligen übrigen Beschreibungen zur Verwendung ihre Gültigkeit behalten sollten. Wir haben dies vor Erscheinen des Buches allerdings nicht mehr testen können.

Herunterladen und Hinterlegen

Als ersten Schritt laden Sie sich sich die SNMP-Plugins von *http://nagios.manubulon.com/index_snmp.html* herunter. Die Datei, bei uns war es noch die 2007 freigegebene Version 1.1.1 als `nagios-snmp-plugins.1.1.1.tgz`, entpacken Sie dann folgendermaßen:

```
root@moni:~# mkdir -p /usr/src/manubulon
root@moni:~# cd /usr/src/manubulon
root@moni:/usr/src/manubulon# tar xzf nagios-snmp-plugins.1.1.1.tgz
```

Damit erhalten Sie einen Ordner *./nagios_plugins*, in dem sich neben den Plugins als Perl-Scripte auch Dokumentationen und ein Installationsscript finden. Sie können das Installationsscript verwenden, um die Plugins gemeinsam mit den bereits vorhandenen Plugins in das Plugin-Verzeichnis zu installieren – das Script wird dabei alle relevanten Daten von Ihnen abfragen.

Alternatives Plugin-Verzeichnis: Alternativ können Sie sich jedoch auch ein eigenes Verzeichnis für diese Plugin-Sammlung erstellen und die Plugins so getrennt von anderen hinterlegen. Dies bietet Ihnen bei Aktualisierungen und Migrationen Vorteile, da Sie die Plugins dann bereits fertig nach Installationsquelle geordnet haben. In unseren folgenden Beschreibungen wird deshalb diese Variante erläutert. Bei einer Installation über das Installationsscript können Sie also die folgenden Schritte überspringen.

Erstellen Sie nun zunächst ein eigenes Verzeichnis für die Manubulon-Plugins:

```
root@moni:~# mkdir -p /usr/local/icinga/libexec-manubulon
```

Machen Sie dieses neue Plugin-Verzeichnis dann über ein Benutzermakro (siehe Rezept 4.3, *Befehle hinzufügen (command)*) verfügbar, indem Sie einen entsprechenden Eintrag in der Datei `resource.cfg` vornehmen:

```
icinga@moni:~$ vim /usr/local/icinga/etc/resource.cfg
[...]
$USER3$=/usr/local/icinga/libexec-manubulon
[...]
```

Von den verfügbaren Plugins werden Sie vermutlich nicht alle verwenden. Bei einer manuellen Installation können Sie sich deshalb auf Plugins beschränken, die Sie verwenden möchten. Im Folgenden betrachten wir exemplarisch die folgenden Manubulon-Plugins:

- `check_snmp_storage.pl`

  ... zur Abfrage der Belegung von Speichermedien.
- `check_snmp_mem.pl`

  ... für den Test der Belegung von Haupt- und Swap-Speicher.
- `check_snmp_int.pl`

  ... zur Abfrage der Schnittstellen-Status und -Auslastung.
- `check_snmp_load.pl`

  ... zur Prüfung von System- und Prozessorlast.
- `check_snmp_env.pl` (beziehungsweise `check_snmp_environment.pl`, siehe unten)

  ... für den Test von Sensordate.

Kopieren Sie diese Dateien nun aus dem entpackten Archiv in das neu angelegte Pluginverzeichnis und legen Sie den Monitoring-Benutzer (bei unserer Icinga-Referenzinstallation hier `icinga`) als Besitzer fest:

```
root@moni:~# cd /usr/src/manubulon/nagios_plugins
root@moni:/usr/src/manubulon/nagios_plugins# cp check_snmp_int.pl check_snmp_load.pl
check_snmp_mem.pl check_snmp_storage.pl /usr/local/icinga/libexec-manubulon/
root@moni:/usr/src/manubulon/nagios_plugins# chown -R icinga:icinga
/usr/local/icinga/libexec-manubulon
```

Execute bit: Achten Sie darauf, dass die Plugins ausführbar sind. Das ist bei externen Plugins nicht immer automatisch der Fall. Sie können dies mit dem Kommando `chmod +x DATEINAME` manuell nachholen.

Anpassungen bei Nutzung eigener Plugin-Verzeichnisse

Sie könnten jetzt bereits aus Icinga/Nagios auf die Plugins zugreifen, allerdings sind diese unter Umständen noch gar nicht lauffähig. Bei der Verwendung eines zusätzlichen Verzeichnisses werden die neuen Plugins in der Regel den Pfad zum Hauptverzeichnis benötigen, um auf die darin enthaltenen Funktions-Bibliotheken zugreifen zu können. Bei den Manubulon-Plugins ist hierzu am Anfang der Scripte eine entsprechende Zeile enthalten.

Da wir bei unserem Referenzsystem Icinga verwenden, die Plugins aber von einem Nagios ausgehen, muss dieser Pfad in jedem Plugin angepasst werden. Öffnen Sie hierzu alle Plugins, die Sie verwenden wollen mit einem Editor, und ändern Sie gegebenenfalls

den Pfad in der entsprechenden Zeile in passender Weise ab. Im Beispiel für unsere Referenzinstallation ersetzen wir die vorhandene Pfadangabe `/usr/local/nagios/libexec` also wie folgt:

```
[...]
use lib "/usr/local/icinga/libexec";
[...]
```

Weitere Vorbereitungen

Häufig benötigen Plugins eine Speichermöglichkeit, um Werte einer Messung für einen Vergleich bis zur nächsten Auswertung vorzuhalten. Bei den hier betrachteten Plugins ist dies etwa `check_snmp_int.pl`, welches die über SNMP abgefragten Zähler-Werte zwischenspeichert und zu den Messungen jeweils Differenzen berechnet. In solchen Fällen müssen Sie sicherstellen, dass der entsprechende Benutzer (bei uns `icinga`) schreibenden Zugriff auf das entsprechende Verzeichnis hat. Bei `check_snmp_int.pl` ist hierzu das Verzeichnis */tmp* vorgesehen. Wenn Sie ein anderes Verzeichnis verwenden möchten, müssen Sie eine entsprechende Änderung am Skriptcode vornehmen. Passen Sie hierzu im Check die folgende Zeile an Ihre Anforderungen an:

```
[...]
my $o_base_dir="/tmp/tmp_Nagios_int.";
[...]
```

Beachten Sie dabei bitte, dass hier nicht nur ein Verzeichnis angegeben wird, sondern auch ein Teil des zu verwendenden Dateinamens.

Allgemeine Parameter aller Plugins

Im Folgenden stellen wir Ihnen einige Anwendungsbeispiele für die neu installierten Plugins vor. Bei jedem Plugin müssen dabei mindestens folgende Parameter mit angegeben werden:

- `-H` / `--hostname`

  Hier geben Sie den Namen oder die Adresse des Zielgerätes an.

Die Anmeldung bei Nutzung von SNMPv2 legen Sie mit den folgenden Optionen fest:

- `-C`, `--community=COMMUNITY`

  Das für Version 2c zu verwendende, auch *Community-String* genannte Passwort.

- `-2`, `--v2c`

  Diese Option gibt an, dass SNMP Version 2c genutzt werden soll.

für SNMPv3

- `-L, --protocols=<authproto>,<privproto>`

  Mit dieser Option legen Sie fest, welche Verschlüsselungsprotokolle Sie für die Anmeldung beziehungsweise Verbindungsverschlüsselung einsetzen. Über `<authproto>` legen Sie das Protokoll für die Authentifizierung fest, welches `md5` oder `sha`

sein kann (Vorgabe: `md5`). Über `<privproto>` legen Sie das Protokoll für die Verschlüsselung der Datenübertragung fset, welches `des` oder `aes` sein kann (Vorgabe `des`).

- `-l`, `--login=LOGIN`

  Hier geben Sie den für die Anmeldung zu verwendenden Benutzernamen an.

- `-x`, `--passwd=PASSWD`

  Hier geben Sie das für die Anmeldung zu nutzende Passwort an.

- `-X`, `--privpass=PASSWD`

  Hier geben Sie gegebenenfalls das für die Verbindungsverschlüsselung zu nutzende Passwort an.

Das Plugin check_snmp_mem.pl

Das Plugin `check_snmp_mem.pl` testet die Speicherbelegung von Haupt- und Swap-Speicher. Abgefragt wird dabei der Teilbaum unter dem MIB-Objekt `UCDAVIS.storage`. Dieser wird beim standardmäßig installierten NET-SNMP mitgeliefert, so dass die fraglichen MIBs bereits bei Ihnen installiert sein sollten.

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_mem.pl -H 10.85.58.1 -l icinga
-L sha,aes -x "PASSPHRASE" -X "PASSPHRASE" -w 80,85 -c 90,95
Ram : 49.40%, Swap : 1.06% : | ram_used=8286896128B;13420986368;15098609664;0;16776232960
shared=143032320B;;;0;16776232960 cache=8249991168B;;;0;16776232960
unused=96313344B;;;0;16776232960 swap_used=364634112B;29205770650;32641743667;0;
34359730176
```

Das Plugin benötigt neben den weiter vorne erwähnten Daten für Zielmaschine und Zugang als weitere Parameter Schwellwerte für `WARNING` und `CRITICAL` jeweils in Form von zwei durch Kommata separierten Prozentwerten, wobei der erste für den Haupt- und der zweite für den Swap-Speicher gilt.

Das Plugin check_snmp_storage.pl

Mit `check_snmp_storage.pl` können Sie die Speicherbelegung von verfügbaren Speichermedien abfragen. Das Plugin fragt dazu den Teilbaum unter dem MIB-Objekt `MIB-2.host` ab. Mit folgendem Beispiel überprüfen Sie die Auslastung des Laufwerks C: einer Windows-Maschine via SNMP Version 2c:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_storage.pl -H 10.85.58.111
-C 'COMMUNITY' -m "C:" -w 50 -c 90
C:\ Label: Serial Number 681bd690: 63%used(13GB/20GB) (>50%) : WARNING
```

Eine Abfrage der Auslastung des Swap-Speichers eines Unix/Linux Rechner via SNMP Version 3 sieht wie folgt aus:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_storage.pl -H 10.85.58.1 -l icinga
-L sha,aes -x "PASSPHRASE" -X "PASSPHRASE" -m "Swap" -w 50 -c 90
Swap space: 0%used(0GB/32GB) (<50%) : OK
```

Neben den weiter vorne genannten Parametern für Zielmaschine und Zugang sind hier als zusätzliche Parameter erforderlich:

- `-m`, `--name=NAME`

  Mit dieser Option geben Sie den Name des abzufragenden Mediums an, wie es unter der OID `HOST-RESOURCES-MIB::hrStorageDescr` geführt wird (dies können Mountpoints wie beispielsweise `/var`, generische Angaben wie `Swap Space` oder Laufwerksbezeichnungen wie `C:\` sein).

- `-w` und `-c`

  Mit diesen Optionen geben Sie die Warnschwellen für die Status `WARNING` (`-w`) und `CRITICAL` (`-c`) in Prozent an.

Sie können die Beschreibungen mit dem Kommando `snmpwalk` abfragen. Die Abfrage über SNMP Version 2c auf einer Windows-Maschine sieht beispielsweise folgendermaßen aus:

```
icinga@moni:~$ snmpwalk -v 2c -c PASSPHRASE 10.85.58.111 HOST-RESOURCES-
MIB::hrStorageDescr
HOST-RESOURCES-MIB::hrStorageDescr.1 = STRING: C:\ Label:  Serial Number 681bd690
HOST-RESOURCES-MIB::hrStorageDescr.2 = STRING: D:\
HOST-RESOURCES-MIB::hrStorageDescr.3 = STRING: Virtual Memory
HOST-RESOURCES-MIB::hrStorageDescr.4 = STRING: Physical Memory
```

Bei einer Abfrage für SNMP Version 3 auf einen Unix/Linux-Rechner sieht das Kommando beispielsweise wie folgt aus:

```
icinga@moni:~$ snmpwalk -v 3 -l authPriv -u icinga -a SHA -A "PASSPHRASE" -x AES
-X "PASSPHRASE" 10.85.58.1 HOST-RESOURCES-MIB::hrStorageDescr
HOST-RESOURCES-MIB::hrStorageDescr.1 = STRING: Physical memory
HOST-RESOURCES-MIB::hrStorageDescr.3 = STRING: Virtual memory
HOST-RESOURCES-MIB::hrStorageDescr.6 = STRING: Memory buffers
HOST-RESOURCES-MIB::hrStorageDescr.7 = STRING: Cached memory
HOST-RESOURCES-MIB::hrStorageDescr.10 = STRING: Swap space
HOST-RESOURCES-MIB::hrStorageDescr.31 = STRING: /
HOST-RESOURCES-MIB::hrStorageDescr.32 = STRING: /boot
HOST-RESOURCES-MIB::hrStorageDescr.33 = STRING: /var
HOST-RESOURCES-MIB::hrStorageDescr.34 = STRING: /tmp
HOST-RESOURCES-MIB::hrStorageDescr.35 = STRING: /sys/fs/fuse/connections
```

Das Plugin check_snmp_int.pl

Der Check `check_snmp_int.pl` fragt ebenfalls die MIB-II Tabelle ab, diesmal jedoch nach dem Betriebszustand der Netzwerkschnittstellen:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_int.pl -H 10.85.58.1 -l icinga
-L sha,aes -x "PASSPHRASE" -X "PASSPHRASE" -n eth0 -k -y -f -w 50,50 -c 90,90
eth0:UP (1.1KBps/1.1KBps):1 UP: OK | 'eth0_in_prct'=0%;50;90;0;100 'eth0_out_prct'=0%;50;
90;0;100
```

Ergänzend werden in diesem Beispiel folgende Parameter eingesetzt:

- `-n` / `--name=NAME`

  Geben Sie hier den Namen einer abzufragenden Schnittstelle an (beispielsweise `ppp`, `eth`, `eth0`).

- `-k` / `--perfcheck`

  Mit dieser Option legen Sie fest, ob neben die Ein- und Ausgangsbandbreite der angegebenen Schnittstelle abgefragt werden soll.

- `-f` / `--perfparse`

  Wenn Sie die Ausgabe als Performancedaten, etwa für PNP4Nagios, benötigen, können Sie dies mit dieser Option veranlassen.

- `-y` / `--perfprct`; `-Y` / `--perfspeed`

  Legen Sie mit dieser Option die Ausgabe der Performancedaten als Prozentwerte (`-y` und `--perfprct`) bzw. Bytes/s (`-Y` oder `--perfspeed`) fest.

- `-w` und `-c`

  Mit diesen Optionen geben Sie die Warnschwellen für die Status `WARNING` (`-w`) und `CRITICAL` (`-c`) in Prozent an.

Das Plugin check_snmp_load.pl

Das Plugin `check_snmp_load.pl` prüft die Systemlast eines Gerätes. Es nutzt dabei ebenfalls die MIB-II, berücksichtigt auch mehrere Prozessoren und funktioniert sowohl mit Windows als auch mit Unix/Linux und vielen weiteren Geräten. Die Abfrage einer Windows-Maschine mittels SNMP Version 2c sieht wie folgt aus:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_load.pl -H 10.85.58.111
-C 'COMMUNITY' -w 80 -c 90
1 CPU, load 7.0% < 80% : OK
```

Dabei geben Sie mit den Optionen `-w` und `-c` jeweils einen Schwellwert für die Status `WARNING` und `CRITICAL` in Prozent an.

Im folgenden Beispiel wird ein Linux-System unter Nutzung von SNMP Version 3 abgefragt:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_load.pl -H 10.85.58.1 -l icinga
-L sha,aes -x "PASSPHRASE" -X "PASSPHRASE" -w 8,6,5 -c 8,7,6 -T netsl
Load : 6.67 5.72 4.35 : OK
```

Hierbei haben wir über den Parameter `-T` einen Gerätetyp mit angeben, um eine spezifische MIB abzufragen, wobei die Auslastung als sogenannter Load abgefragt wird, die typischerweise in Form von Mittelwerten für eine, drei und fünfzehn Minuten angegeben wird. Deshalb müssen Sie in diesem Falle auch jeweils drei Werte für die Warnwerte angeben. Das Plugin kennt daneben auch die entsprechenden OIDs für einige Geräte von Cisco und HP.

Das Plugin check_snmp_environment.pl

Das Perl-Script `check_snmp_environment.pl` ist eine Weiterentwickung des Manubulon Plugins `check_snmp_env.pl`. Sie erhalten es unter *http://exchange.nagios.org/directory/Plugins/Hardware/Network-Gear/Cisco/Check-various-hardware-environmental-sensors/details*. Sie sollten diese erweiterte Version verwenden, da die ursprüngliche nur in einem Bruchteil der Szenarien funktioniert, die die neue Version bedienen kann. Im Wesentlichen überprüft das Script eine Vielzahl von Hardware-Sensoren, die Statusinformationen von Netzteilen, Lüftern, sowie Karten- und Modulstatus geläufiger Switches und Routern liefern. So werden beispielsweise Geräte von Cisco, Brocade/Foundry, Juniper/NetScreen, sowie der HP ProCurve Serie unterstützt.

Zusätzlich lassen sich außerdem Sensoren von Linux-Systemen mit Hilfe des Hardware-Monitoringtools `lm-sensors` von Linux (siehe dazu Rezept 5.6, *Sensor-Daten überwachen (check_sensors)*) auslesen. Mit folgendem Test fragen Sie die von `lm-sensors` bereitgestellten Sensordaten eines Linux-Systems über SNMP Version 2 ab:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_environment.pl -H localhost
-C 'COMMUNITY' -T linux
TSensor MB Temperature : 95
MSensor Core 0 : 56000.00
MSensor CHASSIS1 FAN Speed : 7200000.00
MSensor CHASSIS1 FAN Speed : 922000.00
VSensor Vcore Voltage : 1.70
MSensor Core 1 : 78000.00
TSensor temp1 : 100
MSensor POWER FAN Speed : 0.00
VSensor +3.3 Voltage : 3.34
VSensor Vcore Voltage : 1.18
[...]
```


Diskussion

Mit den gezeigten Schritten haben Sie eine Auswahl der Manubulon-Plugins für die Benutzung vorbereitet, so dass Sie mit dem Anlegen entsprechender Kommandodefinitionen (siehe Rezept 4.3, *Befehle hinzufügen (command)*) fortfahren können. Wir gehen an dieser Stelle nicht auf die vielfältigen Optionen und sich daraus ergebenden Möglichkeiten ein. Im Rahmen der Rezepte in Kapitel 8, *Überwachung von Maschinen, Diensten und Infrastruktur* zeigen wir Ihnen allerdings viele Einsatzmöglichkeiten dieser Plugins auf.

Die in Tabelle 7-3 von den Autoren der Manubulon-Plugins erstellte Kompatibilitätsmatrix gibt Ihnen einen Überblick, auf welchen Systemen sich die Plugins einsetzen lassen.

Tabelle 7-3: Kompatibilitätsmatrix für Manubulon-Plugins

Plugin / System	_storage	_int	_process	_mem	_load	_env
Linux	OK : - '/home', '/var' - 'Swap', 'Real Memory'	OK : - 'eth' - 'ppp'	OK	OK (mem & swap)	OK	OK : mit lm_sensors
Windows	OK : - '^[CDE]:' - 'Physical Memory' - 'Virtual Memory'	(OK)*	OK	NEIN siehe check_snmp_storage	OK	NEIN
Cisco/HP Switch	N/A	OK : - 'Giga' : alle Gigabit-Ports - 'Fast.*0.1[1234]' : prüft FastEternet0/11 bis 0/14.	N/A	OK	OK	OK**

* Gerätebezeichner müssten in vielen Fällen umbenannt werden (siehe dazu *http://www.nagios-wiki.de/nagios/plugins/check_snmp_int.pl*).

** Viele Geräte der Hersteller Cisco, Nokia, BlueCoat, IronPort, Foundry, Linux, Extreme, Juniper, HP ProCurve, NetScreen, Citrix und Transmode lassen sich abfragen. Das Plugin lässt sich jedoch leicht an andere Geräte anpassen.

Manubulon C-Paket: Eine Auswahl der Plugins ist unter *http://nagios.manubulon.com/package_c.html* auch als in C geschriebene Version verfügbar. Diese Plugins müssen Sie vor der Nutzung kompilieren. Damit ist die Installation zunächst etwas aufwendiger, dafür die Ausführung dann aber deutlich schneller. Der Autor hat in dieser Version darüber hinaus eine Zwischenspeicherung der Index-Tabellen eingebaut, was den Beschleunigungseffekt weiter verstärken soll. Wenn Sie also sehr viele Prüfungen auf Basis dieser Plugins durchführen, sollten Sie eventuell die C-Version der betreffenden Plugins testen und gegebenenfalls einsetzen.

Siehe auch

- Homepage der Plugins: *http://nagios.manubulon.com/*, Version in C unter *http://nagios.manubulon.com/package_c.html*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zur Überwachung von Sensordaten: Rezept 5.6, *Sensor-Daten überwachen (check_sensors)*
- Rezepte, in denen die Nutzung der hier gezeigten Plugins aufgegriffen wird: Kapitel 8, *Überwachung von Maschinen, Diensten und Infrastruktur*
- Homepage des Plugins `check_snmp_environment.pl`: *http://exchange.nagios.org/directory/Plugins/Hardware/Network-Gear/Cisco/Check-various-hardware-environmental-sensors/details*

KAPITEL 8

Überwachung von Maschinen, Diensten und Infrastruktur

In diesem Kapitel führen wir für Sie die Ausführungen der vorangegangenen Kapitel zusammen. Für einzelne Szenarien zeigen wir auf, welche der in diesem Buch angesprochenen Rezepte Sie wie zum Einsatz bringen können. Wir haben uns dabei bemüht, häufig auftretende Anwendungsbeispiele zu wählen. Im Einzelnen sind dies folgende:

- Zunächst die generelle Überwachung von Servern unter Unix/Linux (siehe Rezept 8.1, *Überwachung von Unix-/Linux-Servern*) beziehungsweise Windows (siehe Rezept 8.2, *Überwachung von Windows-Maschinen*) und von virtuellen Maschinen (siehe Rezept 8.3, *Überwachung von Hypervisoren und virtuellen Maschinen*) sowie die Einbindung der aktiven Infrastruktur (siehe Rezept 8.4, *Überwachung der aktiven Infrastruktur*).
- Im speziellen betrachten wir dann weiter die folgenden, ergänzenden Szenarien:
  - Dateiserver (siehe Rezept 8.5, *Dateiserver überwachen*)
  - E-Mail-Server (siehe Rezept 8.6, *E-Mail-Server überwachen*)
  - Webserver (siehe Rezept 8.7, *Webserver überwachen*)
  - Datenbankserver (siehe Rezept 8.8, *Datenbankserver überwachen*)
  - Zentrale Netzwerkdienste (siehe Rezept 8.9, *Zentrale Netzwerk-Dienste überwachen (DNS, DHCP, LDAP, AD)*),

Bitte beachten Sie, dass die Rezepte hier auf den vorangegangenen aufbauen. Wir verweisen Sie allerdings jeweils auf die entsprechenden detaillierteren Rezepte, damit Sie bei Bedarf Details nachschlagen können. Wir gehen dabei hauptsächlich auf die hier im Buch beschriebenen Plugins ein.

An dieser Stelle möchten wir Sie noch einmal darauf hinweisen, dass Sie im Zuge des Ausbaus Ihrer Monitoring-Installation vermutlich auch auf externe beziehungsweise eigene Plugins zurückgreifen werden. Wie Sie solche einbinden, haben wir Ihnen im Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins* erläutert. Wenn die mitgelieferten Plugins Ihre Anforderungen also nicht ausreichend erfüllen, schauen Sie sich an, wie andere Benutzer in ähnlichen Fällen vorgegangen sind, indem Sie einige externe Plugins ausprobieren.

8.1 Überwachung von Unix-/Linux-Servern

Problem

Sie möchten eine unter Unix/Linux laufende Maschine überwachen.

Lösung

Sie können auf unterschiedliche Arten auf Maschinen unter Unix/Linux, wie beispielsweise OpenSolaris oder Debian, zugreifen. Wir gehen im Folgenden davon aus, dass Sie auf dem Gerät SNMP eingerichtet haben (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*) und dies für die Abfragen zur Verfügung steht. Alternativ können Sie Plugins aber auch über SSH (siehe hierzu Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten* und Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*) oder NRPE einbinden (siehe hierzu Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*). Im Folgenden beschreiben wir Ihnen die Prüfungen über SNMP, die Sie aber analog auch über die anderen Zugriffsarten durchführen können (siehe hierzu auch die Diskussion im Nachfolgenden).

Bereiten Sie gegebenenfalls zunächst die Maschinen entsprechend vor und hinterlegen Sie die erforderlichen Zugangsdaten auf Ihrem Monitoring-Server. Bei SNMP sind dies für Version 1 und 2 die Community-Strings beziehungsweise für Version 3 das Benutzerkonto nebst dazugehörigem Passwort. Sie sollten diese in entsprechenden Benutzermakros von Nagios/Icinga hinterlegen, also in der Datei *resource.cfg* definieren (siehe hierzu Rezept 4.3, *Befehle hinzufügen (command)*). Wir werden Ihnen basierend auf SNMP Version 3 im Folgenden eine Auswahl an Standard-Überwachungen vorschlagen, die Sie dann entsprechend Ihres Bedarfs einbinden können. Testen Sie vor der Einbindung die jeweiligen Befehle, da die Komplexität der Fehlersuche mit der Einbindung weiter ansteigt.

Performancedaten: Wenn die von Ihnen genutzte Plugins Performancedaten zurückgeben, können Sie diese mit Hilfe von PNP4Nagios auswerten lassen. Sie erhalten dann zum Plugin automatisch einen entsprechenden Graphen.

Erreichbarkeit überwachen

Normalerweise überwachen Sie die Erreichbarkeit einer Maschine bereits über die sogenannten Host-Checks, die in der Maschinen-Definition festgelegt sind (siehe Rezept 4.1, *Maschinen einbinden (host)*). Als Vorgabe, und typischerweise auch in der Praxis, ist dies ein `Ping`, also ein `ICMP Echo Request`. In vielen Fällen ist dies auch bereits ausreichend, aber wenn die fragliche Maschine mehrere Netzwerkschnittstellen besitzt, können Sie diese jeweils separat auf ihre Erreichbarkeit hin überwachen. Sie können hierzu check_ping (siehe Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*) oder check_icmp (Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)*) einsetzen.

Wenn Sie Schnittstellen mit unterschiedlichen IP-Adressen haben, die von der überwachenden Maschine aus erreichbar sind, können Sie die Ersetzung der Adresse allerdings nicht mehr wie gewohnt über das Makro $HOSTADDRESS$ vornehmen, denn dabei wird schließlich immer die gleiche Adresse eingesetzt. Sie sollten sich deshalb eine Kommandodefinition anlegen, die die Zieladresse nicht wie üblich über das Makro $HOSTADDRESS$ setzt, sondern diese als Parameter in der Service-Definition erwartet, wie im folgenden Beispiel für check_ping:

```
# 'check_ping' command definition
define command{
        command_name                    check_ping_ip
        command_line                    $USER1$/check_ping -H $ARG1$ -w $ARG2$ \
                                        -c $ARG3$ -p 5
}
```

Auf diese Weise können Sie über einen weiteren Parameter in den Dienstdefinitionen dann jeweils die abzufragende IP-Adresse übergeben, ohne zusätzliche Maschinendefinitionen anzulegen:

```
define service{
        service_description             PING-eth1
        use                             template-srv
        check_command                   check_ping_ip!10.85.58.1!100.0,20%! \
                                        500.0,60%
        host_name                       hcksp0
}
```

Wenn Ihre Schnittstellen von der überwachenden Maschine aus nicht erreichbar sind, hilft dies allerdings auch nicht weiter. Dies ist regelmäßig der Fall, etwa weil mehrere physikalische Schnittstellen zu einer logischen zusammengefasst sind (IEEE802.11ad Link Aggregation, unter Linux auch bonding genannt) oder weil es sich bei den Schnittstellen um Verbindungen zu einem anderen Netzwerk handelt. Sie können stattdessen dann ersatzweise die Status der Schnittstellen testen. Bei der Nutzung von SNMP können Sie dazu das Plugin check_ifoperstatus (siehe Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*) verwenden. Übergeben Sie dabei jeweils, also pro Schnittstelle mit jeweils einer eigenen Dienstdefinition, den Namen der Schnittstelle mit dem Parameter -d, wie hier gezeigt:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ifoperstatus -v 3 -d eth1 -H 10.85.58.1
-U icinga -a SHA1 -A "PASSPHRASE" -P AES -X "PASSPHRASE" -L authPriv
OK: Interface eth1 (index 2) is up.
```

Wenn Sie die im Rezept vorgesehene Kommandodefinition für SNMP Version 3 verwenden, können Sie diesen Test mit der folgenden Dienst-Definition einbinden:

```
define service {
        service_description             IfOperStatus
        use                             template-srv
        check_command                   check_ifoperstatus_v3!eth1
        host_name                       hcksp0
}
```

Alternativ oder zusätzlich zu check_ifoperstatus für einzelne Schnittstellen können Sie check_ifstatus (siehe Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen*

(check_ifstatus)) als Zusammenfassung für alle Schnittstellen einsetzen. Auf diese Weise können Sie bei vielen Schnittstellen den Status überwachen, ohne die Schnittstellen alle einzeln einpflegen zu müssen. Bei einer Maschine mit mehreren Schnittstellen sieht dies dann beispielsweise so aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ifstatus -v 3 -H 10.85.58.1 -U icinga
-a SHA1 -A "PASSPHRASE" -P AES -X "PASSPHRASE" -L authPriv
OK: host '10.85.58.1', interfaces up: 12, down: 0, dormant: 0, excluded: 0, unused: 0 |up
=12,down=0,dormant=0,excluded=0,unused=0
```

Als Service-Einbindung für diesen Test können Sie dann etwa die folgende Definition verwenden, sofern Sie die im Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)* vorgesehende Befehlsdefinition für SNMP Version 3 mit bereits hinterlegten Zugangsdaten verwenden:

```
define service {
       service_description                    IfStatus
       use                                    template-srv
       check_command                          check_ifstatus_v3
       host_name                              hcksp0
}
```

Beachten Sie, dass wir hier jeweils auf eine Vorlage `template-srv` referenzieren und die Prüfung der Maschine `hcksp0` zuweisen. Denken Sie daran diese Referenzen gegebenenfalls an Ihre Anforderungen anzupassen.

Laufzeit überwachen

Sie können über die Laufzeit (Uptime) testen, wie weit der letzte Neustart des überwachten Systems zurückliegt. Ist das System beispielsweise während eines Wartungsintervalls oder zwischen zwei Checks ungeplant neu gestartet, dann sollten Sie vielleicht darüber informiert werden. Die Laufzeit können Sie wie folgt über SNMP (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*) abfragen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_snmp -P 3 -L authNoPriv -U "icinga"
-a SHA -A "PASSPHRASE" -H 10.85.58.1 -o .1.3.6.1.2.1.25.1.1.0
SNMP OK - Timeticks: (2066812018) 239 days, 5:08:40.18 |
```

Wenn Sie die in Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)* vorgeschlagenen Befehlsdefinitionen für SNMP Version 3 mit hinterlegten Zugangsdaten konfiguriert haben, können Sie den Check mit folgender Dienstdefinition einbinden:

```
define service{
       service_description                    Uptime snmp3
       use                                    template-srv
       check_command                          check_snmp3!.1.3.6.1.2.1.1.3.0
       hostgroup_name                         hg-snmpd
}
```

Sofern Sie Performancedaten benötigen, können Sie diese mit Hilfe eines eigenen Wrapper-Skripts (Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*) nachpflegen. Dies hat den Vorteil, dass Sie beispielsweise durch Hardwaredefekt ausgelöste, ungeplante Neustarts sehr einfach zurückverfolgen können.

Systemzeit: Ergänzend sollten Sie auch erwägen, die Systemzeit zu kontrollieren. Sie können hier auf die in den Nagiosplugins enthaltenen Plugins check_ntp beziehungsweise die aktuelleren check_ntp_time und check_ ntp_peer zurückzugreifen:

```
icinga@moni:~$ /usr/local/nagios/libexec/check_ntp_time -H pool.ntp.org
-w 0.5 -c 1
NTP OK: Offset -0.0002855062485 secs|offset=-0.000286s;0.500000;1.000000;
```

Prozessorlast überwachen

Die CPU-Last ist auf jedem Server relevant und steht bei Nutzung von SNMP bereits zum Abruf bereit. Sie können das Manubulon-Plugin check_snmp_load verwenden, um die Prozessor-Auslastung abzufragen. Sie können dabei zwischen prozentualer Auslastung in der letzten Minute (dies ist die Vorgabe)

```
icinga@moni:~$ /usr/local/icinga/libexec-manubulon/check_snmp_load.pl -H 10.85.58.1
-l icinga -L sha,aes -x "PASSPHRASE" -X "PASSPHRASE" -w 80 -c 90 -f
8 CPU, average load 19.4% < 80% : OK | cpu_prct_used=19.375%;80;90
```

und bei Linux-Maschinen dem Load (über die Option `-T netsl`) mit der durchschnittlichen Systemlast für 1, 5 und 15 Minuten wählen. Dabei sorgt der Parameter `-f` dafür, dass Performancedaten ausgegeben werden:

```
icinga@moni:~$ /usr/local/icinga/libexec-manubulon/check_snmp_load.pl -H 10.85.58.1
-l icinga -L sha,aes -x "PASSPHRASE" -X "PASSPHRASE" -w 6,4,4 -c 8,6,6 -T netsl -f
Load : 4.45 3.51 3.38 : OK | load_1_min=4.45;6;8 load_5_min=3.51;4;6 load_15_min=3.38;4;6
```

Der Vorteil der ersten Variante besteht darin, dass diese Abfrage in der Form auf Linux und auf Windows-System sehr gut funktioniert. Eine passende Dienstdefinition unter Verwendung einer Vorlage `template-srv` und Prüfung aller Maschinen aus der Maschinengruppe `hg-snmpd` sieht dann wie folgt aus:

```
define service{
       service_description                    check_snmp_load
       use                                    template-srv
       check_command                          check_snmp_load!80!95
       hostgroups                             hg-snmpd

}
```

Die Kommandodefinition `check_snmp_load` (siehe Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins* und Rezept 4.3, *Befehle hinzufügen (command)*) mit hinterlegten Zugangsdaten ist dann folgende:

```
define command{
       command_name                           check_snmp_load
       command_line                           $USER2$/check_snmp_load.pl \
                                              -H $HOSTADDRESS$ -l icinga -x $USER12$ \
                                              -L sha -T stand -f -w $ARG1$ -c $ARG2$
}
```

Ein automatisch generierter Graph mit der CPU-Last sieht dann wie in Abbildung 8-1 dargestellt aus.

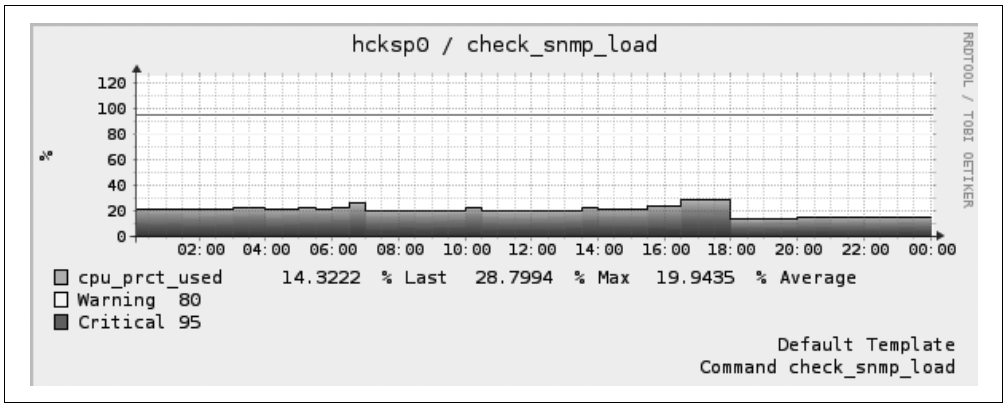


Abbildung 8-1: PNP4Nagios-Graph für Prozesslastdaten, die über check_snmp_load ausgelesen wurden.

Speicherauslastung überwachen

Als weiteren zu überwachenden Wert empfehlen wir Ihnen die Auslastung des Arbeitsspeichers. Da der Zugriff auf Festplattenspeicher um eine Zehnerpotenz langsamer als auf RAM-Speicher erfolgt, sollten Sie darauf achten, dass Daten nur selten ausgelagert werden müssen. Jedoch ist die tatsächliche Bedeutung der Speicherauslastung primär von der Nutzung der Maschine abhängig.

Da die entsprechenden Daten bei Nutzung von SNMP bereits vorliegen, empfehlen wir Ihnen eine Überwachung. Sie können hierzu das Manubulon-Plugin check_snmp_mem.pl einsetzen, welches gleichzeitig die Daten für RAM- und Swap-Speicher ausliest und zusammengefasst zurückgibt. Deshalb sind hier zwei Schwellenwerte jeweils für `WARNING` und `CRITICAL` erforderlich:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_mem.pl -H 10.85.58.1 -l icinga
-L sha,aes -x "PASSPHRASE" -X "PASSPHRASE" -w 80,85 -c 90,95 -f
Ram : 55.91%, Swap : 0.74% : | ram_used=9380372480B;13420986368;15098609664;0;16776232960
 shared=61820928B;;;0;16776232960 cache=7238672384B;;;0;16776232960 unused=95367168B;;;0;
16776232960 swap_used=253456384B;29205770650;32641743667;0;34359730176
```

Um das Plugin als Service einzubinden, können Sie dann etwa die folgende Definition nutzen:

```
define service{
       service_description                    check_snmp_mem
       use                                    template-srv
       check_command                          check_snmp_mem!85,50!95,75
       hostgroups                             hg-snmpd
}
```

Wir setzen hier eine Kommandodefinition `check_snmp_mem` mit hinterlegten Zugangsdaten wie im Rezept beschrieben voraus. Durch den Parameter `-f` veranlassen Sie, dass Perfor-

mancedaten ausgegeben werden. Ein automatisch generierter Graph sieht dann beispielsweise wie in Abbildung 8-2 dargestellt aus.

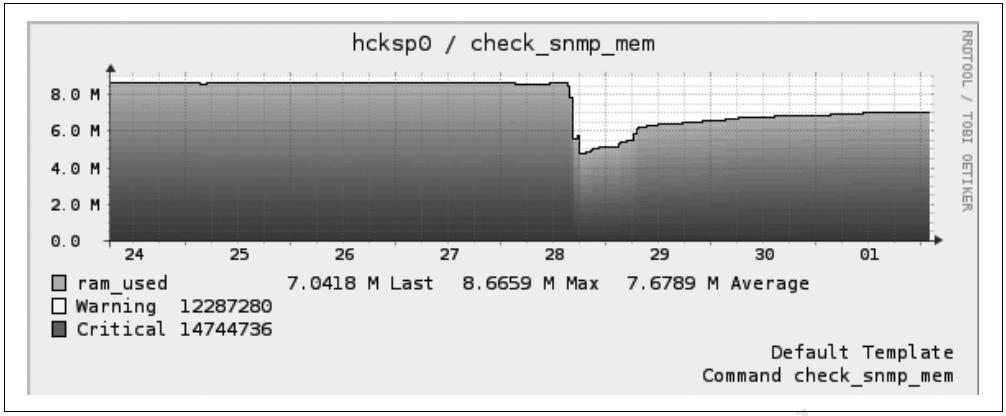


Abbildung 8-2: PNP4Nagios-Graph für Speicherauslastungsdaten, die über check_snmp_mem ausgelesen wurden.

Datenträgerauslastung überwachen

Auch bei Systemen, die nicht primär zur Speicherung von Daten gedacht sind, sollten Sie die Überwachung der Datenträgerauslastung in Betracht ziehen. Bei der Nutzung von SNMP stehen die entsprechenden Daten bereits für Sie bereit, so dass Sie sie lediglich noch abrufen müssen. Hierzu können Sie das Manubulon-Plugin check_snmp_storage.pl verwenden, indem Sie über die Option `-m` den Namen der Einbindung (engl. Mountpoint) angeben, beispielsweise für das Wurzelverzeichnis. Dabei sorgt der Parameter `-f` dafür, dass Performancedaten ausgegeben werden:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_storage.pl -H 10.85.58.1 -l icinga
-L sha,aes -x "PASSPHRASE" -X "PASSPHRASE" -m "/" -r -w 50 -c 90 -f
/: 18%used(23GB/126GB) (<50%) : OK | '/'=24986771456B;67641264128;121754275430;0;13528252
8256
```

Als entsprechende Dienstdefinition können Sie dann beispielsweise die folgende verwenden:

```
define service{
       service_description                    check_fs-root
       use                                    template-srv
       check_command                          check_snmp_storage!'/$'!50!90
       hostgroup_name                         hg-snmpd
}
```

Eine Auswertung der Performancedaten über einen Grapher sieht dann wie in Abbildung 8-3 aus.

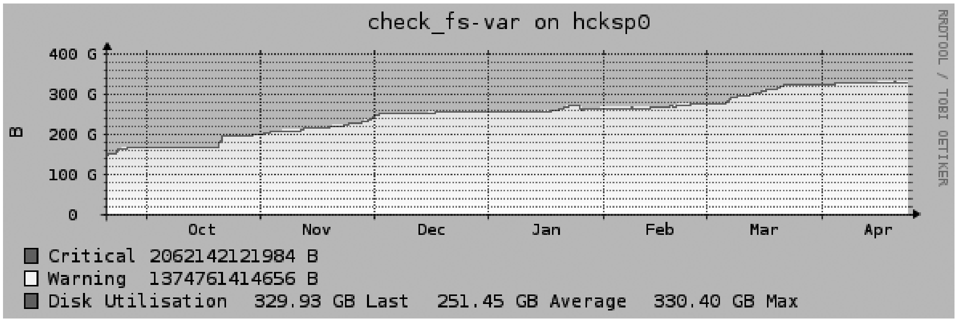


Abbildung 8-3: Graph für Datenträgerauslastungsdaten, die über check_snmp_storage ausgelesen wurden.

Datenträgerstatus überwachen mit SMART

Zumindest auf Maschinen, die viele Daten auslesen und/oder schreiben, sollten Sie neben der Auslastung auch den Datenträgerstatus überwachen. Dazu gibt es die S.M.A.R.T. (Self-Monitoring, Analysis and Reporting Technology; häufig auch nur SMART), die heute eigentlich von allen Festplatten unterstützt wird. Unter Unix/Linux können Sie hierzu das in den Standard-Plugins enthaltene check_ide_smart verwenden, dass allerdings auf dem jeweiligen System installiert sein muss (siehe Rezept 5.5, *Festplatten-Status überwachen (check_ide_smart)*). Wenn Sie die Einbindung über SNMP wie in Rezept 2.5, *Plugins unter Linux über SNMP ausführen* vorgenommen haben und das Wrapper-Script aus Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen* verwenden, können Sie das Plugin also etwa so einbinden:

```
define service {
       service_description                  check_snmp_smart_sda
       use                                  template-srv
       command                              wrapper_check_snmp_plugin.sh! \
                                            check_smart_sda
       hostgroups                           hg-snmpd
}
```

Sie benötigen dabei für jede Festplatte (`sda`, `sdb` und so weiter) eine eigene Definition. Die Abfrage für die erste (typischerweise eben `sda`) können Sie dabei vermutlich wie hier vorgesehen für die Gruppe aller über SNMP eingebundenen Maschinen durchführen. Wenn Ihre Server aber nicht alle die gleiche Anzahl an Platten aufweisen, müssen Sie weitere Platten entweder für jeden Server einzeln oder über entsprechende Gruppen für weitere Platten einbinden.

Prozesse überwachen

Sowohl die CPU-Last als auch die Speicherauslastung sind dabei aggregierte Werte, die sich auf einzelne Prozesse verteilen. Sie können Prozesse mit Hilfe des Plugins check_procs (siehe Rezept 5.3, *Prozesse überwachen (check_procs)*) zusätzlich gezielt auf

bestimmte Merkmale testen. Generell für alle Maschinen empfehlen wir Ihnen die Überwachung der Prozessanzahl, wie im Rezept beschrieben:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_procs
PROCS OK: 90 processes
```

Wenn Sie eine Einbindung von check_procs über SNMP auf dem Zielrechner wie in Rezept 2.5, *Plugins unter Linux über SNMP ausführen* vorgenommen, haben und unser Wrapper-Script aus Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen* verwenden, können Sie das Plugin wie folgt in Ihr Monitoring-System einbinden:

```
define service {
       service_description                    check_snmp_procs
       use                                    template-srv
       command                                wrapper_check_snmp_plugin.sh!check_procs
       hostgroups                             hg-snmpd
}
```

Wenn Sie dabei auch prüfen möchten, ob einzelne Prozesse mehr als 10 beziehungsweise 20 Prozent der Systemlast verursachen, könnten Sie den Check wie folgt modifizieren:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_procs -w 10 -c 20 --metric=CPU
CPU OK: 90 processes
```

Der folgende Check prüft, ob der Prozess ido2db genau einmal gestartet wurde:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_procs -c 1:2 -C ido2db -p 1
PROCS OK: 1 process with command name 'ido2db', PPID = 1
```

Ein weiterer hilfreicher Check ist die Prüfung auf Prozesse im Status zombie. Wenn Sie den Status ab einem entsprechenden Prozess auf WARNING und bei mehr als fünf auf CRITICAL setzen möchten, funktioniert das also etwa folgendermaßen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_procs -w 1 -c 5 -s Z
PROCS OK: 0 processes with STATE = Z
```

Netzwerkverkehr überwachen

Eine erweiterte Überwachung der Erreichbarkeit können Sie durchführen, indem Sie auch die Auslastung der Netzwerk-Schnittstellen überwachen. Bei der Nutzung von SNMP können Sie hierzu beispielsweise das Manubulon-Plugin check_snmp_int verwenden (siehe Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*). Um etwa die Schnittstelle mit dem Namen eth0 abzufragen und prozentuale Werte über die Auslastung zu erhalten, können Sie einen Aufruf wie folgt mit den Parametern -f, -k und -y verwenden:

```
icinga@moni:~$ /usr/local/icinga/libexec-manubulon/check_snmp_int.pl -H 10.85.58.1
-l icinga -L sha,aes -x "PASSPHRASE" -X "PASSPHRASE" -n eth0 -k -f -w 50,50 -c 90,90
-y eth0:UP (1.1KBps/1.1KBps):1 UP: OK | 'eth0_in_prct'=0%;50;90;0;100
'eth0_out_prct'=0%;50;90;0;100
```

Eine entsprechende Einbindung in Nagios/Icinga erreichen Sie aufbauend auf der im Rezept vorgeschlagenen Befehlsdefinition mit folgender Dienstdefinition:

```
define service{
        service_description                     check_snmp_int-eth0-bw
        use                                     template-srv
        check_command                           check_snmp_int!eth0!-k -f -y -w 50,50 \
                                                -c 90,90
        hostgroups                              hg-snmpd

}
```

Alternativ können Sie die Option `-y` durch `-Y` ersetzen und so anstelle prozentualer Werte die tatsächlichen Werte abfragen, zum Beispiel folgendermaßen:

```
icinga@moni:~$ /usr/local/icinga/libexec-manubulon/check_snmp_int.pl -H 10.85.58.1
-l icinga -L sha,aes -x "PASSPHRASE" -X "PASSPHRASE" -n eth0 -f -k -Y -w 50,50 -c 90,90
eth0:UP (1.2KBps/1.0KBps):1 UP: OK | 'eth0_in_Bps'=1195;51200;92160;0;1000000000 'eth0_ou
t_Bps'=1013;51200;92160;0;1000000000
```

Wie im Aufruf auf der Kommandozeile können Sie diese Option auch in der Dienstdefinition ändern. Die Auswertung der Performancedaten könnte dann wie in Abbildung 8-4 dargestellt aussehen.

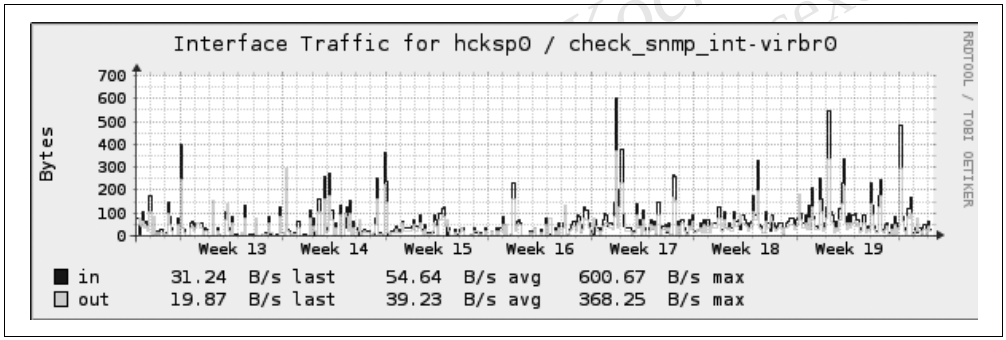


Abbildung 8-4: Graph für Netzwerkverkehrsdaten, die über check_snmp_int ausgelesen wurden.

Umweltparameter überwachen (Temperatur, Lüfter, Netzteil)

Bei Linux-Maschinen ist das Paket `lm-sensors` (unter Debian/Ubuntu beziehungsweise `sensors` unter SLES/OpenSuSE und `lm_sensor` unter RHEL/CentOS) für das Auslesen von Sensordaten für Umweltparameter zuständig. Sofern Ihre Hardware von diesem Paket unterstützt wird, können und sollten Sie die verfügbaren Werte überwachen. Das Plugin check_sensors (siehe Rezept 5.6, *Sensor-Daten überwachen (check_sensors)*) ist genau hierfür ausgelegt. Um es auf einer entfernten Maschine abzufragen, müssen Sie das Paket dort installieren, um es dann beispielsweise wie in Rezept 2.5, *Plugins unter Linux über SNMP ausführen* beschrieben einzubinden. Wenn Sie unser Wrapper-Script aus Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen* verwenden, können Sie es dann etwa so einbinden:

```
define service {
       service_description                    check_snmp_sensors
       use                                    template-srv
       command                                wrapper_check_snmp_plugin.sh! \
                                              check_sensors
       hostgroups                             hg-snmpd
}
```

Beachten Sie bitte, dass wir hier über die entsprechende Maschinengruppe wieder die Einbindung für alle SNMP-Server vorsehen. Wenn Ihre Maschinen nicht alle vom Sensor-Paket unterstützt werden, müssen Sie sie stattdessen entweder jeweils einzeln einbinden oder entsprechende zusätzliche Gruppen definieren.

Die im Rezept 5.6, *Sensor-Daten überwachen (check_sensors)* beschriebene Abfrage einzelner Werte, um daraus Graphen zu erzeugen, wird Ihnen bei der Einbindung über Maschinengruppen allerdings Probleme bereiten, wenn die Namen der Sensoren nicht identisch sind. Wir empfehlen Ihnen sich gegebenenfalls Gruppen für die entsprechenden Varianten anzulegen und diese für die Einbindung zu nutzen.

Aktualisierungen überwachen

Bei vielen Distributionen wird Software heute über sogenannte Paketmanager verwaltet. Sofern Sie eine auf Debian basierende Distribution verwenden, ist dies der Paketmanager apt beziehungsweise aptitude, zu dem in den Standard-Plugins ein entsprechender Test für Aktualisierungen enthalten ist (siehe Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*). Binden Sie diese ein, um über Ihr Monitoring-System hinsichtlich eventuell ausstehender Aktualisierungen informiert zu werden. Bei Einbindung über SNMP wie in Rezept 2.5, *Plugins unter Linux über SNMP ausführen* beschrieben und unter Verwendung unseres Wrapper-Scripts aus Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen* können Sie das Plugin dann etwa so einbinden:

```
define service {
       service_description                    check_snmp_apt
       use                                    template-srv
       command                                wrapper_check_snmp_plugin.sh!check_apt
       hostgroups                             hg-snmpd-debian
}
```

Wir haben bei der Einbindung hier eine eigene Maschinengruppe für auf Debian basierende Maschinen (Debian, Ubuntu) gewählt. Wie in Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)* erwähnt gibt es auch entsprechende Plugins für Systeme, die auf SuSE (`check_zypper`) beziehungsweise RedHat (`check_updates`) basieren. Wenn Sie mit einer heterogene Systemlandschaft arbeiten, sollten Sie sich entsprechend zusätzliche Gruppen für diese anlegen und dort die entsprechenden Plugins einbinden.

Diskussion

Sie werden also typischerweise eine ganze Menge Prüfungen prinzipiell für alle Unix/Linux-Maschinen durchführen. Um den Aufwand der Einbindung möglichst gering zu halten, sollten Sie dabei natürlich nicht etwa jeden Dienst einzeln definieren. Arbeiten Sie stattdessen mit Maschinengruppen und definieren Sie so für Ihre Umgebung zugeschnittene Vorgabe-Konfigurationen. Für die hier beschriebenen Plugins und Einbindung unter SNMP können Sie dabei wie folgt vorgehen:

1. Definieren Sie Maschinengruppen (Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*) entsprechend der bei Ihnen vorhandenen Untergruppen. Dies könnte dann beispielsweise zunächst wie hier von uns verwendet `hg-snmpd` für alle Maschinen mit SNMP sein. Um gezielt auf Aktualisierungen auf Maschinen mit Debian zu prüfen, legen Sie sich dann des Weiteren eine Gruppe `hg-snmpd-debian` für alle Debian-Maschinen mit SNMP an, um diese etwa von Maschinen mit SuSE oder RedHat zu unterscheiden.
2. Definieren Sie die gewünschten Dienste (Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) und referenzieren Sie dabei jeweils die passende Maschinengruppe.
3. Bereiten Sie alle einzubindenden Maschinen durch Installation und Konfiguration von SNMP (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*) sowie gegebenenfalls erforderlicher Plugins (wie in den Installationsrezepten in Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten* beschrieben) vor. Die Konfigurationsdatei für SNMP können Sie dabei gegebenenfalls einfach kopieren und so von einer Vorlage übernehmen.
4. Ordnen Sie Ihre Maschinen bei deren Einbindung (siehe Rezept 4.1, *Maschinen einbinden (host)*) den passenden Maschinengruppen zu, damit alle in Schritt zwei definierten Prüfungen automatisch auf diesen ausgeführt werden.

Dabei legen Sie einen Standardumfang an Tests fest, die Sie auf allen Maschinen ausführen möchten. Nehmen wir einmal an, Sie möchten diese für alle oben aufgeführten Plugins durchführen. Zur Vorbereitung auf den zu überwachenden Maschinen sind dann folgende Schritte erforderlich:

- Installation von `snmpd` (etwa `apt-get install snmpd`)
- Konfiguration von `snmpd` (durch Kopieren einer einmal erstellten Vorlage)
- Installation der Plugins (etwa `apt-get install nagios-plugins`)

Gerade wenn Sie dabei auf allen Maschinen eine vorgefertigte Konfigurationsdatei für den `snmpd` verwenden, können Sie diese Schritte schnell und ohne großen Aufwand auch auf vielen Maschinen durchführen (gegebenenfalls sogar automatisiert). Wenn Sie dann eine Maschinendefinition anlegen, referenzieren Sie über die Maschinengruppe automatisch den von Ihnen festgelegten Satz an Standardprüfungen.

Ein analoges Vorgehen ist möglich, wenn Sie die Plugins statt über SNMP bevorzugt über NRPE oder SSH einbinden möchten. Nur installieren Sie in diesem Fall jeweils das äquivalente Programm und kopieren die entsprechende Konfigurationsdatei. Bei SSH ist dies etwa die Datei `authorised_hosts` inklusive der darin enthaltenen Befehlsaufrufe. Wir sind hier spezifisch auf die SNMP-Variante eingegangen, da diese einen Standard für genau diesen Anwendungsfall darstellt, für den es in der Praxis (gerade für Version 3) bislang wenig Dokumentation gibt. Wir möchten an dieser Stelle noch kurz einen Vergleich dieser Zugangsarten anstellen.

Bei NRPE zeigte sich in den von uns genutzten Versionen die Verschlüsselung als problematisch, da sich nicht ohne Weiteres eigene Zertifikate einbinden lassen. Dies ist aus prinzipiellen Überlegungen inakzeptabel, auch wenn der Einsatz in geschlossenen Umgebungen in der Praxis durchaus möglich ist. SSH ist bezüglich der Sicherheit eher unbedenklich. Allerdings wird SSH typischerweise auch für andere Zwecke als das Monitoring genutzt (eben den gesicherten Shell-Zugang), weshalb die Konfigurationsdatei dafür nicht einfach überschrieben werden kann. Im Gegensatz dazu ist SNMP in erster Linie für das Monitoring gedacht. Da wir es auf von uns betreuten System ausschließlich zu diesem Zweck einsetzen, können wir neue Konfigurationen durch einfaches Ersetzen der Dateien verteilen. Bei SSH müssten wir hingegen jeweils prüfen, welche der SSH-Schlüssel für das Monitoring und welche für den administrativen Zugang genutzt werden sollen, was die Handhabung von SNMP im Vergleich dazu einfacher erscheinen lässt. Allerdings ist für die Abfrage der Plugins über SNMP noch kein standardisiertes Plugin vorhanden, weshalb wir das hier vorgestellte Wrapper-Script nutzen.

Siehe auch

- Einrichtung von SNMP auf Linux-Servern, Einbindung von Plugins darüber und Zugriff darauf: Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*, Rezept 2.5, *Plugins unter Linux über SNMP ausführen*, Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*
- Einrichtung von SSH auf Linux-Servern und Zugriff vom Monitoring-System aus: Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten* und Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*
- Einrichtung von NRPE auf Linux-Servern und Zugriff darauf: Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten* und Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*
- Rezept zur Vereinfachung der Konfiguration mit Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Rezept zur Zusammenfassung von Maschinen in Maschinengruppen: Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*
- Rezepte zu Maschinen, Dienst- und Befehlsdefinitionen: Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* und Rezept 4.3, *Befehle hinzufügen (command)*

- Rezepte zu Erreichbarkeitsprüfungen: Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)* und Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)*
- Rezepte zu den Schnittstellenprüfungen: Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)* und Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*
- Rezept zur Einbindung der Manubulon-Plugins: Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*
- Rezepte zur lokalen Abfrage von Werten: Rezept 5.3, *Prozesse überwachen (check_procs)*, Rezept 5.5, *Festplatten-Status überwachen (check_ide_smart)*, Rezept 5.6, *Sensor-Daten überwachen (check_sensors)* und Rezept 5.11, *Überwachen auf Aktualisierungen des Systems (check_apt)*

8.2 Überwachung von Windows-Maschinen

Problem

Sie möchten Windows-Maschinen überwachen, sind sich aber unsicher, welche Parameter Sie einbeziehen können und sollten.

Lösung

Wir fassen im Folgenden für Sie zusammen, welche Werte wir Ihnen für eine Beobachtung von Windows-Maschinen im allgemeinen empfehlen. Wir gehen dabei in diesem Rezept nur auf die allen Windows-Maschinen gemeinsamen Ziele ein. Für die Prüfung von spezifischen Serverdiensten beachten Sie bitte auch die Rezepte im hinteren Teil dieses Kapitels.

Der in fast allen MS-Windows Versionen enthaltene SNMP-Server unterstützt nur die älteren Protokollversionen 1 und 2c und wird voraussichtlich langfristig nicht weiter unterstützt. Bereits in Windows Server 2012 muss der Dienst seperat aktiviert werden (siehe Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*). Im Folgenden beschreiben wir deshalb neben SNMP auch die Nutzung von NSClient++ (siehe Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*). NSClient und NRPE (Nagios Remote Plugin Executor) arbeiten ähnlich wie `check_by_ssh` (siehe Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*). Ein Kommando wird durch `check_nt` (siehe Rezept 5.9, *Überwachung einer Windows-Maschine mit NSClient (check_nt)*) beziehungsweise `check_nrpe` (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*) beim Zielrechner abgesetzt und die Ausgabe wird an den Monitoring-Server zurückgeliefert.

Beim Einsatz von NSClient++ gibt es dabei unterschiedliche Varianten zur Abfrage von Werten:

1. NSClient (check_nt)

   Enthält nur wenige Checks und wird nicht weiterentwickelt, lässt sich aber sehr leicht einbinden. Die Unterstützung diesen Dienstes dient allerdings in erster Linie der Rückwärtskompatibilität.

2. NRPE (check_nrpe)

   Dies ist die von NSClient++ bevorzugte Methode.

3. NSClient und NRPE

   Die Kombination von 1) und 2) dient in erster Linie der Rückwärtskompatibilität.

4. NSCA-Server

   Bei NSCA (Nagios Service Check Acceptor) handelt es sich um eine passive Methode, die hauptsächlich für verteiltes Monitoring eingesetzt wird (siehe dazu Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*). Diese wird in im vorliegenden Buch nicht weiter erläutert.

Wir empfehlen Ihnen zur Überwachung eines Windows-Servers auf Variante 2 oder Variante 3 zurückzugreifen. Für die Nutzung von NSClient setzen Sie das Plugin `check_nt` ein (siehe Rezept 5.9, *Überwachung einer Windows-Maschine mit NSClient (check_nt)*) und für NRPE verwenden Sie das Plugin `check_nrpe` (siehe Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*).

Die Abfrage über NSClient bietet Ihnen lediglich einige Basischecks. Die Entwickler von NSClient++ empfehlen Ihnen langfristig den Umstieg auf NRPE. Der Vorteil von NSClient besteht allerdings darin, dass es meist auf Anhieb funktioniert. NSClient lässt sich sehr einfach installieren, ist ausgereift, weit verbreitet und funktioniert auf allen Windows-Varianten seit Windows NT.

NRPE erfordert, gerade für komplexere Checks, einiges an Konfigurationsarbeit. Der Vorteil ist der, dass NRPE wesentlich flexibler ist als NSClient und daher langfristig die bevorzugte Methode sein sollte. Es ist zu erwarten, dass die Entwicklung an NSClient eingestellt wird. Wir empfehlen Ihnen für den Einstieg, beide Methoden in Kombination einsetzen, wie in Abbildung 8-5 dargestellt. Bei Bedarf können Sie später immer noch vollständing auf NRPE migrieren.

Performancedaten: Wenn der von Ihnen genutzte Check Performancedaten ausgibt, können Sie diese auch mit Hilfe von PNP4Nagios auswerten lassen. Sie erhalten dann zum Check automatisch einen entsprechenden Graphen.

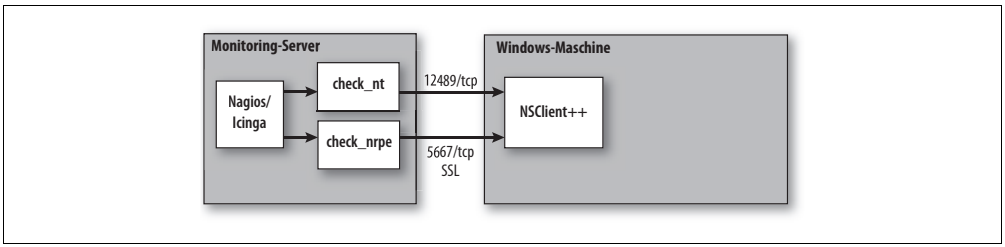


Abbildung 8-5: Der NSClient++-Agent unterstützt Abfragen mit check_nt (NSClient) und check_nrpe (NRPE)

Erreichbarkeit überwachen

Nagios/Icinga überwacht die Erreichbarkeit der Maschine bereits automatisch nach dem Einpflegen über die sogenannten Host-Checks, die in der Maschinen-Definition festgelegt sind (siehe Rezept 4.1, *Maschinen einbinden (host)*). Vorgabe ist typischerweise ein `Ping`, also ein `ICMP Echo Request`. Meist reicht dies aus, sofern die Maschine nicht mehrere Netzwerkschnittstellen besitzt. Sie sollten deren Erreichbarkeit dann jeweils separat überwachen. Dazu können Sie die Plugins `check_ping` (siehe Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*) oder `check_icmp` (Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)*) einsetzen.

`ICMP Echo Request` aktivieren: Auf Windows-Systemen sind `ICMP Echo Requests` zunächst deaktiviert. Mit folgendem Kommandozeilenbefehl können Sie diese explizit zulassen:

```
C:\Windows\system32>netsh advfirewall firewall add rule name="ICMP Allow
incoming V4 echo request" protocol=icmpv4:8,any dir=in action=allow
Ok.
```

Bei unterschiedlichen IP-Adressen die einer zu überwachenden Maschine zugeordnet sind, können Sie die Ersetzung der Adresse allerdings jetzt nicht mehr über das Makro `$HOSTADDRESS$` vornehmen, da dieses schließlich immer die gleiche Adresse einsetzten würde. Deshalb sollten Sie eine Kommandodefinition anlegen, die die Zieladresse als Parameter in der Service-Definition erwartet.

Für `check_icmp` könnte die Kommandodefinition etwa wie folgt aussehen:

```
# 'check_icmp' command definition
define command{
       command_name                    check_icmp_ip
       command_line                    $USER1$/check_icmp -H $ARG1$ -w $ARG2$ \
                                       -c $ARG3$ -n 5
}
```

So können Sie über einen weiteren Parameter in den Dienstdefinitionen dann jeweils die abzufragende IP-Adresse übergeben, ohne zusätzliche Maschinendefinitionen anzulegen:

```
define service{
       service_description             PINGi-eth1
       use                             template-srv
```

```
        check_command                   check_icmp_ip!10.85.58.111!100.0,20%! \
                                        500.0,60%
        host_name                       win2k8serv
}
```

Dies funktioniert allerdings nur dann, wenn die Schnittstellen der überwachenden Maschine erreichbar sind. Dies ist jedoch nicht der Fall, wenn beispielsweise mehrere physikalische Schnittstellen zu einer logischen zusammengefasst (IEEE802.11ad Link Aggregation, unter Windows auch `teaming` genannt) wurden oder weil die Schnittstellen eben mit einem nicht erreichbaren Netzwerk angebunden sind.

Alternativ können Sie dann die Status der Schnittstellen über SNMP testen (siehe Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*). Dazu können Sie das Plugin `check_ifoperstatus` (siehe Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*) verwenden. Diesem können Sie mit dem Parameter `-k` die Ordnungsnummer der gewünschten Schnittstelle mitgeben. Die Ordnungsnummern können Sie mit folgendem Kommando von einem Windows-System über SNMP abfragen:

```
icinga@moni:~$ snmpwalk -v 2c -c "COMMUNITY" 10.85.58.111 .1.3.6.1.2.1.2.2.1.2
IF-MIB::ifDescr.1 = STRING: Software Loopback Interface 1
[...]
IF-MIB::ifDescr.13 = STRING: Realtek RTL8139C+ Fast Ethernet NIC
[...]
```

Die entsprechende Abfrage über SNMP sieht dann wie folgt aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ifoperstatus -v 2 -C "COMMUNITY" -k "13"
-H 10.85.58.111
OK: Interface Realtek RTL8139C+ Fast Ethernet NIC (index 13) is up.
```

Sie können diesen Check mit Hilfe der folgenden Befehlsdefinition in Ihrem Monitoring-System einbinden

```
define command {
        command_name                    check_ifoperstatus_v2
        command_line                    $USER1$/check_ifoperstatus \
                                        -H $HOSTADDRESS$ -v 2 -C $USER11$ \
                                        -k $ARG1$ $ARG2$
}
```

und anschließend mit folgender Dienst-Definition einer Windows-Maschine, hier der Host `win7e`, zuordnen:

```
define service {
        service_description             IfOperStatus
        use                             template-srv
        check_command                   check_ifoperstatus_v2!13
        host_name                       win2k8serv
}
```

Wenn Ihnen eine Zusammenfassung für alle Schnittstellen ausreicht, können Sie alternativ das Plugin `check_ifstatus` (siehe Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)*) einsetzen. Dies ermöglicht Ihnen, den Status vieler Schnittstellen zu überwachen, ohne diese alle einzeln einpflegen zu müssen.

Alternativ oder zusätzlich zu `check_ifoperstatus` für einzelne Schnittstellen können Sie `check_ifstatus` (siehe Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)*) als Zusammenfassung für alle Schnittstellen einsetzen. So können Sie bei vielen Schnittstellen den Status überwachen, ohne die Schnittstellen alle einzeln einpflegen zu müssen. Bei einer Maschine mit mehreren Schnittstellen, bei denen die nicht benötigten explizit ausgenommen sind (durch den Parameter `-u`), sieht dies dann beispielsweise so aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ifstatus -v 2 -C "COMMUNITY"
-H 10.85.58.111 -u 2,3,4,6,7,8,11,14,16,17,18
OK: host '10.85.58.111', interfaces up: 6, down: 0, dormant: 0, excluded: 0, unused: 11 |
up=6,down=0,dormant=0,excluded=0,unused=11
```

Als Service-Einbindung für diesen Test können Sie dann etwa die folgende Definition verwenden, sofern Sie die im Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)* vorgesehene Befehlsdefinition für SNMP Version 2 mit bereits hinterlegten Zugangsdaten verwenden:

```
define service {
       service_description                    IfStatus
       use                                    template-srv
       check_command                          check_ifstatus_v2
       host_name                              win2k8serv
}
```

Sie müssen die voranstehenden Servicevorlagen gegebenenfalls an Ihre Anforderungen anpassen. In diesem Fall weisen wir die Prüfung der Maschine `win2k8serv` zu.

Laufzeit überwachen

Als weiteren einfachen Test können Sie prüfen, wie lange der letzte Neustart zurückliegt. Wurde das System beispielsweise während eines Wartungsintervalls oder zwischen zwei Prüfungen ungeplant neu gestartet, sollten Sie vielleicht darüber informiert werden.

Am einfachsten lässt sich die Laufzeit über SNMP (siehe Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*) wie im Folgenden angeführt abfragen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_snmp -P 2c -C "COMMUNITY"
-H 10.85.58.111 -o .1.3.6.1.2.1.1.3.0
SNMP OK - Timeticks: (30209) 0:05:02.09 |
```

Sofern Sie Performancedaten benötigen, können Sie diese anschließend mit Hilfe eines eigenen Wrapper-Scripts (Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*) nachrüsten. Wenn Sie die in Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)* vorgeschlagenen Befehlsdefinitionen für SNMP Version 2 mit hinterlegten Zugangsdaten konfiguriert haben, können Sie den Check mit folgender Dienstdefinition einbinden:

```
define service{
       service_description                    Uptime snmp2
       use                                    template-srv-win
       check_command                          check_snmp2!.1.3.6.1.2.1.1.3.0
       hostgroup_name                         hg-win
}
```

Wenn Sie NSClient++ auf der Zielmaschine installiert haben (siehe Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*), können Sie die Laufzeit auch mit NSClient oder NRPE testen. Eine Abfrage über NSClient sieht dann wie folgt aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nt -H 10.85.58.110 -p 12489 -s PASSWORT
-v UPTIME
System Uptime - 40 day(s) 4 hour(s) 54 minute(s)
```

Sie kann mit einer entsprechende Dienstdefinition leicht eingebunden werden:

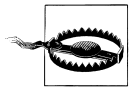
```
define service{
       service_description                    Uptime nsclient
       use                                    template-srv-win
       check_command                          check_nt!UPTIME
       hostgroup_name                         hg-win
}
```

Sie können den Check selbstverständlich auch mit NRPE durchführen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nrpe -H 10.85.58.113 -p 5666 -c alias_up
OK all counters within bounds.|'uptime'=445552000;3600000;0
```

Das hier genutzte Alias-Kommando `alias_up` wird bei NSClient++ mitgeliefert und sollte in Ihrer Konfigurationsdatei `nsclient.ini` bereits enthalten sein:

```
[/settings/external scripts/alias]
alias_up = checkUpTime MinWarn=1d MinWarn=1h
```



UPTIME via NRPE mit `alias_up`: Dieser Check funktioniert etwas anders als der Check via SNMP oder NSClient. Bei NRPE wird in diesem Beispiel geprüft, ob die Uptime <1Tag beziehungsweise <1h ist. Wenn dies der Fall ist, wird ein Alarm ausgelöst.

Sie erkennen an den durch das Pipe-Symbol »|« getrennten Rückgabewerten, dass die NRPE-Variante dieser Prüfung Performancedaten enthält. Damit erhalten Sie automatisch Graphen zu diesen Werten. Dies ist von Vorteil, da Sie auf diese Weise etwa durch Hardwaredefekt ausgelöste, ungeplante Neustarts sehr einfach zurückverfolgen können. Hier die entsprechende Dienstdefinition für Ihr Monitoring-System:

```
define service{
       service_description                    Uptime nrpe
       use                                    template-srv-win
       check_command                          check_nrpe!alias_up
       hostgroup_name                         hg-win
}
```

Prozessor-Last überwachen

Die Überwachung der CPU-Last ist ein Kernmerkmal laufender Maschinen und somit entsprechend immer relevant. Sie können bei Nutzung von SNMP das Manubulon-Plugin `check_snmp_load` verwenden, um die prozentuale Auslastung in der letzten Minute abzufragen. Der Vorteil dieser Methode besteht darin, dass das Plugin sowohl Windows- als auch Linux-Systeme abfragen kann. Dadurch lässt sich die Last auf beiden Systemen sehr gut vergleichen:

```
icinga@moni:~$ /usr/local/icinga/libexec-manubulon/check_snmp_load.pl -H 10.85.58.111
-C "COMMUNITY" -w 80 -c 90 -f
1 CPU, load 2.0% < 80% : OK | cpu_prct_used=2%;80;90
```

Eine passende Dienstdefinition sieht dann beispielsweise wie folgt aus:

```
define service{
       service_description                    Load snmpv2
       use                                    template-srv
       check_command                          check_snmp_load_v2!80!95
       hostgroups                             hg-win
}
```

Die verwendete Kommandodefinition `check_snmp_load_v2` (siehe dazu Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins* und Rezept 4.3, *Befehle hinzufügen (command)*) mit hinterlegten Zugangsdaten wäre dann:

```
define command{
       command_name                           check_snmp-load_v2
       command_line                           $USER2$/check_snmp_load.pl \
                                              -H $HOSTADDRESS$ -2 -C $USER11$ \
                                              -T stand -f -w $ARG1$ -c $ARG2$
}
```

Wenn Sie NSClient auf der Windows-Maschine installiert haben, können Sie die Auslastung auch über NSClient oder NRPE überwachen. Hier wird als Beispiel die durchschnittliche Last innerhalb eines 5-Minuten-Intervalls über NSClient ausgelesen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nt -H 10.85.58.113 -p 12489 -s PASSWORD
-v CPULOAD -l 5,80,90
CPU Load 24% (5 min average) | '5 min avg Load'=0%;80;90;0;100
```

Eine entsprechende Dienstdefinition sähe dann wie folgt aus:

```
define service{
       service_description                    CPU Load nisc
       use                                    template-srv-win
       check_command                          check_nt!CPULOAD!-l 5,80,90
       hostgroup_name                         hg-win
}
```

Bei der Nutzung von NRPE können Sie auf das bei NSClient++ in der Datei `nsclient.ini` vorkonfigurierte Alias-Kommando `alias_cpu` (siehe Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*) zurückgreifen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nrpe -H 10.85.58.113 -p 5666 -c alias_cpu
OK CPU Load ok.|'5m'=0%;80;90 '1m'=0%;80;90 '30s'=0%;80;90
```

Eine entsprechende Dienstdefinition wäre dann etwa:

```
define service{
       service_description                    CPU Load nrpe
       use                                    template-srv-win
       check_command                          check_nrpe!alias_cpu
       hostgroup_name                         hg-win
}
```

Ein automatisch generierter Graph mit der CPU-Last sieht dann wie in Abbildung 8-6 aus.

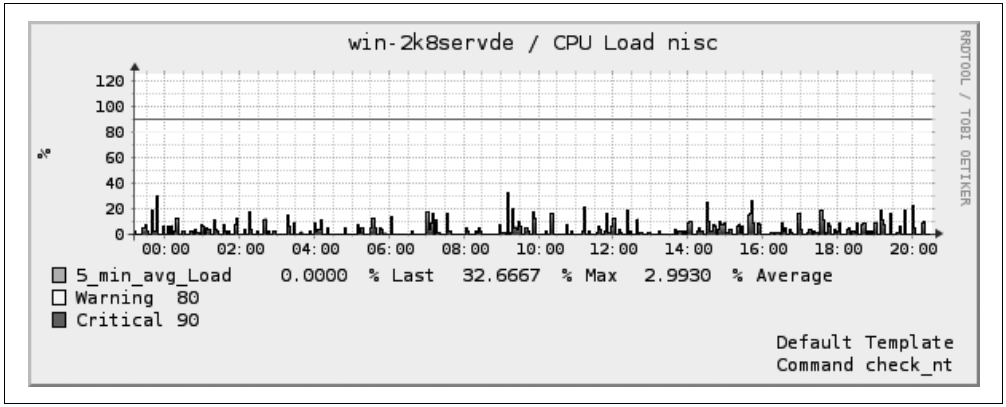


Abbildung 8-6: PNP4Nagios-Graph für über check_nt ausgelesene CPU-Daten

Speicherauslastung überwachen

Als weiteren essentiellen Wert empfehlen wir Ihnen die Überwachung der Speicherauslastung. Da Festplatten um eine Zehnerpotenz langsamer sind als RAM-Speicher, sollten Sie darauf achten, dass Daten nur selten auf Festplattenspeicher ausgelagert werden müssen. Um den Speicher mit SNMP auszulesen, können Sie das eigentlich für Datenträger gedachte Manubulon-Plugin `check_snmp_storage.pl` verwenden. Durch den Parameter `-m` können Sie zwischen physikalischem und virtuellem Speicher wählen; der Parameter `-f` veranlasst, dass Performancedaten ausgegeben werden:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_storage.pl -C "COMMUNITY"
-H 10.85.58.111 -m "Physical Memory" -w 80 -c 90 -f
Physical Memory: 22%used(884MB/4096MB) (<80%) : OK | 'Physical_Memory'=884MB;3276;3686;0;
4096

icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_storage.pl -C "COMMUNITY"
-H 10.85.58.111 -m "Virtual Memory" -w 80 -c 90 -f
Virtual Memory: 10%used(812MB/8189MB) (<80%) : OK | 'Virtual_Memory'=812MB;6552;7370;0;81
89
```

Als entsprechende Dienstdefinition für die Überwachung des virtuellen Speichers können Sie dann beispielsweise die folgende verwenden:

```
define service{
       service_description                    check_virtmem
       use                                    template-srv-win
       check_command                          check_snmp_storage!'Virtual Memory'!80!90
       hostgroup_name                         hg-win
}
```

Mit Hilfe von check_nt können Sie den physischen Speicher ebenfalls abfragen, hier zunächst ohne Schwellenwerte:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nt -H 10.85.58.113 -p 12489 -s PASSWORT
-v MEMUSE
Memory usage: total:4095.23 Mb - used: 766.34 Mb (19%) -
 free: 3328.89 Mb (81%) | 'Memory usage'=766.34Mb;0.00;0.00;0.00;4095.23
```

Zur Einbindung eines entsprechenden Kommandos als Service können Sie die folgende Dienstdefinition verwenden, hier mit Angabe von Schwellenwerten bei 80% für WARNING und 90% für CRITICAL:

```
define service{
       service_description                    Total Memory Usage
       use                                    template-srv-win
       check_command                          check_nt!MEMUSE!-w 80 -c 90
       hostgroup_name                         hg-win
}
```

Das Plugin gibt dabei Performance-Daten zurück, so dass Sie bei Nutzung von PNP4Nagios automatisch einen entsprechenden Graphen wie in Abbildung 8-7 erhalten.

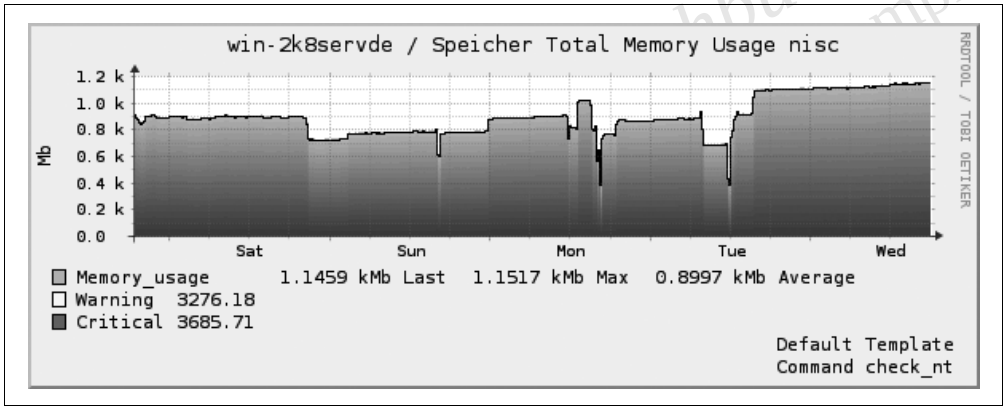


Abbildung 8-7: PNP4Nagios-Graph von über check_nt ermittelter RAM-Auslastung

Für NRPE in NSClient++ ist hierzu ein bereits vordefiniertes Alias-Kommando vorhanden. Dabei bezieht sich der Parameter type auf die unterschiedlichen Speichertypen – physical bezieht sich auf den genutzten physikalischen Hauptspeicher und virtuell auf den virtuellen Speicher in der Auslagerungsdatei. Die Parameter page beziehungsweise paged beziehen sich auf den gesamten Speicher unter Nutzung unterschiedlicher Routinen. Wenn Sie sich ein eigenes Alias-Kommando zusammensetzen, ist es ausreichend, mit paged nur die aktuellere Routine zu verwenden, anderenfalls erhalten Sie die letztgenannte Ausgabe zweimal:

```
[/settings/external scripts/alias]
alias_mem = checkMem MaxWarn=80% MaxCrit=90% ShowAll=long type=physical type=virtual
type=paged type=page
```

Für die Einbindung in Nagios/Icinga können Sie dann folgende Dienstdefinition verwenden:

```
define service{
       service_description                    Speicher Total Memory Usage nrpe
       use                                    template-srv-win
       check_command                          check_nrpe!alias_mem
       hostgroup_name                         hg-win
}
```

Datenträgerauslastung überwachen

Ein weiteres Kernmerkmal für alle Maschinen ist die Datenträgerauslastung beziehungsweise der Füllstand von logischen Laufwerken. Dies gilt auch dann, wenn das System nicht primär zur Speicherung von Daten verwendet wird. Die entsprechenden Informationen lassen sich am einfachsten via SMNP abrufen. Dazu können Sie das Manubulon-Plugin `check_snmp_storage.pl` verwenden. Im folgenden Beispiel geben wir mit der Option `-m` den Namen des zu prüfenden Laufwerkes an. Der Parameter `-f` sorgt dafür, dass Performancedaten ausgegeben werden:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_storage.pl -C "COMMUNITY"
-H 10.85.58.111 -m ^[C]: -w 80 -c 90 -f
C:\ Label:  Serial Number 681bd690: 57%used(23118MB/40859MB) (<80%) : OK | 'C:\_Label:__S
erial_Number_681bd690'=23118MB;32687;36773;0;40859
```

Als entsprechende Dienstdefinition für die Überwachung von Laufwerk C: können Sie dann beispielsweise die folgende verwenden:

```
define service{
       service_description                    check_fs-root
       use                                    template-srv-win
       check_command                          check_snmp_storage!'^[C]:'!80!90
       hostgroup_name                         hg-win
}
```

Sie können die gleiche Abfrage wie folgt alternativ mit `check_nt` durchführen:

```
root@moni:~# /usr/local/icinga/libexec/check_nt -H 10.85.58.111 -p 12489 -s PASSWORT
-v USEDDISKSPACE -l c
c:\ - total: 39,90 Gb - used: 22,58 Gb (57%) -
 free 17,32 Gb (43%) | 'c:\ Used Space'=22,58Gb;0,00;0,00;0.00;39,90
```

Eine entsprechende Dienstdefinition für die Einbindung auf Ihrem Monitoring-System ist:

```
define service{
       service_description                    Drive Space C:\ nisc
       use                                    template-srv-win
       check_command                          check_nt!USEDDISKSPACE!-l c -w 80 -c 90
       hostgroup_name                         hg-win
}
```

Um mit NRPE den Füllstand von Laufwerk `C:` zu überwachen, können Sie in Ihrer NSClient++-Konfigurationsdatei `nsclient.ini` folgendes Alias-Kommando eintragen:

```
[/settings/external scripts/alias]
alias_diskc = CheckDriveSize MinWarn=20% MinCrit=10% ShowAll Drive=c:\
```

Bei einem Test sähe die Ausgabe folgendermaßen aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nrpe -H 10.85.58.113 -p 5666
-c alias_diskc
OK: c:\: 16.3G|'c:\ %'=19%;10;5 'c:\'=16.29G;1.89999;0.98999;0;19
```

Die entsprechende Dienstdefinition für Nagios/Icinga ist dann:

```
define service{
       service_description                    Drive Space C:\ nrpe
       use                                    template-srv-win
       check_command                          check_nrpe!alias_diskc
       hostgroup_name                         hg-win
}
```

Um mehrere Festplatten zu überwachen, müssten Sie bei NSClient den Test für jedes Laufwerk einzeln konfigurieren. Über NRPE können Sie mit folgendem, bereits bei der Installation von NSClient++ vordefinierten Alias-Kommando mehrere Laufwerke gleichzeitig mit einem Test prüfen:

```
[/settings/external scripts/alias]
alias_disk = CheckDriveSize MinWarn=10% MinCrit=5% CheckAll FilterType=FIXED
```

Wenn Sie PNP4Nagios verwenden, erhalten Sie dann bei allen drei Varianten automatisch auch einen entsprechenden Graphen, wie beispielsweise den in Abbildung 8-8 gezeigten.

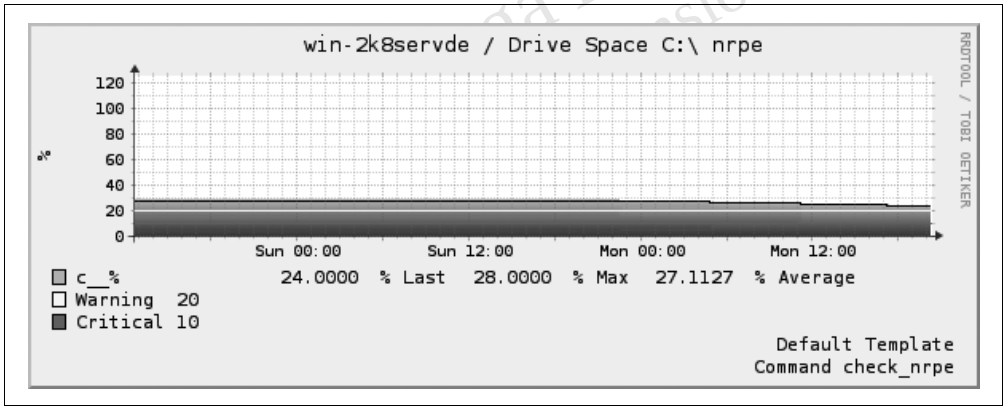


Abbildung 8-8: PNP4Nagios-Graph von über check_nt ermittelter Plattenauslastung

Grundlegende Windows-Prozesse überwachen

Bei den Prozessnamen handelt es sich meistens um ausführbare Dateien vom Typ `.exe`. Die Namen der Prozesse ermitteln Sie am besten aus dem Windows-Taskmanager. Dort gibt es eine Registerkarte `Prozesse`, in der unter anderem der Prozessname (Abbildname) und eine kurze Beschreibung zu finden sind.

In der folgenden Tabelle 8-1 finden Sie einige grundlegenden Prozesse für verschiedene Windows-Varianten aufgelistet, die auf jedem System laufen sollten. Darauf basierend können Sie sich eine individuell angepasste Liste zusammenstellen.

Tabelle 8-1: Grundlegende Prozesse verschiedener Windows-Varianten

Windows-Varianten	Grundlegende Prozesse
Windows XP	svchost.exe,services.exe,lsass.exe
Windows 7 / Server 2008/2012	svchost.exe,spoolsv.exe,lsass.exe

Die Prozessüberwachung über die Prozessnamen wäre dann beispielsweise:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nt -H 10.85.58.113 -p 12489 -s PASSWORT
-v PROCSTATE -l svchost.exe,spoolsv.exe,lsass.exe,nscp.exe,winlogon.exe,snmp.exe
 OK: All processes are running.
```

Sie können sich mit der Option `-d SHOWALL` auch eine Liste mit den Zuständen aller Prozesse (laufend und nicht laufend) anzeigen lassen. Im folgenden Beispiel haben wir exemplarisch einen nicht laufenden Prozess `meinprog.exe` hinzugefügt:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_nt -H 10.85.58.113 -p 12489 -s PASSWORT
-v PROCSTATE -d SHOWALL -
l svchost.exe,spoolsv.exe,lsass.exe,nscp.exe,winlogon.exe,snmp.exe,meinprog.exe
 svchost.exe: Running - spoolsv.exe: Running - lsass.exe: Running - nscp.exe: Running -
 winlogon.exe: Running - snmp.exe: Running - meinprog.exe: not running
```

Eine entsprechende Dienstdefinition für Nagios/Icinga zum Testen einiger grundlegender Services sieht dann wie folgt aus:

```
define service{
       service_description                    Win2k8 Essential Processes nisc
       use                                    template-srv-win
       check_command                          check_nt!PROCSTATE!-l svchost.exe, \
                                              spoolsv.exe,lsass.exe
       host_name                              win-2k8servde
}
```

Mit dem folgendem Alias-Kommando lässt sich unter NRPE ein Dienst (hier `svchost.exe`) testen. Der Test liefert auch Performancedaten zurück, aus denen hervorgeht, wie oft dieser Dienst läuft. Damit können Sie auch Zähler für `WARNING` und `CRITICAL` angeben:

```
alias_procstate_svchost_2k8 = checkProcState "svchost.exe=started" MinCritCount=10
MaxCritWarn=12
```

Sie können auch, wie bei NSClient, mehrere Dienste mit NRPE auf einmal testen. Dazu empfehlen wir Ihnen die Ausgabe von Performancedaten zu unterdrücken:

```
alias_procstate_2k8 = checkProcState ignore-perf-data
"svchost.exe=started" "spoolsv.exe=started" "lsass.exe="started"
```

Um einen Alias zu testen führen Sie dann das Plugin auf der Monitoring-Maschine entsprechend aus:

```
root@moni:~# /usr/lib/nagios/plugins/check_nrpe -H 10.85.58.113 -p 5666
-c alias_procstate_2k8
OK: All processes are running.
```

Eine entsprechende Dienstdefinition für Nagios/Icinga sieht dann wie folgt aus:

```
define service{
       service_description                    Win2k8 Essential Processes nrpe
       use                                    template-srv-win
       check_command                          check_nrpe!alias_procstate_2k8
       host_name                              win-2k8servde
}
```

Im Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* finden Sie eine detaillierte Beschreibung der Konfiguration von NSClient++.

Grundlegende Windows-Dienste überwachen

Die Namen der Dienste ermitteln Sie am besten aus dem Windows-Taskmanager. Dort gibt es eine Registerkarte `Dienste`, in der für jeden Dienst der Dienstname, eine kurze Beschreibung und dessen Status zu finden sind. In der Tabelle 8-2 habe wir Ihnen die grundlegenden Dienstnamen der Dienste aufgelistet, die nach einer Standardinstallation auf jeden Fall laufen sollten. Sie können ausgehend von dieser Liste einfach zu überwachende Dienste hinzufügen oder entfernen:

Tabelle 8-2: Grundlegende Dienste verschiedener Windows-Varianten

Windows-Varianten	Grundlegende Dienste
Windows XP	DcomLaunch, Eventlog, PlugPlay, RpcSs, SamSs, AudioSrv, winmgmt, lanmanworkstation
Windows 7	EventSystem, CryptSvc, DcomLaunch, Dhcp, gpsvc, NlaSvc, nsi, PlugPlay, Power, Spooler, RpcSs, RpcEptMapper, SamSs, LanmanServer, SysMain, SENS, Schedule, ProfSvc, AudioSrv, AudioEndpointBuilder, EventLog, FontCache, Winmgmt, LanmanWorkstation
Windows Server 2008	BFE, EventSystem, CryptSvc, DcomLaunch, UxSms, Dhcp, TrkWks, Dnscache, gpsvc, NlaSvc, nsi, PlugPlay, Power, Spooler, RpcSs, RpcEptMapper, SamSs, LanmanServer, SENS, Schedule, lmhosts, ProfSvc, EventLog, MpsSvc, Winmgmt, LanmanWorkstation
Windows Server 2012	BFE, EventSystem, CryptSvc, DcomLaunch, Dhcp, TrkWks, Dnscache, gpsvc, NlaSvc, nsi, PlugPlay, Power, Spooler, RpcSs, RpcEptMapper, SamSs, LanmanServer, SENS, Schedule, lmhosts, ProfSvc, EventLog, MpsSvc, Winmgmt, LanmanWorkstation

Mittels der folgenden Prüfung über `check_nt` können Sie beispielsweise testen, ob der Dienst Dhcp(-Client) auf dem Zielrechner läuft:

```
root@moni:/usr/local/icinga/libexec# ./check_nt -H 10.85.58.110 -p 12489 -s PASSWORT
-v SERVICESTATE -d SHOWALL -l Dhcp
 Dhcp: Started
```

Ein Dienstdefinition, um mit `check_nt` die Basisdienste auf einem Windows Server 2008 zu überwachen, sieht dann beispielsweise wie folgt aus:

```
define service{
       service_description                     Win2k8 Essential Services
       use                                     template-srv-win
       check_command                           check_nt!SERVICESTATE!-l BFE,EventSystem,\
CryptSvc,DcomLaunch,UxSms,Dhcp,TrkWks,Dnscache,gpsvc,NlaSvc,nsi,PlugPlay,Power,Spooler,\
RpcSs,RpcEptMapper,SamSs,LanmanServer,SENS,Schedule,lmhosts,ProfSvc,EventLog,MpsSvc,\
Winmgmt,LanmanWorkstation
       host_name                               win-2k8servde
}
```

Sie können den soeben für NSClient vorgestellten Test auch mit NRPE durchführen und eine Liste von zu prüfenden Diensten angeben. Dies ist sinnvoll, wenn Sie nur eine Teilmenge von Diensten testen möchten. Zusätzlich können Sie auch prüfen, ob ein bestimmter Dienst, hier `RemoteAccess`, nicht läuft. Das entsprechende Alias-Kommando für Windows Server 2008 wäre dann beispielsweise das im Folgenden angeführte. Achten Sie darauf, dass hier die Aufzählung ohne Trennung durch Kommata erfolgt:

```
;Services
alias_servicestate_2k8 = checkServiceState CheckAll BFE EventSystem CryptSvc DcomLaunch
UxSms Dhcp TrkWks Dnscache gpsvc NlaSvc nsi PlugPlay Power Spooler RpcSs RpcEptMapper
SamSs LanmanServer SENS Schedule lmhosts ProfSvc EventLog MpsSvc Winmgmt LanmanWorkstation
RemoteAccess=stopped
```

Alternativ können Sie auch prüfen, ob alle Dienste, die im Windows-System als Starttyp `Automatisch` gesetzt sind, auch tatsächlich laufen, und entsprechend alle, die als Starttyp `Deaktiviert` gesetzt sind, nicht laufen. Zu diesem Zweck steht bei NSClient++ für NRPE das folgende, vordefinierte Alias-Kommando zur Verfügung:

```
alias_service = checkServiceState CheckAll
```

Wie Sie sehen, wird dadurch die Konfiguration wesentlich einfacher. Testen können Sie diesen Alias wie folgt:

```
root@moni:~# /usr/lib/nagios/plugins/check_nrpe -H 10.85.58.113 -p 5666 -c alias_service
OK: All services are in their appropriate state.
```

Wenn Sie beispielsweise einige automatisch gestartete Dienste von diesem Test ausschließen möchten, fügen Sie den Parameter `exclude` mit dem Dienstnamen des auszuschließenden Services hinzu:

```
alias_service_excl = checkServiceState CheckAll exclude=RemoteAccess exclude=SNMPTRAP
```

Update-Status überwachen

NSClient++ enthält unter anderem das Script `check_updates.vbs`, mit dem sich über NRPE der Update-Status einer Windows-Maschine überwachen lässt. Dieses binden Sie wie folgt in die NSClient-Konfigurationsdatei `nsclient.ini` ein. Die dafür notwendigen Abschnitte sind im Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten* erläutert:

```
[/settings/external scripts/wrappings]
bat = scripts\\%SCRIPT% %ARGS%
ps1 = cmd /c echo scripts\\%SCRIPT% %ARGS%; exit($lastexitcode) |
powershell.exe -command -vbs = cscript.exe //T:30 //NoLogo scripts\\lib\\wrapper.vbs
%SCRIPT% %ARGS%
[/settings/external scripts/wrapped scripts]
check_updates = scripts\\check_updates.vbs
[/settings/external scripts/alias]
alias_updates = check_updates -warning 0 -critical 0
```

Die entsprechend erforderliche Dienstdefinition ist dann:

```
define service{
       service_description                    Windows Update Status
       use                                    template-srv-win
       check_command                          check_nrpe!alias_updates
       hostgroup_name                         hg-win
}
```

Diskussion

Für das Monitoring von lokalen Ressourcen von Windows-Systemen kann ein Großteil der Nagiosplugins nicht genutzt werden, und SNMP lässt sich auch nur eingeschränkt einsetzen. Als Alternative haben wir Ihnen gezeigt, wie Sie die meisten der unter Linux möglichen Abfragen mit Hilfe des Monitoring-Agenten NSClient++ umsetzen können. Seitens Ihres Nagios/Icinga-Servers benötigen Sie dann lediglich die mit den Nagiosplugins mitgelieferten Plugins `check_nt` und `check_nrpe`.

Wir empfehlen Ihnen, die zu überwachenden Systeme weitestgehend in Gruppen zusammenzufassen und Vorgabe-Konfigurationen zu erstellen. Gehen Sie dazu wie folgt vor:

1. Definieren Sie eine Maschinengruppe (siehe Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*), beispielsweise `hg-win`, als Hauptgruppe für alle Windows-Systeme. Wenn Ihre Prüfungen nicht auf allen Windows-Maschinen laufen, dann müssen Sie eventuell weitere Gruppen definieren, um diese abzubilden.
2. Definieren Sie sich nun die zu überwachenden Dienste (siehe Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*) und referenzieren Sie jeweils die passende Maschinengruppe.
3. Installieren Sie den NSClient++-Agenten (siehe Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*) auf den zu überwachenden Systemen beziehungsweise installieren und konfigurieren Sie den SNMP-Agenten (Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*).
4. Ordnen Sie die Windows-Maschinen bei deren Einbindung (siehe Rezept 4.1, *Maschinen einbinden (host)*) der passenen Maschinengruppe zu. Es werden dann automatisch alle die Gruppe referenzierenden Dienste für diese Maschinen durchgeführt.

Legen Sie dabei einen Standardumfang von Diensten fest, die Sie auf allen Geräten prüfen wollen. Unter Umständen werden Sie dabei feststellen, dass einige Windows-Varianten spezifische Anpassungen erfordern. Erwägen Sie in diesem Fall für unterschiedliche Windows-Varianten eigene Gruppen beziehungsweise Untergruppen zu definieren.

Bei der Nutzung von NSClient beziehungsweise des Plugins `check_nt` sollten Sie beachten, dass die Kommunkation unverschlüsselt stattfindet und das Passwort im Klartext übertragen wird. Bei NRPE ist die Situation etwas besser, da hier der Kommunikationskanal SSL-verschlüsselt wird. Allerdings ist das Zertifikat fest einkompiliert und lässt sich nicht ohne Weiteres austauschen. Hierfür müssten Sie die bei NSClient++ Version ≧ 0.4 mitgelieferte Variante von `check_nrpe` auf dem Monitoring-Server einsetzen.

Siehe auch

- Rezept zur Einrichtung von SNMP auf Windows-Maschinen: Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*
- Rezept zur Konfiguration von NRPE und NSClient auf Windows-Maschinen: Rezept 2.8, *NRPE und NSClient auf Windows-Maschinen vorbereiten*
- Webseiten des Projekts NSClient++: *http://www.nsclient.org/*
- NSClient++ auf Sourceforge: *http://sourceforge.net/projects/nscplus/files/nscplus/*
- Rezept zur Remote-Ausführung von lokalen Plugins über SSH: Rezept 5.10, *Entfernte Ausführung über SSH (check_by_ssh)*
- Rezept zur Remote-Ausführung von lokalen Plugins über NSClient: Rezept 5.9, *Überwachung einer Windows-Maschine mit NSClient (check_nt)*
- Rezept zur Remote-Ausführung von lokalen Plugins über NRPE: Rezept 5.8, *Überwachung einer Maschine mit NRPE (check_nrpe)*
- Rezept zur Vereinfachung der Konfiguration mit Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Rezept zur Einbindung einer Maschine: Rezept 4.1, *Maschinen einbinden (host)*
- Rezept zur Zusammenfassung von Maschinen in Maschinengruppen: Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*
- Rezepte zur Überwachung der Erreichbarkeit: Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)* und Rezept 6.2, *Erreichbarkeit überwachen mit ICMP (check_icmp)*
- Rezept zur Statusprüfung einzelner Netzwerkschnittstellen: Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*
- Rezept zur Statusprüfung von Netzwerkschnittstellen: Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)*
- Rezept zur Erstellung generischer SNMP-Abfragen: Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*

- Rezept zur Erstellung eines Wrapper-Scripts: Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*
- Rezept zur Einbindung der Manubulon-Plugins: Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*

8.3 Überwachung von Hypervisoren und virtuellen Maschinen

Problem

Sie betreiben auf einer Maschine, dem sogenannten Hypervisor (etwa unter Xen oder libvirt), virtuelle Maschinen und möchten diese überwachen.

Lösung

Die Überwachung eines Hypervisors und entsprechender virtueller Maschinen setzt sich aus mehreren Teilen zusammen. Wir erläutern Ihnen im Folgenden zunächst, wie Sie die Überwachung generell aufbauen. Danach beschreiben wir Ihnen mögliche spezielle Erweiterungen für den Hypervisor selbst am Beispiel einer KVM-Installation.

Genereller Aufbau

Den Hypervisor, also den Rechner, auf dem die virtuellen Maschinen laufen, können Sie zunächst grundlegend einbinden wie jede andere Maschine auch (siehe Rezept 8.1, *Überwachung von Unix-/Linux-Servern* beziehungsweise Rezept 8.2, *Überwachung von Windows-Maschinen*). Im Sinne des Monitorings können und sollten Sie virtuelle Maschinen dann zunächst ebenfalls wie physikalisch vorhandene Rechner betrachten. Entsprechend binden Sie auch jede zu überwachende virtuelle Maschine ein und tragen dabei den Hypervisor als Elternbeziehung (`parent`) ein (siehe hierzu Rezept 4.1, *Maschinen einbinden (host)*).

Die virtuellen Maschinen werden automatisch nur dann getestet, wenn der Hypervisor als laufend eingestuft ist. Zumindest auf dem Hypervisor sind die allgemeinen Merkmale aus den Rezepten bereits sehr wichtig, damit Sie mögliche Engpässe frühzeitig erkennen können. Wenn Sie diese Daten auch für die virtuellen Maschinen erheben, können Sie bei entstandenen Engpässen leichter feststellen, ob diese etwa von einer einzelnen virtuellen Maschine verursacht werden. Mit diesen Informationen lassen sich Probleme oft erheblich schneller beseitigen.

Typischerweise wird ein Hypervisor einen eigenen Service für die Verwaltung der virtuellen Maschine aus der Ferne anbieten. Wenn Sie etwa über VNC auf die virtuellen Maschinen zugreifen, können Sie den entsprechenden Port mit Hilfe der Plugins `check_udp` beziehungsweise `check_tcp` überwachen (siehe Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*).

Verschlüsseltes VNC: Sofern Ihr Hypervisor aus einem öffentlichen Netz erreichbar ist, sollten Sie dafür sorgen, dass die entsprechenden Verbindungen verschlüsselt sind und dann die Ports entsprechend anpassen. Sie können dann zusätzlich testen, dass die unverschlüsselten Dienste nicht erreichbar sind, um mögliche Konfigurationsfehler zu erkennen.

Eine häufige Option, um entsprechende Verbindungen verschlüsselt abzuwickeln, ist auch die Nutzung von SSH. Auf diese Weise unterstützt die Anwendung zum Management von KVM (`libvirt-manager`) das Tunneln der eigentlich unverschlüsselten Verbindungen über SSH. Wenn Sie diesen Mechanismus nutzen, können Sie mittels `check_ssh` (siehe Rezept 6.12, *Überwachung eines SSH-Dienstes (check_ssh)*) die Erreichbarkeit des SSH-Dienstes auf dem Hypervisor überprüfen.

Überwachen Sie auf den virtuellen Maschinen dann weiter die Dienste, für die Sie diese Maschine betreiben (etwa Webserver, Verzeichnis-Freigaben, SSH-Server). Nehmen Sie hierzu gegebenenfalls jeweils die Rezepte (siehe Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*) für die Überwachung des entsprechenden Dienstes zu Hilfe. Auf diese Weise erhalten Sie die wichtige Information dazu, ob die gewünschten Dienste tatsächlich erreichbar sind.

Hypervisor überwachen

Sie sollten dann weiter gezielt den Prozess des Hypervisors überwachen. Wir zeigen Ihnen dies am Beispiel von KVM auf. Der entsprechende Prozess heißt `kvm` und lässt sich auf der Kommandozeile wie folgt abfragen:

```
root@hcksp0:~# ps -C kvm
 PID TTY TIME CMD
11026 ? 3-10:45:14 kvm
11420 ? 3-06:04:52 kvm
11439 ? 3-00:38:06 kvm
11499 ? 7-07:03:01 kvm
12072 ? 3-12:47:16 kvm
12291 ? 2-23:35:30 kvm
```

Die Anzahl der Prozesse unter diesem Namen gibt dabei gleichzeitig die Anzahl der aktuell laufenden virtuellen Maschinen an. Um diese Information nun vom Monitoring-Server aus abzufragen, bietet sich eine entsprechende Option des SNMP-Servers an. Dies setzt allerdings voraus, dass Sie auf der Maschine SNMP installieren und konfigurieren (siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* beziehungsweise Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*). Wir gehen im folgenden Beispiel von einem Linux-System mit installiertem SNMP aus. Fügen Sie Ihrer SNMP-Konfigurationsdatei auf dem Hypervisor dann die folgende Option hinzu, um diese KVM-Prozesse überwachen zu lassen:

```
[...]
# Add KVM-Hypervisor to UCD-SNMP-MIB::prTable
proc kvm
[...]
```

Starten Sie danach den SNMP-Server neu, damit diese Änderung wirksam wird:

```
root@hckspO:~# service snmpd restart
```

Damit wird der SNMP-Dienst Informationen zu den Prozessen über die MIB `UCD-SNMP-MIB` zur Verfügung stellen. Die entsprechende Tabelle `prTable` können Sie über die Kommandozeile etwa wie folgt abfragen (beachten Sie bitte, dass Sie die Zugangsdaten entsprechend Ihrer Umgebung anpassen müssen):

```
root@moni:~# snmpwalk -u icinga -l authPriv -a SHA -A "PASSPHRASE"- x AES
-X "PASSPHRASE" hckspO .1.3.6.1.4.1.2021.2
UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prNames.1 = STRING: kvm
UCD-SNMP-MIB::prMin.1 = INTEGER: 0
UCD-SNMP-MIB::prMax.1 = INTEGER: 0
UCD-SNMP-MIB::prCount.1 = INTEGER: 6
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: noError(0)
UCD-SNMP-MIB::prErrMessage.1 = STRING:
UCD-SNMP-MIB::prErrFix.1 = INTEGER: noError(0)
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
```

Im vorliegenden Beispiel wird nur dieser eine Prozess überwacht. Sofern Sie für SNMP mehrere zu überwachende Prozesse eingerichtet haben, fällt die Ausgabe gegebenenfalls entsprechend länger aus (ein entsprechender Abschnitt für jeden überwachten Prozess). Als interessanten Wert können Sie hier nun zum Beispiel die Anzahl laufender Prozesse mit diesem Namen über das Monitoring abfragen. Nutzen Sie hierzu das in den Standardplugins enthaltene `check_snmp` (siehe Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*), um `UCD-SNMP-MIB::prCount` (nummerische OID ist hier `.1.3.6.1.4.1.2021.2.1.5.1`) auszuwerten. Auf der Kommandozeile ergibt ein entsprechender Aufruf des Plugins zum Beispiel die folgende Rückgabe (achten Sie wieder darauf, die Verbindungsparameter anzupassen):

```
icinga@moni:~# /usr/local/icinga/libexec/check_snmp -H 10.85.58.1 -P 3 -L authPriv
-U "icinga" -a SHA -A "PASSPHRASE" -x SHA -X "PASSPHRASE" -o .1.3.6.1.4.1.2021.2.1.5.1
SNMP OK - 6 | iso.3.6.1.4.1.2021.2.1.5.1=6
```

An dem Pipe-Symbol »|« können Sie erkennen, dass das Plugin Performance-Daten zurückgibt. Entsprechend können Sie sich die Werte dieser Prüfung dann auch in einem Graphen darstellen lassen. Der in Abbildung 8-9 gezeigte Graph basiert auf einem Abfrageintervall von 1 Minute. Wenn Sie größere Intervalle für die Abfrage verwenden, werden die Graphen bei Ihnen gegebenenfalls gröber aufgelöst.

Diskussion

Wie gezeigt können Sie virtuelle Maschinen weitestgehend wie echte Maschinen behandeln. Ausnahmen hierzu sind beispielsweise Temperaturmessungen oder Messungen einer Lüftergeschwindigkeit – bei virtuellen Maschinen gibt es diese schlicht nicht. Als Besonderheit basieren virtuelle Maschinen aber eben auf demselben Hypervisor, weshalb bestimmte weitergehende Betrachtungen interessant sein können. Dies möchten wir Ihnen am Beispiel der Auslastung der Dateisysteme aufzeigen.

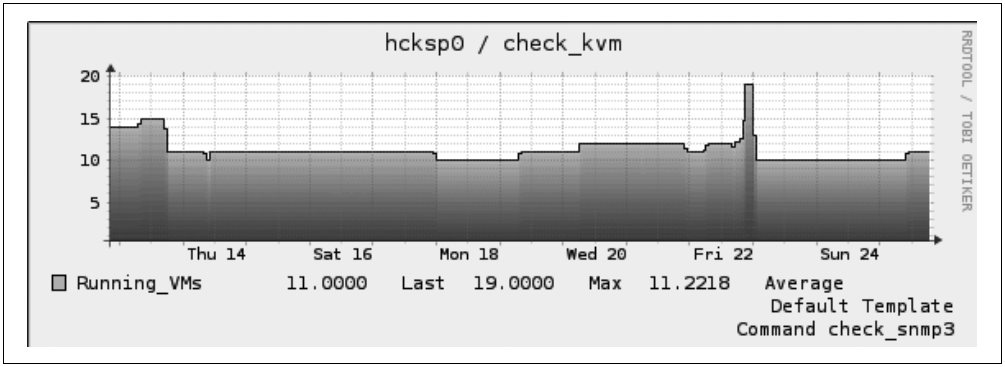


Abbildung 8-9: PNP4Nagios-Graph für die Anzahl der laufenden VMs auf einem KVM-Hypvervisor

Die Festplatten der virtuellen Maschinen sind auf dem Hypervisor meist als Dateien enthalten, wobei die Größe der Festplatte die Größe dieser Datei bestimmt. Auf dem Hypervisor sieht es deshalb in der Regel so aus, als wäre der gesamte Speicherplatz belegt. Entsprechend sind Änderungen im Füllstand nicht zu erkennen, wenn Sie die Dateisysteme lediglich auf dem Hypervisor überwachen. Abbildung 8-10 zeigt Ihnen den Graphen einer entsprechenden Überwachung auf einem Hypervisor. Beachten Sie, dass hier kaum eine Veränderung zu erkennen ist.

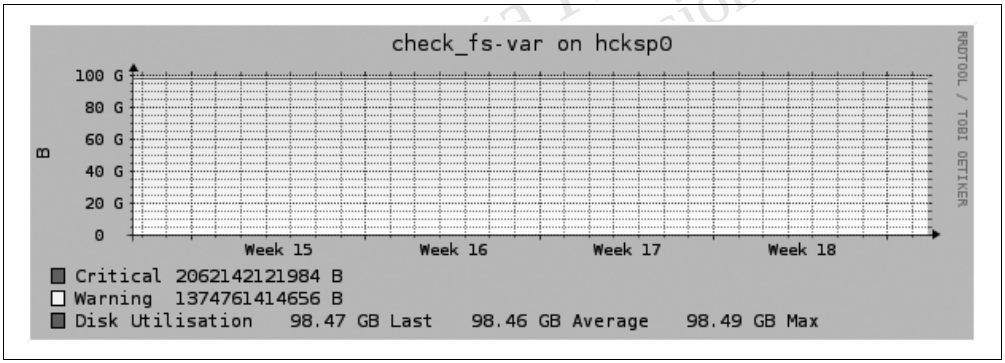


Abbildung 8-10: Graph des Dateisystems auf einem Hypervisor

Wenn Sie dazu Tests auf allen virtuellen Maschinen durchführen, haben Sie entsprechend viele unterschiedliche Werte, die zu vergleichen zunächst recht mühselig ist. Sie können sich hier jedoch leicht einen Überblick verschaffen, indem Sie die entsprechenden Daten zusammenfassen. Innerhalb von Nagios/Icinga erreichen Sie dies über Host- (siehe Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)*) und/oder Servicegruppen (siehe Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*). Wenn Sie auf Gruppen zugreifen, werden Ihnen nur noch die entsprechenden Werte angezeigt (siehe hierzu auch Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*). Allerdings werden diese Werte dabei immer noch einzeln dargestellt, was einen Vergleich mühselig macht.

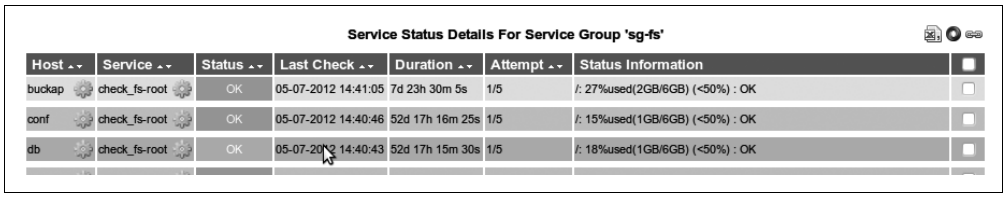


Abbildung 8-11: Dateisysteme virtueller Maschinen zusammengefasst als Servicegruppe

Sofern Sie sich mit PNP4Nagios Graphen zu den Werten zeichnen lassen, können die entsprechenden Werte jedoch auch in einem einzelnen Graphen zusammenfassen (siehe Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*). Abbildung 8-12 zeigt Ihnen den Multi-Graphen der virtuellen Maschinen für den gleichen Zeitraum wie der weiter vorne angeführte Graph für den dazugehörigen Supervisor.

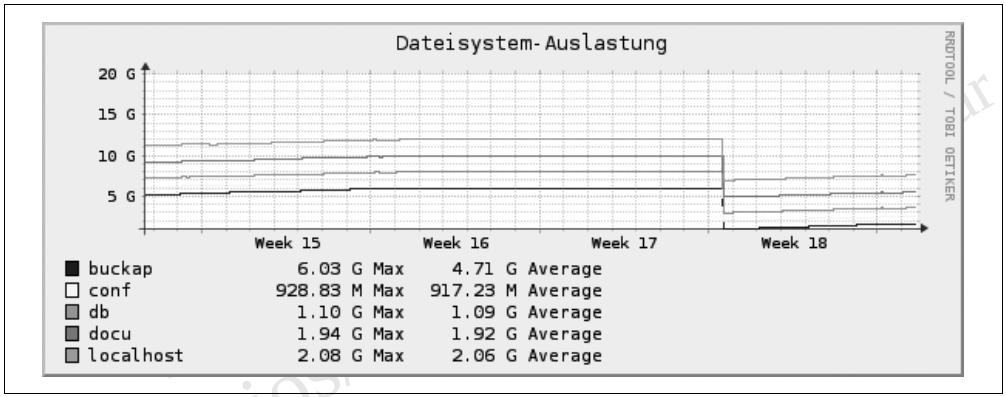


Abbildung 8-12: PNP4Nagios: Multigraph von Dateisystemen

Vergleichen Sie, welche Dynamik der Multigraph gegenüber dem voranstehenden einfachen Graphen aufzeigt. Dies liegt im vorliegenden Fall daran, dass die Abbild-Dateien für die virtuellen Maschinen immer die maximal zugewiesene Größe haben und auf dem Hypervisor entsprechend immer gleich viel Platz benötigen. Entsprechend zeigen Ihnen hier nur die Messungen auf den virtuellen Maschinen selber den tatsächlichen Zustand.

Abhängig davon, wie Sie die virtuellen Maschinen nutzen, können Sie sich nach Bedarf passende Auswertungen erstellen. Mit diesen können Sie sich dann immer schnell einen Überblick über die für Sie wichtigen Parameter verschaffen und Soll-Abweichungen gegebenenfalls schneller erkennen und beheben.

Siehe auch

- Grundlegendes Monitoring von Maschinen unter Linux beziehungsweise Windows: Rezept 8.1, *Überwachung von Unix-/Linux-Servern* und Rezept 8.2, *Überwachung von Windows-Maschinen*
- Rezept zur Überwachung beliebiger TCP/UDP-Ports: Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*

- Rezept zur Überwachung eines SSH-Servers: Rezept 6.12, *Überwachung eines SSH-Dienstes (check_ssh)*
- Kapitel zur Überwachung von Netzwerkdiensten: Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezepte zur Einrichtung von SNMP auf Linux und Windows: Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten* und Rezept 2.6, *SNMP auf Windows-Maschinen vorbereiten*
- Rezept zur generischen Abfrage von SNMP-Werten: Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*
- Rezept zur Einbindung einer Maschine: Rezept 4.1, *Maschinen einbinden (host)*
- Rezepte zu Maschinen- und Dienstgruppen: Rezept 4.6, *Maschinen zu Gruppen zusammenfassen (host-group)* und Rezept 4.7, *Services zu Gruppen zusammenfassen (service-group)*
- Rezepte zu den Oberflächen Classic und Icinga-Web: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*
- Rezept zur Zusammenfassung von Graphen über Special Templates: Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*

8.4 Überwachung der aktiven Infrastruktur

Problem

Neben Servern bestehen Netzwerke typischerweise aus weiteren aktiven Komponenten, wie etwa Switches, Router/Firewall, PoE-Injektor, SAN-Switches, unterbrechungsfreie Stromversorgung und so weiter. Diese bilden die sogenannte aktive Infrastruktur. Sie möchten diese in die Überwachung einbinden.

Lösung

Die aktiven Komponenten einer Netzwerk-Infrastruktur sind aus der Sicht des Netzwerks typischerweise wichtiger, als die Server, die sie verwenden, selbst. Aufgrund der großen Anzahl unterschiedlicher Hersteller gibt es hier allerdings kein in allen Fällen funktionierendes Standard-Verfahren für die Einbindung.

Zwar können die Geräte typischerweise über das SNMP (Simple Network Management Protocol) kommunizieren, so dass Sie das in den Standard-Plugins enthaltene `check_snmp` (siehe Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*) einsetzen können. Allerdings haben viele Hersteller in diesem Bereich die Erweiterungsmöglichkeiten des SNMP genutzt und bieten Daten basierend auf eigenen MIBs an (MIB = Management Information Base, siehe hierzu Rezept 2.4, *SNMP mit MIBs verwenden*). Dadurch werden

Ihnen dann beispielsweise statt des standardmäßig vorgesehenen einzelnen Netzteils die Daten für alle vorhandenen Netzteile zur Verfügung gestellt. Entsprechend besteht hier jedoch häufig der Bedarf, die MIBs des Herstellers einzubinden und/oder spezifische Plugins für den jeweiligen Hersteller einzusetzen.

Wir zeigen Ihnen im Folgenden auf, was Sie bei Geräten der aktiven Infrastruktur überwachen können und sollten. Um dabei auf die breite Palette an Möglichkeiten eingehen zu können, verzichten wir auf die Betrachtung von herstellerspezifischen Aspekten und geben Ihnen hier lediglich Hinweise auf entsprechende Durchführungsmöglichkeiten.

Generelle Überwachungen

Bevor wir gezielt auf einzelne Gerätetypen eingehen, möchten wir Ihnen zunächst übergreifend gültige Aspekte nennen. Bei allen Geräten können Sie typischerweise die folgenden Daten auslesen und -werten:

- Erreichbarkeit

  Die Erreichbarkeit über das Netzwerk testen Sie typischerweise wie bei Servern auch über die zu den Maschinen-Definitionen hinterlegten Host-Checks (siehe hierzu Rezept 4.1, *Maschinen einbinden (host)*).

- Stromversorgung

  Herkömmlich haben fast alle Netzwerkgeräte ein oder mehrere Netzteile. Kleinere Geräte, wie beispielsweise WLAN Access Points, werden teilweise auch durch PoE (Power-over-Ethernet) mit Strom versorgt. Im Endeffekt sind jedenfalls alle Geräte abhängig von der Stromversorgung und bieten entsprechend häufig betreffende Kennzahlen, wie den generellen Zustand (an oder aus) oder detaillierte Werte wie Spannungen.

- CPU-Last

  Viele Netzwerk-Geräte (beispielsweise vom Typ Router oder Switch) haben auch CPUs. Die Auslastung dieser zu überwachen, ist hier häufig weniger kritisch als bei Servern. Aber auch bei solchen Geräten kann es diesbezüglich zu Problemen mit der Auslastung kommen, weshalb wir eine Überwachung durchaus empfehlen möchten.

- Speichernutzung

  Geräte mit einer CPU benötigen normalerweise auch Speicher. Diesen zu überwachen ist unserer Erfahrung nach häufig sogar wichtiger als die Überwachung der CPU. Da die Geräte typischerweise ununterbrochen laufen und nur in Ausnahmefällen neu gestartet werden, kann es hier besonders leicht zu einem Volllaufen kommen.

- Temperatur

  Sehr viele Netzwerkgeräte verfügen über einen oder mehrere Temperatursensoren, häufig generisch und zum Gehäuse gehörend, aber auch im Zusammenhang mit bestimmten Komponenten wie CPUs, Netzteilen oder Festplatten.

- Lüfter

  Ebenfalls häufig generisch zum Gehäuse gehörend, aber insbesondere bei Vorhandensein von CPUs oder Netzteilen an diese gekoppelt sind Lüfter. Für die Temperaturentwicklung spielen diese eine große Rolle, so dass die Überwachung der Drehzahlen und ihrer Entwicklung von Bedeutung ist.

Dies ist bereits eine stattliche Anzahl an Tests, die Sie auf fast allen Netzwerkgeräten durchführen können und in der Regel auch sollten. Gerade bei Problemen mit Geräten können Sie nach Bedarf weitere Detail-Prüfungen hinzufügen. Dies können beispielsweise folgende sein:

- Seriennummern

  Wenn bei Ihnen beispielsweise durch einen Dienstleister regelmäßig Geräte ausgetauscht werden, kann Ihnen ein automatisiertes Auslesen der Seriennummern unter Umständen die Arbeit erleichtern.

- Versionsstände

  Bei Problemen mit der auf den Geräten verwendeten Software können Sie durch eine Aufnahme in die Prüfungen feststellen, welche Geräte noch einen veralteten Stand aufweisen und aktualisiert werden müssen.

Diese Art der Prüfungen lohnt sich aber in der Regel nicht für die standardmäßige Durchführung. Implementieren Sie diese nur dann, wenn Sie sie tatsächlich benötigen.

Netzwerke überwachen

Zu den Netzwerkgeräten zählen wir hier beispielsweise managed Switches, Router, Firewalls und VPN-Gateways. Netzwerkgeräte, die sich konfigurieren lassen, sogenannte verwaltete Geräte (englisch `managed network devices`), unterstützen meist das Auslesen vielfältiger Informationen über SNMP.

Die primäre Funktion von Switches besteht darin, Netzwerksegmente zu verbinden. Neben den generellen Prüfungen der Funktionalität, wie beispielsweise Erreichbarkeit, Temperatur und Stromversorgung, spielt bei diesen typischerweise insbesondere die Beobachtung des Status einzelner (Uplink-)Ports eine Rolle. Dies hilft Ihnen dabei, Ausfälle frühzeitig zu erkennen. Je nachdem, ob ein Switch Endgeräte, Server oder ganze Netzwerksegmente anbindet, werden Sie dabei unterschiedlich detailliert vorgehen wollen.

Aus der Sicht des Monitoring-Servers sind sich Router und Switches sehr ähnlich. Von den allgemeinen Funktionalitätsprüfungen kommen beim Router die Auslastung von Prozessor, Speicher und Bandbreite als weitere wichtige Informationsquellen hinzu. Bei einem Router sollten Sie erwägen, die Mehrzahl der aktiven Anschlüsse detailliert zu überwachen, also neben dem Status eben auch die Auslastung.

Da Router IP-Pakete zwischen mehreren Netzwerken weiterleiten, verfügen diese naturgemäß auch über mehrere IP-Adressen. Entsprechend wichtig ist neben der prinzipiellen Erreichbarkeit des Gerätes auch, die Erreichbarkeit dieser einzelnen IP-Adressen aus

unterschiedlichen IP-Netzen sicherzustellen, weshalb Sie diese auch separat überwachen sollten. Sie können sich hierfür ein entsprechendes Kommando bauen, in dem Sie die zu prüfende Adresse nicht per Makro aus der Maschinendefinition entnehmen, sondern über die Dienstdefinition manuell festlegen. Für jede Adresse der Maschine definieren Sie dann darauf aufbauend einen entsprechenden Dienst.

Für Netzwerkgeräte sollten Sie den Einsatz des Plugins `check_snmp_environment` (siehe Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*) in Erwägung ziehen. Mit diesem Plugin können Sie bereits eine generelle Funktionprüfung durchführen, indem Sie die Hardwaresensoren aus den Netzwerkgeräten auslesen. Dazu gehören Statusinformationen von Netzteilen, Lüftern sowie in einigen Fällen auch Card- und Modulstatus.

Bei der Abfrage eines Procurve Ethernet Switch von Hewlett-Packard werden Ihnen beispielsweise folgende Informationen bereitgestellt:

```
icinga@moni:~/plugins$ ./check_snmp_environment.pl -H 10.85.57.15 -C COMMUNITY
-T procurve
5 power OK, 1 fan OK, 1 temp OK : OK
```

Bei einem Cisco-Router werden die Statusinformationen zur Stromversorgung und zur Temperatur sowie der Status der installierten Module in zwei unterschiedlichen Abfragen ausgelesen:

```
icinga@moni:~/plugins$ ./check_snmp_environment.pl -H 10.85.57.25 -C COMMUNITY -T cisco
1 Fan OK, 2 ps OK, 5 volt OK, 3 temp OK : OK
icinga@moni:~/plugins$ ./check_snmp_environment.pl -H 10.85.57.25 -C COMMUNITY -T ciscoSW
6 cards OK, : OK
```

Bei einem Brocade Switch ist die Ausgabe noch etwas ausführlicher. Eine Abfrage sieht dann in etwa wie folgt aus:

```
icinga@moni:~/plugins$ ./check_snmp_environment.pl -H 10.85.57.4 -C COMMUNITY -T brocade
ps 1: OK; ps 2: OK; ps 3: OK; ps 4: OK;
Bottom Fan Tray (1): OK; Bottom Fan Tray (2): OK;
Back Fan A (7): OK;
Line module 1, sensor 1 temperature of 49C: OK;
[...]
Slot 1 (RX-BI-4XG 4-port 10GbE Module): Module status: OK; Redundant status: OK (other);
Slot 2 (RX-BI-24F 24-port 1 GbE Module): Module status: OK; Redundant status: OK (other);
Slot 3 (RX-BI-4XG 4-port 10GbE Module): Module status: OK; Redundant status: OK (other);
Slot 4 (RX-BI-24C 24-
port 1 GbE Copper Module): Module status: OK; Redundant status: OK (other);
[...]
all OK
```

Die Auslastung von Prozessor und Speicher können Sie meistens mit den Plugins `check_snmp_load` und `check_snmp_mem` abdecken:

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_load.pl -H 10.85.57.25
-C 'COMMUNITY' -T cisco -w 80,80,80 -c 90,90,90
CPU : 2 1 1 : OK
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_mem.pl -H 10.85.57.25
-C 'COMMUNITY' -w 80 -c 90 -I
Processor:12%,Fast:50% : 12% :  ; OK
```

Die Netzwerkports können Sie zunächst in der Menge mit dem Plugin `check_ifstatus` generell überwachen (siehe hierzu Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)*). Wichtige Schnittstellen, wie etwa die sogenannten Uplinks-Verbindungen, können Sie dann weiter mit `check_ifoperstatus` gezielt abfragen (siehe hierzu Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*). Zumindest für letztere könnten Sie dann auch die Bandbreiten-Nutzung überwachen. Hier steht Ihnen das Plugin `check_snmp_int.pl` zur Verfügung (siehe Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*).

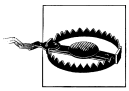
Trunks: Häufig wird die Anbindung an das Backbone über mehrere gekoppelte Leitungen (sogenannte Trunks) und damit Anschlüsse realisiert, um den Anforderungen an die erforderliche Bandbreite gerecht zu werden. In diesen Fällen benötigen Sie zusätzlich zu der Abfrage der Bandbreiten eine zusammenfassende Auswertung. Diese können Sie über entsprechende Graphen mit PNP4Nagios realisieren, in denen Sie die einzelnen Werte zusammenfassen (siehe Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*).

Bei VPN-Gateways, also dedizierten Geräten zum Aufbau und der Verwaltung von VPN-Tunneln, empfehlen wir Ihnen die Überwachung der Anzahl der angemeldeten Benutzer, der ungültigen Anmeldeversuche und der aufgebauten Tunnel. Wenn Ihre Geräte von den vorgestellten Plugins nicht unterstützt werden, kommen Sie nicht umhin, sich entsprechend passende Plugins zu besorgen oder sie selbst umzusetzen (siehe dazu den Diskussionsteil).

Unterbrechungsfreie Stromversorgungen (USVs) überwachen

Unterbrechungsfreie Stromversorgungen (kurz USV) sind in größeren Netzwerken typischerweise Standard für zentrale Geräte der Netzwerkinfrastruktur. Entsprechend können Sie diese ebenfalls in die Überwachung einbeziehen. Besonders wichtig ist es hier, den Status und die Leistungsfähigkeit der verwendeten Akkus zu überwachen. Akkus können während der Lebenszeit eines Gerätes verschleißen und müssen entsprechend erneuert werden. Wir haben bereits erlebt, dass bei Einbindung der Geräte in das Monitoring-System festgestellt wurde, dass ein erheblicher Teil der Geräte gar nicht mehr einsatzfähig war.

Diese Art von Geräten sind nicht primär Netzwerkgeräte. Entsprechend ist bei diesen die Funktionalität zur Überwachung über SNMP häufig nur eine Art Beiwerk und nicht besonders gut ausgestaltet. Oft wird dabei nur SNMP Version 1 oder bestenfalls 2c unterstützt (zu den Versionen und ihren Unterschieden siehe Rezept 2.3, *SNMP auf Linux-Maschinen vorbereiten*). Entsprechend wichtig ist dann eine Anbindung über ein Management-Netzwerk (siehe Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*).



Kritische SNMP-Funktionen: Beachten Sie, dass die Geräte gleichzeitig zur Nutzung der unsicheren Protokolle beispielsweise über SNMP steuerbare Testfunktionen beinhalten können, die etwa die Simulation eines Stromausfalls erlauben. Unberechtigte Zugriffe können hier also durchaus erhebliche Folgen haben, auch wenn das nicht sofort offensichtlich ist.

Bei den USVs werden Sie in aller Regel auch auf herstellerspezifische Plugins aus externen Quellen zurückgreifen oder Plugins selbst erstellen müssen (siehe Diskussion).

Pover-over-Ethernet (PoE) Injektoren überwachen

In modernen Umgebungen werden Geräte eingesetzt, um Komponenten über das Ethernet- Netzwerkkabel mit Strom zu versorgen. Die Technologie dahinter heißt Pover-over-Ethernet (kurz PoE). Da diese Geräte für die Stromversorgung anderer Geräte zuständig sind, spielen die Kennzahlen für Spannungen bei diesen natürlich eine besondere Rolle. Entsprechend können Sie beispielsweise den Stromverbrauch verfolgen und somit Rückschlüsse auf vorhandene Reserven ziehen.

Wenn bei Ihrer Umgebung beispielsweise die WLAN-Stationen über PoE versorgt werden, sollten Sie die dafür verwendeten Ports in die Überwachung einbinden. Sie können dann Abhängigkeiten für die jeweilig angeschlossenen Stationen einpflegen. Dies ermöglicht Ihnen bei Ausfällen zeitnah nachzuvollziehen, ob die WLAN-Station defekt ist oder lediglich nicht mit Strom versorgt wird.

SAN-Switches überwachen

Storage-Area-Networks (kurz SANs) bilden typischerweise ein eigenes Netzwerk, das sogenannte SAN-Netzwerk, in dem Geräte über eine eigene Infrastruktur angebunden sind. Häufig haben Sie aus Ihrem eigentlichen Netzwerk dann nur Zugriff auf einen SAN-Controller und sind für Prüfungen auf diesen festgelegt. Je nach Hersteller bieten solche Controller ganz unterschiedliche Daten – und nicht immer über Standards wie SNMP, sondern teilweise auch nur über eine eigens zu diesem Zweck bereitgestellte Software.

Mehr noch als bei den anderen besprochenen Geräten sind Sie hier also auf die herstellerspezifische Implementierung angewiesen. Prüfen Sie an dieser Stelle, ob Ihnen Daten über SNMP zur Verfügung stehen und welche Sie davon einbinden möchten. Oft lässt sich zumindest der allgemeine Zustand des Gerätes, dessen Bezeichner und Seriennummer sowie die Sensoren für Temperatur, Lüftergeschwindigkeit und Netzgerätstatus auslesen.

FibreAlliance: Eine MIB, die von einer Vielzahl von SAN-Switches unterstützt wird, ist die FibreAlliance. Diese sollten in jedem Fall bei Geräten der Hersteller IBM, Brocade, HP, Qlogic und EMC/Clarion funktionieren. Sie können dafür gegebenenfalls externe Plugins einsetzen, wie beispielsweise das Plugin `check_fibrealliance.sh` (siehe *http://exchange.nagios.org/directory/Plugins*). Wir haben dieses allerdings nicht getestet und gehen auch nicht weiter darauf ein.

Diskussion

Dieser Teil der Netzwerküberwachung ist der eigentliche Kern, dabei aber sehr divers ausgestaltet, so dass eine standardisierte Vorgehensweise schwer zu definieren ist. Selbst wenn alle Geräte von einem Hersteller stammen, sind die Unterschiede häufig groß. Deshalb ist die Einbindung von externen und/oder die eigene Anpassung von bestehenden Plugins (siehe hierzu Kapitel 7, *Erstellung eigener und Einbindung externer Plugins*) hier meist erforderlich.

Wir möchten Sie an dieser Stelle deshalb noch einmal auf die Plugin-Verzeichnisse unter *http://exchange.nagios.org/directory/Plugins* und *https://www.monitoringexchange.org/inventory/Check-Plugins* verweisen. Dort gibt es Tausende von Plugins, die eventuell genau das leisten, was Sie benötigen. Unsere Erfahrungen mit externen Plugins sind durchwachsen, was die Zuverlässigkeit und insbesondere Fehlerbehandlung betrifft. Aber selbst wenn Sie für eine bestimmte Aufgabe kein exakt passendes Plugin finden, so können Sie eventuell doch einen Teil der gewünschten Abfragen abdecken oder finden Hinweise zu eigenen Erweiterungen.

Der Aufwand zur detaillierten Einbindung ist hier häufig hoch. Wenn Sie Ihr Monitoring gerade erst aufbauen, sollten Sie zunächst die einfach zu implementierenden Prüfungen einbinden. Beim späteren Ausbau Ihres Monitoring-Systems können Sie weitere Anforderungen dann Stück für Stück nach Priorität abarbeiten. Alles auf einmal und abschließend einzubinden, wird Ihnen typischerweise nicht gelingen. Netzwerk-Monitoring ist mehr ein Prozess denn eine einmalige Einrichtung.

Siehe auch

- Rezept zur Planung der Netzwerkanbindung: Rezept 1.2, *Netzwerkanbindung des Monitoring-Servers planen*
- Rezept für generische Abfragen über SNMP: Rezept 6.7, *Erstellung generischer SNMP-Abfragen (check_snmp)*
- Rezept zur Einbindung von MIBs: Rezept 2.4, *SNMP mit MIBs verwenden*
- Rezept zu den Erreichbarkeitsprüfungen (host-checks): Rezept 4.1, *Maschinen einbinden (host)*
- Rezepte zu Abfragen von Schnittstellen mit Standard-Plugins: Rezept 6.8, *Schnittstellen über SNMP allgemein prüfen (check_ifstatus)* Rezept 6.9, *Den Status einzelner Schnittstellen über SNMP gezielt überpüfen (check_ifoperstatus)*
- Kapitel zu eigenen und externen Plugins: Kapitel 7, *Erstellung eigener und Einbindung externer Plugins*
- Rezept zur Einbindung externer Plugins am Beispiel der Manubulon-Plugins: Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*

- Rezept zur Zusammenfassung von Graphen als Special-Pages in PNP4Nagios: Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*
- Plugin-Börsen: *http://exchange.nagios.org/directory/Plugins* und *https://www.monitoringexchange.org/inventory/Check-Plugins*

8.5 Dateiserver überwachen

Problem

Sie betreiben einen oder mehrere Dateiserver, die Sie in die Überwachung einbinden möchten.

Lösung

Wir zeigen Ihnen im Folgenden auf, wie Sie die beschriebenen Plugins einsetzen können, um Dateiserver in Ihre Überwachung einzubeziehen. Binden Sie die entsprechenden Maschinen zunächst generell ein, wie in Rezept 8.1, *Überwachung von Unix-/Linux-Servern* und Rezept 8.2, *Überwachung von Windows-Maschinen* beschrieben. Dabei sollten Sie dann zunächst ein besonderes Augenmerk auf die Überwachung der Festplattenbelegung und die Auslastung der Netzwerkschnittstellen richten. Daneben ist der Zustand der Laufwerke auf Dateiservern natürlich von besonderer Wichtigkeit, weshalb Sie alle Laufwerke auch diesbezüglich überwachen sollten.

Von diesen eher allgemeinen Überwachungen abgesehen sollten Sie dann den von Ihnen verwendeten Zugriff auf die Dateisysteme testen. Da es hierzu eine Vielzahl von Möglichkeiten gibt, können wir nicht auf alle Varianten und Variationen eingehen. Wir fassen im Folgenden für Sie zusammen, welche Dienstarten Sie mit Hilfe der mitgelieferten Plugins überwachen können. Für andere Varianten müssen Sie sich gegebenenfalls entsprechende externe Plugins suchen.

Server Message Block (SMB) überwachen / Common Internet File System (CIFS) überwachen

Besonders häufig wird in der Praxis das SMB-Protokoll verwendet, um Dateien im Netzwerk verfügbar zu machen. SMB ist zwischen Windows-Systemen Standard und wird von Unix-Systemen ebenfalls unterstützt. Daher hat sich SMB in heterogenen Umgebungen weitestgehend durchgesetzt.

Das Protokoll wurde im Zuge von zahlreichen Erweiterungen später CIFS benannt, das intern aber weiter SMB verwendet. Entsprechend wird häufig auch von CIFS/SMB gesprochen. Entsprechende Netzwerk-Freigaben können Sie mit dem Plugin `check_smb` überwachen, wie in Rezept 6.16, *SMB-Dateidienst überwachen (check_smb)* beschrieben. Binden Sie dazu für jede Freigabe einen entsprechenden Test mit einer Dienstdefinition wie der im Rezept angegebenen ein:

```
define service{
       service_description                    SMB
       use                                    template-service
       check_command                          check_smb!gruppe!benutzer!123!freigabe
       host_name                              file-serv
}
```

Beachten Sie dabei zunächst, dass Sie die Angaben für Benutzer, Gruppe, Passwort und Freigabe an Ihre Umgebung anpassen müssen. Sie können dabei über die zusätzlichen Parameter `-w` und `-c` auch Schwellenwerte für die Belegung angeben und überwachen. Häufig sind die über eine Freigabe erreichbaren Daten über mehrere Platten verteilt. Auch wenn Sie die einzelnen Festplatten bei der allgemeinen Einbindung oben bereits entsprechend eingebunden haben, macht es dann Sinn, die Auslastung auf diesem Wege zusätzlich zu überwachen.

Sie testen damit die Erreichbarkeit über den spezifischen Dienst und dabei auch die dafür notwendige Authentifizierung. Zusätzlich können Sie auf dieser Ebene die Auslastung überwachen, was insbesondere dann Sinn macht, wenn die Freigabe auf zusammengesetzten Datenträgern aufbaut.

Network File System überwachen (NFS)

Unter Unix-Systemen wird der Zugriff auf Dateien meist über das verteilte Dateisystem NFS geregelt. NFS kann daher als das Unix-Pendant von SMB bezeichnet werden. Bei den Standard-Plugins wird leider kein Plugin mitgeliefert, das sich für die Überwachung von NFS-Freigaben eignet. Sie müssten sich daher gegebenenfalls um ein externes Plugins bemühen. Dazu kommt, dass die beiden zur Zeit verbreitetsten NFS-Versionen 3 und 4 sich grundlegend darin unterscheiden, wie der Client authentifiziert wird. Bis einschließlich NFS Version 3 authentifiziert sich der Client-Rechner gegenüber dem Server mit seinen Unix-Benutzer- und Gruppennummern. Erst ab Version 4 wird auch die bei SMB üblichere Benutzerauthentifizierung angeboten.

Wir empfehlen Ihnen unter NFS Version 3 das Plugin `check_rpc` für einen Basischeck, ob der NFS-Dienst überhaupt angeboten wird. Das Plugin ist Teil des Nagios-Plugins-Standardpakets. Sie können das Plugin beispielsweise mit folgenden Definitionen in Ihr Monitoring-System einbinden:

```
#'check_rpc' command definition
define command{
       command_name                           check_rpc
       command_line                           $USER1$/check_rpc -H $HOSTNAME$ -C $ARG1$
}

define service{
       service_description                    NFSv3
       check_command                          check_rpc!nfs
       host_name                              mrbig
}
```

Für das Monitoring eines NFS Version 4 könnten Sie beispielsweise auf das externe Plugin `check_nfs4` (siehe *http://nfsv4.bullopensource.org/doc/admin_tools/HOWTO_NFSv4_admin_2006.php*) zurückgreifen, das wir hier aber nicht näher beschreiben.

File Transfer Protocol überwachen (FTP)

Ein weiteres, immer noch häufig anzutreffendes Protokoll ist das FTP. Das entsprechende mitgelieferte Plugin `check_ftp` (siehe Rezept 6.10, *Überwachung eines FTP-Servers (check_ftp)*) erlaubt Ihnen nur einen minimalistischen Test der Erreichbarkeit des Dienstes. Mit der entsprechenden Dienstdefinition aus dem Rezept können Sie eine Überwachung durchführen:

```
define service{
        service_description                 FTP
        use                                 template-service-generic
        check_command                       check_ftp!-j
        host_name                           FTP-Server
}
```

Weitere Parameter wie etwa Passwörter oder eine Auslastungsprüfung wie bei SMB können Sie mit diesem Plugin leider nicht durchführen. Bei Bedarf können Sie hierzu jedoch ebenfalls auf entsprechende externe Plugins zurückgreifen oder sich gegebenenfalls mit einem selbstgeschriebenen Plugin behelfen.

Diskussion

Über die generelle Einbindung der entsprechenden Maschinen haben Sie für den Anwendungsfall des Dateiservers bereits die wichtigsten internen Parameter abgedeckt. Die Auslastung von Netzwerk-Schnittstellen und der Füllstand sowie der funktionale Status der Festplatten sind hier typischerweise die entscheidenden Werte. Die Einbindung einer funktionalen Prüfung der verwendeten Protokolle wie hier beschrieben ist allerdings eine notwendige Erweiterung dieser Prüfungen. Je nachdem, was für eine Konfiguration Sie verwenden, sind die mitgelieferten Plugins hierzu unter Umständen nicht ausreichend. Während Sie bei der Verwendung des SMB-Protokolls neben der Erreichbarkeit auch Authentifizierung und Füllstand abfragen können, bleibt bei der Verwendung von NFS und FTP nur noch die funktionale Prüfung übrig. Deshalb sollten Sie sich hier gegebenenfalls nach externen Plugins umsehen, um weitere Tests durchzuführen zu können. Dies gilt im besonderen Maße beim Einsatz von NFS.

Sie können auch gegebenenfalls die mit NFS und SMB verbundenen Prozesse auf dem Dateiserver überwachen. Dazu können Sie auf das Plugin `check_procs` zurückgreifen (siehe Rezept 5.3, *Prozesse überwachen (check_procs)*), müssen es dazu allerdings über SNMP (siehe Rezept 2.5, *Plugins unter Linux über SNMP ausführen*), SSH (siehe Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten*) oder NRPE (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*) einbinden. Wenn Sie SNMP auf der Maschine einsetzen, können Sie hierzu allerdings auch direkt die unter SNMP verfügbare Überwachung verwenden. Diese haben Ihnen im Rezept zur Überwachung von Hypervisoren (siehe Rezept 8.3, *Überwachung von Hypervisoren und virtuellen Maschinen*) beschrieben.

Siehe auch

- Rezepte zur generellen Überwachung von Unix/Linux und Windows-Maschinen: Rezept 8.1, *Überwachung von Unix-/Linux-Servern* und Rezept 8.2, *Überwachung von Windows-Maschinen*
- Rezepte zu den angesprochenen Plugins: Rezept 6.16, *SMB-Dateidienst überwachen (check_smb)*, Rezept 6.10, *Überwachung eines FTP-Servers (check_ftp)* und Rezept 5.3, *Prozesse überwachen (check_procs)*
- Hilfeseiten zum Plugin `check_nfs4`: *http://nfsv4.bullopensource.org/doc/admin_tools/HOWTO_NFSv4_admin_2006.php*
- Rezepte zur Remote-Ausführung von eigentlich lokalen Plugins: Rezept 2.5, *Plugins unter Linux über SNMP ausführen*, Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten*, Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*
- Rezept mit Beispiel zur Prozessüberwachung per SNMP Rezept 8.3, *Überwachung von Hypervisoren und virtuellen Maschinen*

8.6 E-Mail-Server überwachen

Problem

Sie möchten einen E-Mail-Server überwachen, sind sich aber unsicher, welche Tests Sie durchführen sollen.

Lösung

Wir machen Ihnen im Folgenden Vorschläge, wie Sie die beschriebenen Plugins sinnvoll einsetzen können, um unterschiedliche Aspekte von E-Mail-Servern zu überwachen. Sie sollten die allgemeinen Parameter der Maschine zunächst in Abhängigkeit vom verwendeten Betriebssystem entsprechend dem Rezept 8.1, *Überwachung von Unix-/Linux-Servern* und Rezept 8.2, *Überwachung von Windows-Maschinen* überwachen. Hier beschreiben wir Ihnen die spezifischen Erweiterungen für die Überwachung eines E-Mail-Servers.

E-Mail-Dienste grundlegend überwachen

Typischerweise werden heute für den Versand von E-Mails das Protokoll Simple Mail Transfer Protocol (SMTP) und für die Auslieferung die Protokolle Post Office Protocol (POP) und/oder Internet Message Access Protocol (IMAP) verwendet. Die Standard-Plugins `check_smtp`, `check_pop` und `check_imap` (siehe Rezept 6.13, *Überwachung des E-Mail-Versanddienstes (check_smtp)* und Rezept 6.14, *Überwachung der E-Mail-Auslieferungsdienste (check_pop, check_imap)*) ermöglichen Ihnen, diese auf unkomplizierte Weise zu überwachen. Sofern der zu überwachende E-Mail-Server entsprechende Zugänge ohne Ver-

schlüsselung anbietet, können Sie diese einfach einbinden, indem Sie die Kommando-Definition aus den Rezepten ohne weitere Parameter einbinden:

```
define service {
       service_description                    SMTP
       use                                    template-service
       check_command                          check_smtp
       host_name                              smtp.googlemail.com
}

define service {
       service_description                    IMAP
       use                                    template-service
       check_command                          check_imap
       host_name                              imap.googlemail.com
}

define service {
       service_description                    POP
       use                                    template-service
       check_command                          check_pop
       host_name                              pop.googlemail.com
}
```

Die dabei ausgeführten Prüfungen entsprechen dann einem Aufruf der Plugins mit dem Zielhost als einzige Option, für `check_smtp` also etwa so:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_smtp -H smtp.googlemail.com
SMTP OK - 0.069 sec. response time|time=0.068778s;;;0.000000
```

Entsprechend der zurückgegebenen Performancedaten, Sie erkennen diese wie immer an dem Pipe-Symbol »|« , erhalten Sie bei Verwendung von PNP4Nagios automatisch einen Graphen für die Antwortzeiten.

Verschlüsselung überwachen

Wenn Sie SMTP-Verschlüsselung mit `STARTTLS` verwenden, so können und sollten Sie dieses (gegebenenfalls zusätzlich) testen. Um die Verschlüsselung zu verwenden, nutzen Sie den Parameter `-S`. Für das voranstehende Beispiel mit `check_smtp` sähe das wie folgt aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_smtp -H smtp.googlemail.com -S
SMTP OK - 0.069 sec. response time|time=0.068778s;;;0.000000
```

Wenn dieser Test erfolgreich ist und Ihr SMTP-Server somit STARTTLS unterstützt, sollten Sie diesen Zugriff – gegebenenfalls zusätzlich zu der Prüfung der unverschlüsselten Verbindung – überwachen. Hierzu nutzen Sie eine um den Parameter `-S` ergänzte Service-Einbindung:

```
define service {
       service_description                    SMTP
       use                                    template-service
       check_command                          check_smtp!-S
       host_name                              smtp.googlemail.com
}
```

Die Plugins check_imap und check_pop unterstützten Verschlüsselung über SSL, die Sie ebenfalls mit einer Option -S aktivieren können. Wenn der betreffende Mail-Server diese Verfahren unterstützt, sollten Sie hier analog verfahren und ebenfalls zusätzlich einen entsprechenden Test durchführen.

Zertifikate überwachen

Sofern Sie bei den E-Mail-Protokollen auf eigenen Servern Verschlüsselung nutzen, sollten Sie dann in einem weiteren Test die dafür eingesetzten Zertifikate überwachen. Diese haben typischerweise eine Ablaufzeit und müssen immer wieder erneuert werden, so dass eine Warnung vor dem baldigen Ablaufen sehr nützlich sein kann. Um das Ablaufdatum des Zertifikats zu überwachen, können Sie hierzu zusätzlich zu der Option -S den Parameter -D verwenden. Dabei übergeben Sie die Anzahl der Tage, die das Zertifikat mindestens noch gültig sein muss, damit keine Warnung ausgegeben wird. Für SMTP mit einer Warnzeit von 60 Tagen also beispielsweise:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_smtp -H smtp.googlemail.com -S -D 60
OK - Certificate will expire on 01/04/2013 23:41.
```

Beachten Sie, wie sich dabei im Vergleich zu den vorherigen Aufrufen die Ausgabe ändert. In diesem Modus gibt das Plugin keine Performancedaten mehr zurück, so dass Sie zu dieser Prüfung also keine Graphen erhalten werden. Analog können Sie diese Prüfungen bei POP und/oder IMAP für die dort verwendete Verschlüsselung per SSL einsetzen, z.B. folgendermaßen:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_imap -H imap.googlemail.com -p 993
-S -D 60
OK - Certificate will expire on 06/07/2013 19:43.
IMAP OK - 0.068 second response time on port 993 [* OK Gimap ready for requests from
176.9.84.37 b10if1600905bkq.42]|time=0.067680s;;;0.000000;10.000000
```

Zur Einbindung einer entsprechenden Prüfung fügen Sie eine weitere Dienst-Definition hinzu, die um den Parameter -D mit einer entsprechenden Angabe von Tagen zur Restlaufzeit erweitert wird, also für SMTP etwa wie folgt:

```
define service {
       service_description                    SMTP
       use                                    template-service
       check_command                          check_smtp!-S -D 60
       host_name                              smtp.googlemail.com
}
```

Da Sie vor dem Ablaufen der Zertifikate neue installieren sollten, ist dies eine recht wichtige Prüfung, um Unannehmlichkeiten für die Benutzer zu vermeiden. Wenn Sie nicht persönlich für die Zertifikate zuständig sind, weisen Sie dem Dienst einen entsprechenden Kontakt zu (siehe hierzu Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* beziehungsweise Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*).

Authentifizierung auf Funktion überwachen

Je nach Konfiguration eines Servers kann die Authentifizierung an weitere Systeme, etwa einen LDAP-Server, gebunden sein. In diesen Fällen können Sie zusätzlich testen, ob die Authentifizierung wie gewünscht funktioniert. Das Plugin **`check_smtp`** ermöglicht Ihnen, Daten zur Authentifizierung zu übergeben und diese zu testen. Setzen Sie für eine entsprechende Überprüfung mit der Option `-A LOGIN` das Authentifizierungsverfahren und übergeben Sie den Benutzernamen mit `-U` und das dazugehörige Passwort mit `-P`. Für eine verschlüsselte Verbindung sieht das etwa wie folgt aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_smtp -H smtp.googlemail.com -S -A LOGIN
-U "USERNAME" -P "PASSWORD" -v
HELOCMD: EHLO moni
220 mx.google.com ESMTP f1sm46453317wiy.2
sent EHLO moni
250-mx.google.com at your service, [176.9.84.37]
250-SIZE 35882577
250-8BITMIME
250-AUTH LOGIN PLAIN XOAUTH XOAUTH2
250 ENHANCEDSTATUSCODES
sent AUTH LOGIN
received 334 VXNlcm5hbWU6

sent bXdpbmQ5Nw==

received 334 UGFzc3dvcmQ6

sent U2Vuc29yMjQwYXh0

received 235 2.7.0 Accepted

sent QUIT
received 221 2.0.0 closing connection f1sm46453317wiy.2

SMTP OK - 0.436 sec. response time, 221 2.0.0 closing connection f1sm46453317wiy.2
|time=0.435920s;;;0.000000
```

Einbinden sollten Sie das Plugin allerdings ohne die Option `-v`. Dann fallen die ausführlichen Ausgaben weg und das Plugin gibt nur die relevanten Daten zurück:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_smtp -H smtp.googlemail.com -S -A LOGIN
-U "USERNAME" -P "PASSWORD"
SMTP OK - 0.458 sec. response time|time=0.458078s;;;0.000000
```

Authentifizierungsdaten: Für die Einbindung eines entsprechenden Dienstes müssen Sie das Passwort für den zu testenden Benutzer hinterlegen. Dieser steht ist dann im Klartext in den Konfigurationsdateien des Monitoring-Systems aufgeführt, weshalb Sie hierzu einen eigenen Benutzer vorsehen sollten. Damit das Passwort nicht zusätzlich in den Log-Dateien auftaucht, sollten Sie es als Benutzermakro (siehe Rezept 4.3, *Befehle hinzufügen (command)*) hinterlegen.

Eine entsprechend erweiterte Service-Definition ist dann etwa diese:

```
define service {
       service_description                  SMTP
       use                                  template-service
       check_command                        check_smtp!-S -A LOGIN -U "mailtest" \
                                            -P $USER15$
       host_name                            smtp.googlemail.com
}
```

Wir haben hierbei wie vorgeschlagen das Passwort als Benutzermakro `$USER15$` hinterlegt und nur dieses Makro referenziert.

Diskussion

Mit den mitgelieferten Plugins können Sie wie beschrieben die üblicherweise eingesetzten Dienste auf Erreichbarkeit testen und darüber hinaus besondere Aspekte wie Funktionsfähigkeit der Verschlüsselung und ablaufende Zertifikate überwachen. Die entsprechenden Prüfungen können Sie leicht einbinden und erreichen damit bereits eine gute Testabdeckung. Das so entstehende Prüfungsszenario können Sie aber durchaus noch erweitern. Sie könnten beispielsweise testen, dass eine E-Mail auch tatsächlich zugestellt wird. Hierzu müssen Sie aber externe (oder eigene) Plugins einbinden, so dass die entsprechenden Arbeiten aufwendiger sind. Gleichzeitig steigt der Mehrwert gegenüber den einfach einzubindenden Tests nur wenig, da nur sehr spezifische Fehler zusätzlich abgefragt werden können.

Siehe auch

- Rezepte zur generellen Einbindung von Maschinen unter Unix/Linux beziehungsweise Windows: Rezept 8.1, *Überwachung von Unix-/Linux-Servern* und Rezept 8.2, *Überwachung von Windows-Maschinen*
- Rezepte zu den hier angesprochenen Plugins: Rezept 6.13, *Überwachung des E-Mail-Versanddienstes (check_smtp)* und Rezept 6.14, *Überwachung der E-Mail-Auslieferungsdienste (check_pop, check_imap)*
- Rezepte mit Details zur Einrichtung von Benachrichtigungen für Dienste: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*, Rezept 3.11, *Maßgeschneiderte Benachrichtigungen einrichten*
- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*

8.7 Webserver überwachen

Problem

Sie möchten einen Webserver überwachen, wissen aber nicht genau, welche Prüfungen Sie einsetzen sollten.

Lösung

Zunächst sollten Sie die betreffende Maschine entsprechend Rezept 8.1, *Überwachung von Unix-/Linux-Servern* beziehungsweise Rezept 8.2, *Überwachung von Windows-Maschinen* einbinden. Damit überwachen Sie bereits eine Anzahl von relevanten Kennzahlen. Insbesondere sollten Sie ein besonderes Augenmerk auf Prozessor-, Speicher-, Netzwerk- und Plattenauslastung richten. Im Folgenden unterbreiten wir Ihnen eine Reihe von Vorschlägen zur gezielten Überwachung der Webserver-Dienste, anhand derer Sie sich ein für Sie geeignetes Prüfungsinstrumentarium zusammenstellen können.

Prozesse überwachen

Je nachdem, welchen Webserver Sie wie einsetzen, hängt die Anzahl an Serverprozessen möglicherweise von der Anzahl der Anfragen ab. In diesem Fall können Sie den Wert der parallel laufenden Serverprozesse überwachen und gegebenenfalls entsprechende Graphen zeichnen lassen. Hierzu können Sie das Standard-Plugin `check_procs` verwenden (siehe Rezept 5.3, *Prozesse überwachen (check_procs)*) und es über SNMP (siehe Rezept 2.5, *Plugins unter Linux über SNMP ausführen*), SSH (siehe Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten*) oder NRPE (siehe Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*) einbinden oder bei Verwendung von SNMP direkt den dafür vorgesehenen Mechanismus zur Prozessüberwachung nutzen. Da wir Ihnen die anderen Varianten in den entsprechenden Rezepten beschrieben haben, zeigen wir Ihnen im Folgenden, wie Sie die letzte Variante umsetzen können.

Damit Sie die Prozessüberwachung von SNMP nutzen können, erweitern Sie Ihre SNMP-Konfigurationsdatei auf der betreffenden Maschine wie folgt um eine entsprechende Zeile (hier am Beispiel des Webservers Apache):

```
root@docu:~# vim /etc/snmp/snmpd.conf
[...]
# Add apache to UCD-SNMP-MIB::prTable
proc apache2
[...]
```

Starten Sie den SNMP-Dienst `snmpd` neu, damit diese Änderung wirksam wird:

```
root@docu:~# service snmpd restart
Restarting network management services: snmpd.
```

Testen Sie dann von der überwachenden Maschine aus, ob Sie unter dem OID `.iso.org.dod.internet.private.enterprises.ucdavis.prTable` entsprechende Einträge vorfinden:

```
icinga@moni:~$ snmpwalk -Of -v 3 -l authNoPriv -u icinga -a sha
-A "Passphrase" 10.85.58.41 .iso.org.dod.internet.private.enterprises.ucdavis.prTable
.iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prIndex.1 = INTEGER: 1
.iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prNames.1 = STRING:
apache2
.iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prMin.1 = INTEGER: 0
.iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prMax.1 = INTEGER: 0
.iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prCount.1 = INTEGER: 11
.iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prErrorFlag.1 = INTEGER
```

```
: noError(0)
.iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prErrMessage.1 =
STRING:
.iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prErrFix.1 =
INTEGER: noError(0)
.iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prErrFixCmd.1 = STRING:
```

Die Anzahl der Prozesse ist hier unter dem OID `.iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prCount.1` abrufbar. Sie können jetzt das Standard-Plugin `check_snmp` einsetzen, um diesen Wert zu überwachen und bei Verwendung von PNP4Nagios auch zu graphen. Wenn Sie das Plugin wie im Rezept beschrieben eingebunden haben, benötigen Sie nur noch eine entsprechende Service-Definition. Ermitteln Sie hierzu zunächst die numerische Darstellung des OID, um bei den automatisierten Abfragen die entsprechenden Konvertierungen einzusparen:

```
icinga@moni:~$ snmptranslate
-On .iso.org.dod.internet.private.enterprises.ucdavis.prTable.prEntry.prCount.1
.1.3.6.1.4.1.2021.2.1.5.1
```

Testen Sie die Abfrage dann zunächst von der Kommandozeile:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_snmp -H 10.85.58.41 -P 3 -L authNoPriv
-a SHA -U "icinga" -A "PASSPHRASE" -o .1.3.6.1.4.1.2021.2.1.5.1 -l "Apache Processes"
SNMP OK - Apache Processes 11 | Apache Processes=11
```

Nutzen Sie dann aus Gründen der Performance am besten die numerische Darstellung für die Einbindung:

```
define service{
       service_description                    check_apache
       use                                    template-service
       check_command                          check_snmp3!.1.3.6.1.4.1.2021.2.1.5.1! \
                                              -l "Apache Processes"
       hostgroup_name                         docu
}
```

Beachten Sie, dass hier gegebenenfalls die folgenden Werte an Ihre Umgebung angepasst werden müssen:

- die verwendete Vorlage `template-service`
- das verwendete Kommando `check_snmp3`
- die Parameter für das Kommando und insbesondere der OID
- die Zuweisung an die Maschine `docu`

Da das Plugin den Rückgabewert auch in Form von Performancedaten zurückgibt, erhalten Sie bei Verwendung von PNP4Nagios automatisch einen Graphen zu der Prüfung:

Mit der Anzahl der Prozesse haben Sie dann bereits eine wichtige Kennzahl und mit dem entsprechenden Graphen auch die Möglichkeit der zeitlichen Verfolgung.

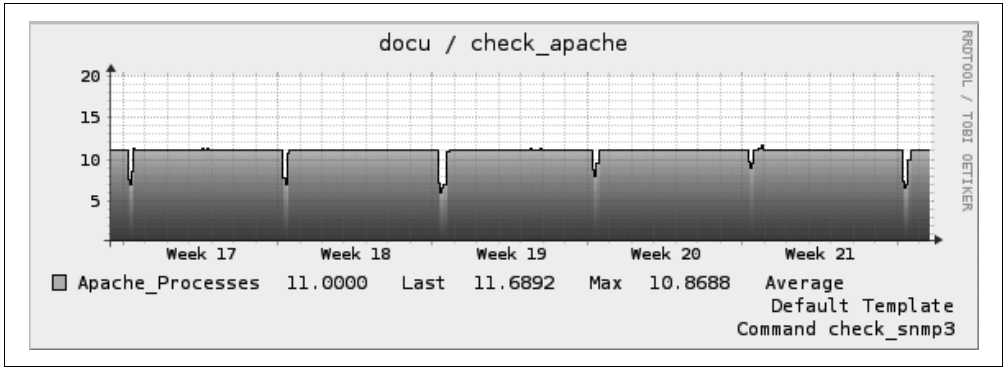


Abbildung 8-13: PNP4Nagios-Graph zu check_snmp auf Apache-Procs

Erreichbarkeit überwachen

Webserver bieten Ihre Dienste über Internet-Serviceports an. Die bei Ihnen hierzu genutzten Ports sollten Sie entsprechend überwachen. Hierzu bietet sich zunächst das Standard-Plugin `check_http` (siehe Rezept 6.11, *Überwachung eines Webservers (check_http)*) an. Setzen Sie das Plugin mit entsprechenden Optionen für alle bei Ihnen verwendeten Ports ein. Für unverschlüsselten HTTP-Verkehr über den Standardport 80 können Sie die folgende Definition verwenden:

```
define service{
        service_description                     HTTP
        use                                     template-service
        check_command                           check_http!--warning 50 --critical 100
        host_name                               docu
}
```

Diese Definition baut wie in Rezept 6.11, *Überwachung eines Webservers (check_http)* beschrieben auf der in der Standard-Installation mitinstallierten Kommandodefinition für `check_http` auf. Über die Option `-p` für das Plugin können Sie gegebenenfalls weitere Definitionen für beliebige Ports einbinden oder diese für die verwendeten Ports anpassen. Mit den Optionen `--warning` und `--critical` haben wir hier außerdem Schwellenwerte für die Antwortzeiten angegeben, ab denen sich der Status entsprechend ändern wird.

Zusätzlich zu der prinzipiellen Erreichbarkeit des HTTP-Dienstes auf dem gewünschten Port können Sie dann weiter gezielt die auf diesem Port angebotenen Seiten überwachen. Nehmen Sie dazu jeweils eine Service-Definition mit folgenden Anpassung auf:

```
check_command check_http!--url "/book"
```

und

```
check_command check_http!--url "/docu"
```

Natürlich müssen Sie dabei die von Ihnen verwendeten URLs einsetzen. Auf diese Weise können Sie gezielt die angebotenen Dienste überwachen.

Erreichbarkeit mit SSL-Verschlüsselung überwachen

Wenn Ihr Webserver ebenfalls SSL-Verschlüsselung verwendet, sollten Sie auch dies testen. Eine entsprechend angepasste Zeile der obigen Definition ist die Folgende:

```
check_command check_http!--ssl --warning 50 --critical 100
```

Dadurch verändert sich der entsprechende Test und wird SSL-verschlüsselt über den Standardport 443 durchgeführt. Wenn Sie abweichende Ports verwenden, müssen Sie diese (gegebenenfalls zusätzlich) über die Option `-p` angeben. Die Beispielzeile aktiviert neben der SSL-Prüfung und der damit verbundenen Umschaltung des verwendeten Ports auch wieder eine Prüfung der Antwortzeiten.

SSL-Zertifikat überwachen

Für jedes genutzte SSL-Zertifikat sollten Sie darüber hinaus gegebenenfalls einen Test auf das Ablaufdatum des verwendeten Zertifikats einrichten, um sich frühzeitig an anstehende Zertifikatserneuerungen erinnern zu lassen. Eine entsprechend angepasste Zeile ist

```
check_command check_http!--ssl -C 90
```

für einen Test auf mindestens 90 Tage verbleibender Gültigkeit. Ein entsprechender Service hat dann einen Rückgabewert ohne Performance-Daten:

```
OK - Certificate will expire on 03/20/2013 12:54.
```

Diese Prüfung dient wirklich nur der Erinnerung bei ablaufenden Zertifikaten, die Sie über eine entsprechende Kontaktdefinition (siehe Rezept 4.4, *Kontakte definieren (contact)*) direkt an den jeweiligen Verantwortlichen adressieren können.

Authorisierung überwachen

Wenn Sie für Webseiten eine HTTP-Authentifikation voraussetzen, können Sie testen, ob diese auch aktiv ist. Fragen Sie hierzu eine Seite ab, ohne die Authentifizierungsdetails zu übergeben, und testen Sie auf den Rückgabe-Code 401. Eine entsprechend angepasste Zeile für die Service-Definition ist diese:

```
check_command check_http!--expect="401 Authorization Required"
```

Ein entsprechender Test auf einer Monitoring-Maschine mit Icinga über HTTPS (bei dem standardmäßig eine HTTP-Authorisierung vorgesehen ist), sieht dann beispielsweise wie folgt aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_http -I 127.0.0.1 --url="/icinga" --ssl
--expect="401 Authorization Required"
HTTP OK: Status line output matched "401 Authorization Required" - 742 bytes in
0.006 second response time |time=0.006024s;;;0.000000 size=742B;;;0
```

Sie sollten mit Nagios/Icinga über ein Web-Interface mit HTTP-Authentifizierung verfügen. Dazu hier eine vollständige Service-Definition für das vorangegangene Beispiel, dass Sie schnell an Ihre Umgebung anpassen können:

```
define service{
       service_description                    HTTP-Auth-Icinga
       use                                    template-service-generic
       check_command                          check_http!--url="/icinga" --ssl \
                                              --expect="401 Authorization Required"
       host_name                              localhost
}
```

Inaktive Vorgabe-Ports überwachen

Wenn Sie einen oder auch beide der Standard-Ports nicht verwenden, können Sie das Standard-Plugin `check_tcp` (siehe Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*) verwenden, um zu überprüfen, dass auf diesem Port auch wirklich kein Dienst läuft. Sie können auf diese Weise sicherstellen, dass es nicht etwa ein Angreifer geschafft hat, einen Dienst auf diesem Port anzubieten, indem er etwa eine Konfigurationsdatei für Ihren Server geändert oder hinzugefügt hat. Dazu müssen Sie dann gezielt angeben, dass die Zurückweisung eben erwünscht ist. Hier ein Beispiel der Aufrufe des Plugins zunächst ohne und dann mit der entsprechenden Option:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_tcp -H conf -p 80
Connection refused
icinga@moni:~$ /usr/local/icinga/libexec/check_tcp -H conf -p 80 --refuse=ok
TCP OK - 0.000 second response time on port 80|time=0.000307s;;;0.000000;10.000000
```

Zur Einbindung können Sie die folgende Definition verwenden:

```
define service{
       service_description                    Port80
       use                                    template-service
       check_command                          check_tcp!80!--refuse=ok
       host_name                              conf
}
```

Beachten Sie, dass Sie hier gegebenenfalls die folgenden Werte an Ihre Umgebung anpassen müssen:

- die Bezeichnung für den Service `Port80`
- die verwendete Vorlage **`template-service`**
- den verwendeten Port in der Kommandozeile
- die Angabe der zu prüfenden Maschine `conf`

Sie erhalten mit diesem Test zwar auch Performancedaten und bei Verwendung von PNP4Nagios einen Graphen, allerdings ist dieser bei dem hier angeführten Test vermutlich nur bei sehr ungewöhnlichen Problemen eine besondere Hilfe.

Diskussion

Die vorgeschlagenen Prüfungen bieten Ihnen eine Überwachung der Prozesse und beliebiger Ports. Beim Einsatz von SSL können Sie zusätzlich auf ablaufende Zertifikate und auf aktivierte Authorisierung über HTTP-Auth prüfen .

Neben der oben vorgeschlagenen Beobachtung der Anzahl der Prozesse lassen sich gegebenenfalls weitere Prüfungen für Prozesse einbinden. Mit dem Standard-Plugin `check_procs` können Sie wie in Rezept 5.3, *Prozesse überwachen (check_procs)* beschrieben auch die Speicherauslastung oder CPU-Nutzung dieser Prozesse beobachten und gegebenenfalls mit entsprechenden Schwellen für Benachrichtigungen versehen.

Überwachung von Weiterleitungen

Auch das Plugin `check_http` können Sie nach Bedarf mit weiteren Optionen einsetzen. Auch wenn wir hier nicht alle sinnvollen Variationen darstellen können, möchten wir Ihnen folgendes weiteres Beispiel zu diesem Plugin vorstellen:

Bei einer Kompromittierung eines Webservers hatte ein Angreifer über eine Modifikation der Datei *.htaccess* Umleitungen auf eine Schadseite eingerichtet. Dabei hat er allerdings nur bei Weiterleitung durch eine Suchmaschine reagiert, so dass alle Standard-Überwachungen einwandfrei durchgeführt wurden, aber neue Besucher von einer Suchmaschine her kommend auf eine Schadseite weitergeleitet wurden.

Um solche Angriffe künftig zu erkennen, setzen wir `check_http` ein. Hierzu übergeben wir als Referer-Header die Adresse einer Suchmaschine und lassen das Plugin bei einer Weiterleitung auf den Status `WARNING` springen. Die entsprechende Service-Definition sieht so aus:

```
define service {
       service_description                    HTTP-REDIR
       use                                    template-service
       check_command                          check_http!--header= \
                                              "Referer:http://www.google.com" \
                                              --onredirect=warning
       host_name                              docu
}
```

Durch diese zusätzliche Überwachung können auch Sie entsprechende Angriffe erkennen. Binden Sie sie dazu unter Anpassung der folgenden Werte für Ihre Umgebung ein:

- die verwendete Vorlage `template-service`
- die zu prüfende Maschine `docu`

Definition von Service-Abhängigkeiten

Wenn Sie wie oben dargestellt mehrere unterschiedliche Prüfungen durchführen, die alle davon abhängen, dass der Dienst grundsätzlich läuft, empfehlen wir Ihnen ebenfalls Service-Abhängigkeiten zu definieren. Sie können auf diese Weise unnötige Benachrichtigungen unterbinden. Dies ist insbesondere von Bedeutung, wenn Sie für diese

unterschiedliche Ansprechpartner hinterlegt haben. Zählen Sie dazu die abhängigen Dienste in einer Abhängigkeitsdefinition auf und verweisen Sie auf die grundsätzliche Prüfung des Dienstes. Um also etwa auf die oben als `HTTP-book` und `HTTP-docu` beschriebenen Prüfungen von der ebenfalls dort beschriebenen Prüfung `HTTP` abhängig zu machen, nutzen Sie folgende Syntax:

```
define servicedependency {
        dependent_host_name                     docu
        dependent_service_description           HTTP-book,HTTP-docu
        host_name                               docu
        service_description                     HTTP
}
```

Passen Sie hierbei die Namen der Maschine und Dienste an Ihre Umgebung an. Die Bedeutung der Hinterlegung solcher Abhängigkeiten steigt mit der Anzahl der abhängigen Dienste und betroffenen Ansprechpartner an. Sie müssen entscheiden, ob sich der zusätzliche Konfigurationsaufwand für Sie lohnt.

Siehe auch

- Rezepte zur generellen Überwachung von Maschinen: Rezept 8.1, *Überwachung von Unix-/Linux-Servern*, Rezept 8.2, *Überwachung von Windows-Maschinen*
- Rezepte zu den hier besprochenen Plugins: Rezept 5.3, *Prozesse überwachen (check_procs)*, Rezept 6.11, *Überwachung eines Webservers (check_http)*, Rezept 6.3, *Überwachung beliebiger IP-Serviceports (check_tcp/check_udp)*
- Rezepte zur Remote-Durchführung lokaler Plugins: Rezept 2.5, *Plugins unter Linux über SNMP ausführen*, Rezept 2.9, *SSH auf Linux-Maschinen vorbereiten*, Rezept 2.7, *NRPE auf Unix-Maschinen vorbereiten*
- Rezepte zu Maschinen, Dienst- und Befehlsdefinitionen: Rezept 4.1, *Maschinen einbinden (host)*, Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* und Rezept 4.3, *Befehle hinzufügen (command)*
- Rezept zur Vereinfachung der Konfiguration mit Vorlagen: Rezept 3.5, *Vereinfachen der Konfiguration mit dem Vorlagen-System*
- Rezept zur Definition von Kontakten: Rezept 4.4, *Kontakte definieren (contact)*

8.8 Datenbankserver überwachen

Problem

Sie möchten Ihre Datenbankserver überwachen, sind sich aber unsicher, welche Prüfungen Sie dafür einsetzen sollten.

Lösung

Binden Sie die Maschine(n) allgemein ein, indem Sie zunächst Prüfungen entsprechend der verwendeten Plattform verwenden (siehe Rezept 8.1, *Überwachung von Unix-/Linux-*

Servern beziehungsweise Rezept 8.2, *Überwachung von Windows-Maschinen*). Im Folgenden gehen wir nur auf Erweiterungen bezüglich Datenbanken ein. Zu diesen zeigen wir Ihnen auf, wie Sie diverse Eigenschaften überprüfen und diese entsprechend in Ihr Monitoring-System einbinden können.

Datenbank-Prozesse überwachen

Abhängig von der bei Ihnen verwendeten Datenbank können Sie zunächst generisch die entsprechenden Prozesse mit check_procs überprüfen (siehe Rezept 5.3, *Prozesse überwachen (check_procs)*). Um also bei einer MySQL-Datenbank zu testen, dass (nur) genau 1 Prozess mit dem Namen `mysqld` läuft, können Sie etwa den folgenden Aufruf verwenden:

```
icinga@db:~$ /usr/lib/nagios/plugins/check_procs -C mysqld -w 1:1 -c 1:1
PROCS OK: 1 process with command name 'mysqld'
```

Alternativ oder zusätzlich zur Überwachung der Prozessanzahl können Sie mit entsprechenden Optionen auch gezielt die Speicher- oder CPU-Nutzung der Prozesse überprüfen.

Zur Durchführung auf anderen Maschinen können Sie das Plugin über SNMP einbinden (siehe Rezept 2.5, *Plugins unter Linux über SNMP ausführen*). Dazu müssen Sie die Plugins auf der jeweiligen Maschine verfügbar machen und dann über die Konfigurationsdatei des `snmpd` einbinden. Für das vorherige Beispiel sähe das etwa folgendermaßen aus:

```
extend check_procs_mysqld /usr/lib/nagios/plugins/check_procs -C mysqld -w 1:1 -c 1:1
```

Anschließend liefert der SNMP-Daemon `snmpd` Ihnen bei Abfrage die entsprechenden Rückgaben des Plugins. Im vorherigen Fall sind die relevanten OIDs etwa folgende:

```
.iso.org.dod.internet.private.enterprises.netSnmp.netSnmpObjects.nsExtensions.
nsExtendObjects.nsExtendOutput1Table.nsExtendOutput1En
try.nsExtendResult."check_procs_mysqld" = INTEGER: 0
.iso.org.dod.internet.private.enterprises.netSnmp.netSnmpObjects.nsExtensions.
nsExtendObjects.nsExtendOutput2Table.nsExtendOutput2En
try.nsExtendOutLine."check_procs_mysqld".1 = STRING: PROCS OK: 1 process with command
name 'mysqld'
```

Zur Einbindung in das Monitoring können Sie dann beispielsweise generisch mit `check_snmp` den Rückgabewert abfragen oder wie in Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen* beschrieben beide Werte abfragen und zu einer Rückgabe zusammenfassen. Damit erhalten Sie dann auf der Monitoring-Maschine beispielsweise die folgende Ausgabe zum voranstehenden Test:

```
icinga@moni:~$ /usr/local/icinga/libexec2/wrapper_check_snmp_plugin.sh 10.85.58.42
check_procs_mysqld
PROCS OK: 1 process with command name 'mysqld' |
```

Diese Prüfung können Sie dann mit Hilfe der folgenden Dienstdefinition einbinden (wir setzen hier voraus, dass Sie ein Wrapper-Script wie in Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen* als Kommando `wrapper_check_snmp_ plugin` konfiguriert haben):

```
define service{
        service_description                     MySQL Process Count
        use                                     template-service
        check_command                           wrapper_check_snmp_plugin!check_\
                                                procs_mysqld
        host_name                               db
}
```

Beachten Sie bitte, dass Sie die verwendete Vorlage `template-service` und die Zuweisung an Maschinen (hier an die einzelne Maschine `db`) vermutlich an Ihre Umgebung anpassen müssen.

Weitere Prozesse überwachen

Abhängig von der Nutzung Ihrer Datenbank möchten Sie eventuell weitere relevante Prozesse überwachen. Als Beispiel erläutern wir Ihnen hier die Beobachtung von IDO2DB-Prozessen, da Sie diese wahrscheinlich sowieso für Ihren Monitoring-Server nutzen (siehe zur Installation und Rolle von IDO2DB unter Icinga beziehungsweise NDO2DB unter Nagios Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*).

Damit die Datenbankanbindung mittels IDO2DB funktioniert, müssen zu jeder Zeit genau zwei entsprechende Prozesse laufen. Der eine wird vor dem Start von Icinga gestartet, der andere sobald sich das Monitoring-System verbunden hat. Laufen diese nicht, so funktioniert das Monitoring selbst zunächst weiter, nur werden die Daten dann eben nicht mehr in die Datenbank geschrieben. Damit Sie mögliche Probleme mit dieser Datenbankanbindung bemerken, können Sie nun also die Anzahl der laufenden IDO2DB-Prozesse überwachen. Rufen Sie hierzu das Plugin mit entsprechenden Optionen zur Auswahl und Anzahl der Prozesse auf (hier wieder am Beispiel unseres Referenzsystems):

```
icinga@moni:~$ /usr/local/icinga/libexec/check_procs -C ido2db -w 2:2 -c 2:2
PROCS OK: 2 processes with command name 'ido2db'
```

Die Schwellenwerte für die Status `WARNING` und `CRITICAL` sind hier gleich gewählt, da jede von 2 abweichende Anzahl in diesem Fall ein kritisches Problem darstellt. Eine lokale Einbindung für den Fall, dass Ihre Datenbank auf dem Monitoring-Server selbst läuft, können Sie mit der folgenden Definition erreichen:

```
define service{
        service_description                     ido2db Process
        use                                     local-service
        check_command                           check_procs!-C ido2db -w 2:2 -c 2:2
        host_name                               localhost
}
```

Beachten Sie hier, dass Sie die verwendete Vorlage `local-service` gegebenenfalls an Ihre Umgebung anpassen müssen.

MySQL-Datenbanken überwachen

Für die Prüfung von MySQL-Datenbanken (also auch MariaDB) können Sie das Standard-Plugin `check_mysql` verwenden, um einige Status des Datenbanksystems abzufragen. Wenn Sie eine andere Datenbank verwenden, müssten Sie hier zunächst nach einem äquivalenten Plugin suchen und dieses einbinden und testen.

Der Aufruf des Plugins `check_mysql`, um die allgemeinen Daten abzufragen, sieht so aus:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_mysql -H db -u sql_monitor -p "PASSWORD"
Uptime: 4483800 Threads: 3 Questions: 32158954 Slow queries: 8 Opens: 53154 Flush tables:
 1 Open tables: 64 Queries per second avg: 7.172
```

Die Einbindung dieser Prüfung mit dem Plugin `check_mysql` finden Sie in Rezept 6.17, *MySQL-Datenbanken überwachen (check_mysql)*. Da das Plugin leider keine Performancedaten liefert, erhalten Sie so allerdings zunächst keine Graphen dieser eigentlich für diesen Zweck gut geeigneten Daten. Sie können sich hier mit einem Wrapper wie in Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* beschrieben behelfen. Wir haben für Sie ein entsprechendes Script vorbereitet:

```
#!/bin/bash

# Auslesen der Parameter
HOST_NAME=$1

# Ausführen des Befehls und speichern der textuellen Rückgabe in der Variable LINE
LINE=`/usr/local/icinga/libexec/check_mysql -H $HOST_NAME -u sql_monitor -p "PASSWORD"`

# Speicherung des nummerischen Rückgabewerts in der Variable RC
RC=$?

# Ermittlung der Zahlenwerte und Speicherung in Variablen
UPTIME=`echo $LINE | egrep -o "Uptime: [0-9]{1,9}" | egrep -o "[0-9]{1,9}"`
THREADS=`echo $LINE | egrep -o "Threads: [0-9]{1,9}" | egrep -o "[0-9]{1,9}"`
QUESTIONS=`echo $LINE | egrep -o "Questions: [0-9]{1,9}" | egrep -o "[0-9]{1,9}"`
SLOW_QUERIES=`echo $LINE | egrep -o "Slow queries: [0-9]{1,9}" | egrep -o "[0-9]{1,9}"`
OPENS=`echo $LINE | egrep -o "Opens: [0-9]{1,9}" | egrep -o "Opens: [0-9]{1,9}"`
FLUSH_TABLES=`echo $LINE | egrep -o "Flush tables: [0-9]{1,9}" | egrep -o "[0-9]{1,9}"`
OPEN_TABLES=`echo $LINE | egrep -o "Open tables: [0-9]{1,9}" | egrep -o "[0-9]{1,9}"`
QUERIES_PER_SECOND=`echo $LINE | egrep -o "Queries per second avg: [0-9]{1,9}\.
[0-9]{1,9}" | egrep -o "[0-9]{1,9}\.[0-9]{1,9}"`

# Ausgabe der Rückgabe ergänzt um Performancedaten
echo $LINE \| uptime=$UPTIME threads=$THREADS questions=$QUESTIONS slow=$SLOW_QUERIES
opens=$OPENS flush_tables=$FLUSH_TABLES open_tables=$OPEN_TABLES avg_queries=$QUERIES_
PER_SECOND

# Rückgabe des erhaltenen nummerischen Rückgabewertes
exit $RC
```

Beachten Sie, dass Sie den hier verwendeten Benutzernamen und das dazugehörige Passwort sowie gegebenenfalls den verwendeten Pfad an Ihre Umgebung anpassen müssen.

Wenn die Ausgabe des Plugins geändert wird, müssen Sie das Script weiter modifizieren. Testen Sie Ihr Script deshalb zunächst von der Kommandozeile aus:

```
icinga@moni:~$ /usr/local/icinga/libexec2/wrapper_check_mysql.sh db
Uptime: 4675177 Threads: 4 Questions: 34051718 Slow queries: 8 Opens: 55133 Flush tables:
 1 Open tables: 64 Queries per second avg: 7.283 | uptime=4675177 threads=4 questions=340
51718 slow=8 opens=55133 flush_tables=1 open_tables=64 avg_queries=7.283
```

Um das Plugin über das Wrapper-Script einzubinden, definieren Sie dann zunächst ein entsprechendes Kommando:

```
define command {
       command_name                    wrapper_check_mysql
       command_line                    $USER2$/wrapper_check_mysql.sh \
                                       $HOSTADDRESS$
}
```

Dieses können Sie dann über einen Service einbinden:

```
define service{
       service_description             MySQL
       use                             template-service
       check_command                   wrapper_check_mysql
       host_name                       db
}
```

Beachten Sie, dass Sie hier zumindest die verwendete Vorlage `template-service` sowie die Zuweisung an die Maschine `db` an Ihre Umgebung anpassen. Wenn Sie das Wrapper-Script auf diese Weise verwenden, erhalten Sie für jedes Datum der Ausgabe zunächst einen eigenen Graphen. Abbildung 8-14 zeigt Ihnen den entsprechenden Standard-Graphen für die durchschnittliche Anzahl an Abfragen.

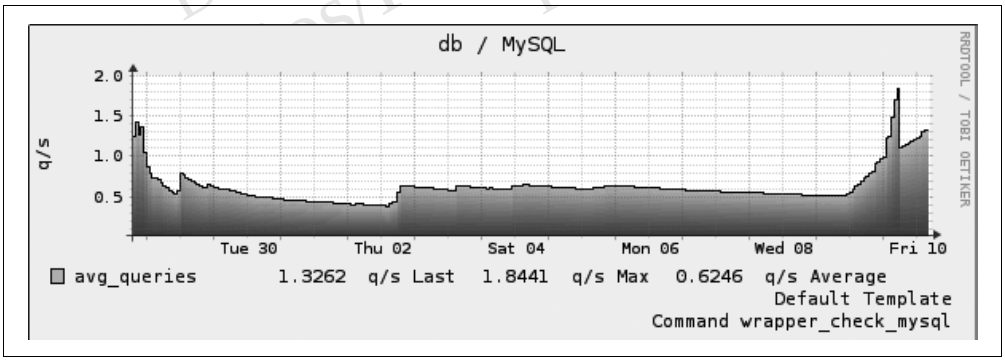


Abbildung 8-14: PNP4Nagios-Graph über wrapper_check_mysql.sh

Neben diesen die ganze Datenbank betreffenden Status können Sie das Plugin außerdem nutzen, um das Vorhandensein einer spezifischen Datenbank zu überwachen, indem Sie eine weitere Service-Definition mit dem Parameter `-d` verwenden (siehe hierzu ebenfalls Rezept 6.17, *MySQL-Datenbanken überwachen (check_mysql)*). Die Datenbank könnte dabei allerdings komplett leer sein, ohne einen Fehler zu liefern, und die ausgegebenen Daten gelten weiter für die gesamte Datenbank, so dass der Mehrwert für Sie hierdurch eher gering ist.

MySQL Datenbankabfragen durchführen

Sie können bei Nutzung von MySQL ergänzend das Standard-Plugin `check_mysql_query` verwenden (siehe Rezept 6.17, *MySQL-Datenbanken überwachen (check_mysql)*), um detaillierte Prüfungen an den von Ihnen genutzten Daten vorzunehmen. Hierzu machen wir Ihnen im Folgenden spezifische Vorschläge. Wenn Sie eine andere Datenbank verwenden, müssen Sie sich erneut zunächst nach einem möglichst äquivalenten Plugin umsehen und können dann gegebenenfalls entsprechende Abfragen durchführen.

Oracle & Postgres: Um eine Datenbank von Oracle zu überwachen, können Sie das externe Plugin `check_oracle_health` (siehe *http://labs.consol.de/lang/de/nagios/check_oracle_health/*) einsetzen. Für die Überwachung von Postgres können Sie das Plugin `check_postgres` (siehe *http://bucardo.org/wiki/Check_postgres*) verwenden.

Da detaillierte Prüfungen von der verwendeten Datenstruktur abhängen, sind diese sehr spezifisch. Zur Veranschaulichung möchten wir Ihnen hier deshalb einige Abfragen bei Nutzung des Content Management Systems (CMS) Drupal vorstellen, welches auf einer MySQL-Datenbank aufsetzt. Ein interessantes Datum bei derartigen Systemen ist beispielsweise die Anzahl aktiver Verbindungen mit dem System. Um an diese Kennzahl zu gelangen, reicht es, in diesem Fall die Anzahl der Datensätze in der Tabelle `sessions` der entsprechenden Datenbank zu zählen. Die Abfrage sieht für eine Datenbank `drupal_docu` dann so aus:

```
define service{
       service_description                    db-docu-sessions
       use                                    template-service
       check_command                          check_mysql_query_on_db!"SELECT COUNT(\*) \
                                              FROM drupal_docu.sessions"
       host_name                              db
}
```

Ein weiteres Beispiel für ein Drupal-System ist die Anzahl aller Beiträge. Hierzu lassen Sie analog die Anzahl der Einträge in der Tabelle `node` zählen. Ein entsprechend angepasster Aufruf ist folgender:

```
define service{
       service_description                    db-docu-nodes
       use                                    template-service
       check_command                          check_mysql_query_on_db!"SELECT COUNT(\*) \
                                              FROM drupal_docu.node"
       host_name                              db
}
```

Beachten Sie bei diesen Beispielen zunächst, dass Sie hier neben der genutzten Vorlage `template-service` und der für die Zuweisung genutzten Maschine `db` auch die SQL-Abfrage an Ihre Umgebung anpassen müssen. Bei einem Drupal-System müssten Sie hier nur den Datenbanknamen anpassen, bei anderen Systemen entsprechend vermutlich auch die in der Abfrage referenzierte Tabelle. Das Plugin gibt von Hause aus keine Performancedaten aus, so dass Sie zu diesen Werten zunächst keine Graphen erhalten. Wir

haben für Sie jedoch in Rezept 6.17, *MySQL-Datenbanken überwachen (check_mysql)* einen entsprechenden Wrapper erstellt, den Sie verwenden können, um auch für diese Werte Graphen zu erstellen.

Diskussion

Neben einfachen, allgemeingültigen Prüfungen einer Datenbank (Anzahl Prozesse), können Sie bei MySQL-Datenbanken bereits mit den Standard-Plugins auch sehr weitreichende Tests durchführen. Neben der Status-Abfrage, die Sie ohne weitere Kenntnisse der Datenbank-Interna einsetzen können, haben Sie auch die Möglichkeit, gezielt Datenbankabfragen durchzuführen und damit beliebige Aspekte der verwendeten Datenstrukturen zu testen. Ein externes Plugin, mit dem Sie umfangreiche Tests an einer Datenbank durchführen können, ist das Plugin `check_mysql_health` (siehe *http://labs.consol.de/lang/de/nagios/check_mysql_health/*). Für die Datenbanken Oracle und Postgres können Sie die externen Plugins `check_oracle_health` (siehe *http://labs.consol.de/lang/de/nagios/check_oracle_health/*) beziehungsweise `check_postgres` (siehe *http://bucardo.org/wiki/Check_postgres*) einsetzen.

Siehe auch

- Rezepte zur generellen Einbindung von Unix/Linux und Windows-Maschinen: Rezept 8.1, *Überwachung von Unix-/Linux-Servern* und Rezept 8.2, *Überwachung von Windows-Maschinen*
- Rezepte zu den angesprochenen Plugins: Rezept 5.3, *Prozesse überwachen (check_procs)*, Rezept 6.17, *MySQL-Datenbanken überwachen (check_mysql)*
- Rezept zur Remote-Durchführung lokaler Plugins via SNMP: Rezept 2.5, *Plugins unter Linux über SNMP ausführen*
- Rezepte zu den angesprochenen Wrapper-Scripten: Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* und Rezept 7.3, *Erweitertes Wrapper-Script zur Zusammenfassung von Prüfungen*
- Rezepte zur Installation (auch von) IDO2DB/NDO2DB: Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*

8.9 Zentrale Netzwerk-Dienste überwachen (DNS, DHCP, LDAP, AD)

Problem

Sie möchten mit Ihrem Monitoring-System zentrale Dienste (DNS, DHCP, LDAP, Active Directory) überwachen.

Lösung

Zunächst sollten Sie auch die für zentrale Dienste zuständigen Maschinen systemspezifisch einbinden (siehe hierzu Rezept 8.1, *Überwachung von Unix-/Linux-Servern* beziehungsweise Rezept 8.2, *Überwachung von Windows-Maschinen*). Für die gezielte Überwachung der eben genannten Dienste sind in den Standard-Plugins (siehe Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen* und Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*) dann weiter entsprechende Plugins für die genutzten Protokolle enthalten. Wir zeigen Ihnen im Folgenden entsprechende Optionen für die jeweiligen Plugins auf.

Domain Name System (DNS) überwachen

Das Domain Name System (DNS) zur Auflösung von Internet-Adressen wie etwa *http://www.oreilly.de/* in ihre numerischen Versionen wie in diesem Falle `213.168.78.214` ist meist auch in internen Netzwerken im Einsatz. Ebenso wie im Internet scheint Ihr Netzwerk bei einem Ausfall aus Sicht des Nutzers nicht mehr richtig zu funktionieren. Auch wenn die Maschinen unter ihren Adressen noch erreichbar sind, kennen Benutzer typischerweise eben nur die Namen von Maschinen, aber nicht deren numerische Adressen. Entsprechend wichtig ist es dann auch, diese zu überwachen.

Setzen Sie hierzu das Plugin `check_dns` ein (siehe Rezept 6.5, *Das DNS überwachen (check_dns)*). Nutzen Sie dabei beide Prüfungsmöglichkeiten, also neben DNS auch die umgekehrte Auflösung Reverse-DNS, wie im Rezept angegeben. In folgendem Fall soll getestet werden, ob die Nameserver-Konfiguration des Rechners `hcksp0.de` korrekt funktioniert. Beachten Sie, dass Sie die Vorlage (`template-srv`) und den Host (`hcksp0.de`) entsprechend Ihrer Konfiguration anpassen müssen. Die Makros `$HOSTADDRESS$` und `$HOSTNAME$` müssen dabei auf die aufzulösende IP-Adresse beziehungsweise den voll qualifizierten Rechnernamen verweisen:

```
define service {
       service_description                    DNS+
       use                                    template-srv
       check_command                          check_dns!-a $HOSTADDRESS$
       host_name                              hcksp0.de
}
define service {
       service_description                    DNS-Reverse+
       use                                    template-srv
       check_command                          check_dns_reverse!-a $HOSTNAME$
       host_name                              hcksp0.de
}
```

Wir empfehlen Ihnen, dabei möglichst die erweiterte Version aus der Diskussion zu verwenden, die den Rückgabewert über die Option `-a` auf seine Korrektheit prüft. Legen Sie dabei zumindest für Ihre internen Maschinen den zu verwendenden DNS-Server mit der Option `-s` fest, also etwa wie folgt:

```
define service {
       service_description                    DNS-int+
       use                                    template-srv
       check_command                          check_dns!-a $HOSTADDRESS$ -s 192.168.0.1
       host_name                              hcksp0.de
}
define service {
       service_description                    DNS-int-Reverse+
       use                                    template-srv
       check_command                          check_dns_reverse!-a $HOSTNAME$.
                                              -s 192.168.0.1
       host_name                              hcksp0.de
}
```

DNS cache poisioning: Eine beliebter Angriff ist die Manipulation der Zuordnung von einem Hostnamen und der zugehörigen IP-Adresse. Wenn dies gelingt, lassen sich Nutzer auf beliebige Fremdrechner umleiten. Der Angriff erfolgt im Wesentlichen durch Änderung der DNS-Einträge auf dem DNS-Server. Bei der von uns favorisierten Variante mit den Optionen `-a` und `-s` wird bei Änderung der Zuordnung auf einem angegebenen DNS-Server für das angegebene Rechnername/IP-Addresse-Paar automatisch der Zustand `CRITITAL` ausgelöst.

Beachten Sie zunächst, dass Sie die Adresse des Servers durch die Ihres DNS-Servers ersetzen. Sie können dazu auch ein Benutzermakro verwenden und den DNS-Server dort hinterlegen oder On-Demand-Makros nutzen (siehe jeweils Rezept 4.3, *Befehle hinzufügen (command)*). Wenn Sie beispielsweise einen eigenen DNS-Server unter dem Namen `ns1` definiert haben, können Sie alternativ die folgende Dienstdefinition verwenden:

```
define service {
       service_description                    DNS-ns1+
       use                                    template-srv
       check_command                          check_dns!-a $HOSTADDRESS$ \
                                              -s $HOSTADDRESS:ns1$
       host_name                              hcksp0.de
}
define service {
       service_description                    DNS-ns1-Reverse+
       use                                    template-srv
       check_command                          check_dns_reverse!-a $HOSTNAME$ \
                                              -s $HOSTADDRESS:ns1$
       host_name                              hcksp0.de
}
```

Dieses Vorgehen hat den Vorteil, dass Sie eine Änderung der Adresse des Servers gegebenenfalls nur an einer einzigen Stelle, nämlich eben der Definition der entsprechenden Maschine, nachpflegen müssen.

Für externe Domänen im Internet sollten Sie des Weiteren auch externe Server abfragen. Achten Sie dabei aber darauf, das Prüfintervall nicht zu klein zu wählen, da Sie anderenfalls vielleicht seitens des externen DNS-Servers gesperrt werden.

Neben dem Plugin check_dns steht Ihnen alternativ auch das Plugin check_dig zur Verfügung. Das Plugin check_dig ist ebenfalls in den Standardplugins enthalten und nutzt das Unix-Kommando dig. Hier ist es zusätzlich möglich, eine Port-Nummer, den Typ des abzufragenden DNS-Eintrags oder auch beliebige andere Parameter an das Kommando dig zu übergeben.

Dynamic Host Configuration Protocol (DHCP) überwachen

Das Dynamic Host Configuration Protocol (DHCP) ist für die automatische Vergabe von Netzwerkparametern zuständig. Über diese erfragen sich Maschinen im Netzwerk gegebenenfalls eine Adresse, die Netzmaske, das Gateway sowie die zu verwendenden DNS-Server. Zur Prüfung der ordnungsgemäßen Funktionsweise können Sie das Plugin check_dhcp (siehe Rezept 6.6, *DHCP überwachen (check_dhcp)*) verwenden. Beachten Sie dabei, dass dieses Plugin mit root-Rechten ausgeführt werden muss (siehe Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten*).

Verwenden Sie zunächst die Service-Definition aus dem Rezept zu DHCP ohne weitere Parameter. Wir verwenden hier eine Vorlage template-srv-generic und führen das Plugin check_dhcp auf dem Monitoring-Server selber aus (localhost). Denken Sie daran, dass Sie auch hier Ihrer Umgebung entsprechend anpassen müssen.

```
define service{
       service_description                    DHCP
       use                                    template-srv-generic
       check_command                          check_dhcp
       host_name                              localhost
}
```

Damit prüfen Sie allerdings nur, dass überhaupt ein DHCP-Server antwortet. Um gezielt zu prüfen, dass Ihr eigener DHCP-Server funktioniert, geben Sie diesen mit der Option -s an:

```
define service{
       service_description                    DHCP-own
       use                                    template-srv-generic
       check_command                          check_dhcp!-s 10.85.58.1
       host_name                              localhost
}
```

An dieser Stelle können Sie auch wieder On-Demand-Makros verwenden. Wenn Sie Ihren DHCP-Server unter dem Namen dhcp als Maschine definiert haben, können Sie hier alternativ die folgende Definition verwenden:

```
define service{
       service_description                    DHCP-own
       use                                    template-srv-generic
       check_command                          check_dhcp!-s $HOSTADDRESS:dhcp$
       host_name                              localhost
}
```

Da DHCP-Probleme häufig dadurch entstehen, dass ein anderer als der gewollte Server antwortet, sollten Sie möglichst eine feste Adresse hinterlegen und diese dann abfragen. Dadurch können Sie testen, ob ein DHCP-Server auf einem fehlkonfigurierten Gerät »übernommen« hat. Dabei übergeben Sie dann zusätzlich die zu verwendende MAC-Adresse sowie die erwartete Rückgabe:

```
define service{
       service_description                    DHCP+
       use                                    template-srv-generic
       check_command                          check_dhcp!-m 00:16:36:fb:7a:f6 \
                                              -r 10.85.58.207
       host_name                              localhost
}
```

In diesem Beispiel ist auf dem Server für die MAC-Adresse `00:16:36:fb:7a:f6` die statische IP-Adresse `10.85.58.207` hinterlegt. Beachten Sie, dass hier eben nicht mehr gezielt der eigene DHCP-Server abgefragt wird, sondern eben der tatsächlich genutzte. Da ein fremder DHCP-Server diese hinterlegte Zuweisung normalerweise nicht hätte, würden Sie auf diese Weise erkennen, wenn ein anderer als der gewünschte verwendet wird.

Lightweight Directory Access Protocol (LDAP) überwachen

Der Verzeichnisdienst Lightweight Directory Access Protocol (LDAP) wird häufig zentral für wichtige Aufgaben wie etwa die Authentifizierung eingesetzt, eben auch bei Active Directory (siehe unten). Wenn Sie diesen Dienst (LDAP beziehungsweise OpenLDAP) einsetzen, können Sie neben dem in Rezept 6.15, *Verzeichnisdienst LDAP überwachen (check_ldap)* beschriebenen einfachen Test weitere Überprüfungen einsetzen. Typischerweise möchten Sie die Authentifizierung testen, was mit dem Plugin `check_ldap` problemlos möglich ist, zum Beispiel so:

```
icinga@moni:~$ /usr/local/icinga/libexec/check_ldap -3 -H 10.85.58.44
-b "ou=people,dc=hcksp0,dc=de" -D "uid=darum,ou=People,dc=hcksp0,dc=de" -P "PASSPHRASE"
LDAP OK - 0.005 seconds response time|time=0.005360s;;;0.000000
```

Wenn Sie mehrere Prüfungen mit der entsprechenden Passphrase durchführen möchten, sollten Sie diese als Benutzermakro hinterlegen. Legen Sie hierzu eine entsprechende Variable in der Datei `resource.cfg` (bei unserer Referenzinstallation unter `/usr/local/icinga/etc/resource.cfg`) an:

```
[...]
# LDAP-Passphrase
$USER14$="PASSPHRASE"
[...]
```

Jetzt können Sie diese Passphrase direkt in der Kommandodefinition referenzieren:

```
# check_ldap v3 login
define command {
       command_name                           check_ldap3login
       command_line                           $USER1$/check_ldap -3 -H $HOSTADDRESS$ \
                                              -b $ARG1$ -D $ARG2$ -P $USER14$ $ARG3$
}
```

Auf dieser Definition aufbauend können Sie dann Services definieren, um beispielsweise den vorherigen Test zu konfigurieren:

```
define service{
       service_description                   LDAPv3login
       use                                   template-srv
       check_command                         check_ldap3login!\
ou=People,dc=hcksp0,dc=de!uid=darum,ou=People,dc=hcksp0,dc=de
       host_name                             eldub
}
```

Sie können diese Abfrage noch erweitern und beispielsweise Schwellenwerte für die Status `WARNING` und `CRITICAL` hinterlegen. Diese beziehen sich dann auf die für die Abfrage benötigte Zeit. Entsprechend erhalten Sie dann bereits eine Warnung, wenn sich Probleme anbahnen und die Authentifizierung zwar noch funktioniert, aber immer langsamer wird.

Active Directory (AD) überwachen

Das Dienste-Paket Active Directory (AD) beinhaltet zunächst Implementierungen der beschriebenen Dienste DNS und LDAP. Prüfen Sie diese wie oben beschrieben. Darüber hinaus enthält AD noch Dateidienste über CIFS/SMB (Common Internet File System/Server Message Block) sowie die Authentifizierung über Kerberos. Diese Dienste sollten Sie zusätzlich prüfen, wenn Sie eine zentrale Rolle in Ihrem Netzwerk spielen.

Für die Überprüfung von DNS und LDAP können Sie die in diesem Rezept beschriebenen Plugins check_dns und check_ldap verwenden. Für CIFS/SMB können Sie das Plugin `check_smb` (siehe Rezept 6.16, *SMB-Dateidienst überwachen (check_smb)*) einsetzen, so wie wir es auch im Rezept 8.5, *Dateiserver überwachen* beschrieben haben. Damit haben Sie die wichtigen Bestandteile von AD eingebunden und führen somit insgesamt einen Test hinsichtlich der betreffenden Funktionalität durch.

> **Kerberos**: Wenn Sie in Ihrem Netzwerk das Protokoll Kerberos für die Authentifizierung verwenden und dessen Funktionalität ebenfalls überwachen wollen, benötigen Sie zusätzlich zu den hier beschriebenen Prüfungen ein entsprechendes externes Plugin.

Diskussion

Die hier beschriebenen Dienste sind typischerweise von zentraler Bedeutung für Ihr Netzwerk. Sollte die Namensauflösung eines zugewiesenen DNS-Server nicht korrekt funktionieren, werden wohl die meisten Benutzer ein Problem haben. Ohne DHCP erhalten Maschinen gegebenenfalls keine korrekten Netzwerkparameter und werden somit erst gar nicht im Netzwerk eingebunden.

Sofern die Dienste DHCP und DNS korrekt funktionieren, dann spielt in vielen Fällen LDAP eine besondere Rolle. Außer zur Authentifizierung wird das Protokoll auch für andere Dienste, wie etwa zentrale Adressbücher eingesetzt. Neben der Dienste-Samm-

lung Active Directory bauen auch weitere Umgebungen auf LDAP als zentralen Verzeichnisdienst zur Authentifizierung auf. Entsprechend wichtig ist die Einbindung entsprechender Prüfungen in Ihr Netzwerk-Monitoring.

Siehe auch

- Rezept zu Befehlsdefinitionen und Makros: Rezept 4.3, *Befehle hinzufügen (command)*
- Rezepte zur generellen Überwachung von Unix/Linux und Windows-Maschinen: Rezept 8.1, *Überwachung von Unix-/Linux-Servern* und Rezept 8.2, *Überwachung von Windows-Maschinen*
- Kapitel zu den Nagiosplugins: Kapitel 5, *Lokale Parameter mit Standard-Plugins überwachen* und Kapitel 6, *Netzwerk-Dienste mit Standard-Plugins überwachen*
- Rezepte zu den angesprochenen Plugins: Rezept 6.5, *Das DNS überwachen (check_dns)*, Rezept 6.6, *DHCP überwachen (check_dhcp)*, Rezept 6.16, *SMB-Dateidienst überwachen (check_smb)* und Rezept 6.15, *Verzeichnisdienst LDAP überwachen (check_ldap)*
- Rezept zur Ausführung von Plugins mit administrativen Rechten: Rezept 2.2, *Plugins zur Ausführung mit administrativen Rechten vorbereiten*
- Rezept zur Überwachung eines Dateiservers: Rezept 8.5, *Dateiserver überwachen*

KAPITEL 9

Datenvisualisierung mit PNP4Nagios und NagVis

Im vorangegangenen Kapitel haben wir für Sie noch einmal zusammenfassend dargestellt, wie Sie verschiedene Arten von Servern einbinden. Für die grundlegende Überwachung steht Ihnen damit das nötige Wissen zu Verfügung. Wie Sie die Ergebnisse in den mitgelieferten Oberflächen (Standard und Icinga-Web) abrufen können, haben wir Ihnen in Kapitel 3 gezeigt. Zum Abschluss des Buches möchten wir nun noch die Verwendung von PNP4Nagios und NagVis erläutern.

Zunächst zeigen wir Ihnen die Bedienung der Oberfläche von PNP4Nagios (siehe Rezept 9.1, *PNP4Nagios verwenden*). Die von PNP4Nagios verwendeten und erzeugten Daten liegen in eigenen Dateien vor. Diese werden zwar nach Bedarf automatisch erzeugt, allerdings nicht automatisch gepflegt und entfernt. Deshalb stellen wir Ihnen zunächst ein Rezept zur Verfügung, das Ihnen zeigt, wie Sie bei Umbenennungen von Maschinen oder Diensten die Daten erhalten können (siehe Rezept 9.2, *Migration von Performancedaten bei Umbenennungen*). Des Weiteren werde wir erläutern, wie Sie die entsprechenden Dateien entfernen, wenn Sie eine Maschine oder einen Dienst außer Betrieb genommen haben (siehe Rezept 9.3, *Alte Graphen und Daten aus PNP4Nagios entfernen*). Damit Sie PNP4Nagios optimal an Ihre Bedürfnisse anpassen können, zeigen wir Ihnen dann außerdem zwei besondere Mechanismen dieses Werkzeugs: die Vereinigung von Graphen über die sogenannten »Special Templates« (siehe Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*) und die Aggregation unterschiedlicher Graphen auf einer Seite, den sogenannten »Pages« (siehe Rezept 9.5, *Benutzerdefinierte Zusammenfassung von PNP4Nagios-Graphen*). Damit sind Sie dann gerüstet, um die generierten Graphen zu verwenden und nach Bedarf Zusammenstellungen dieser zu erstellen.

Während PNP4Nagios auf den Performancedaten basiert und Graphen aus dienen generiert, können Sie mit NagVis alle Status von Maschinen und Diensten quasi nach Bedarf darstellen. Wir zeigen Ihnen zunächst die grundlegende Bedienung der Anwendung und das Erstellen von Karten (siehe Rezept 9.6, *Verwendung von NagVis und Erstellung von Karten*). Dabei platzieren Sie zunächst nach Belieben Symbole, die je nach Status des repräsentierten Objektes (Maschine, Dienst oder eine Gruppe von diesen) ihr Aussehen ändern. Darüber hinaus bietet Ihnen NagVis aber auch noch erweiterte Darstellungsmöglichkeiten: Wetterlinien und die sogenannten Gadgets (siehe hierzu Rezept 9.7, *Erweiterte Karten in*

NagVis (Wetterlinien und Gadgets)). Diese verändern ihr Aussehen nicht nur in Abhängigkeit vom Status, sondern stellen die aktuellen Werte dar. Im Falle der Wetterlinien durch farbliche Einfärbung, im Falle der Gadgets über ganz unterschiedliche Darstellungsmöglichkeiten, wie beispielsweise eine Tacho-Anzeige. Anzeigen, die Sie auf diese Weise erstellt haben, eignen sich insbesondere auch für die Darstellung etwa auf öffentlich aufgestellten Monitoren. Abschließend zeigen wir Ihnen daher auch, wie Sie dabei über die sogenannten Rotationen automatisch mehrere Karten abwechselnd darstellen können (siehe Rezept 9.8, *Karten in NagVis zu Rotationspools zusammenfassen*).

Nach dieser weiterführenden Nutzung Ihres Monitoring-Systems schließen wir das Buch mit einem Ausblick auf das angekündigte Icinga2, das viel Neues bringen wird (siehe Rezept 9.9, *Ausblick auf das kommende Icinga2*). Wir erläutern Ihnen, wie die Entwickler mit alten Beschränkungen brechen wollen, ohne die Kompatibilität aufzuheben.

9.1 PNP4Nagios verwenden

Problem

Sie haben PNP4Nagios installiert und möchten dessen Nutzung und Konfigurationsmöglichkeiten verstehen.

Lösung

Wir führen Sie im Folgenden zunächst in die allgemeine Verwendung von PNP4Nagios ein. Wir gehen dabei von einer Installation und Konfiguration wie in Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten* aus. Neben der Verwendung der PNP4Nagios-Oberfläche erläutern wir entsprechend den Zugriff über die Integration mit den Weboberflächen von Nagios beziehungsweise Icinga-Classic sowie Icinga-Web. In den folgenden Rezepten zeigen wir Ihnen dann Konfigurations- und Anpassungsmöglichkeiten auf.

Die Erzeugung von Graphen

Damit Graphen überhaupt erzeugt werden, müssen die von den Plugins zurückgegebenen Daten entsprechend ausgewertet werden. Für Graphen wird dabei zunächst davon ausgegangen, dass nur die sogenannten Performancedaten in Betracht kommen. Dies sind erweiterte Rückgaben der Plugins (siehe hierzu Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*), die allerdings nicht von allen Plugins zurückgeliefert werden.

Performancedaten nachrüsten: Wir beschreiben in Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*, wie Sie zu einem Plugin, das keine Performancedaten liefert, dennoch Performancedaten aus den eigentlichen Rückgabewerten erzeugen können.

Wenn Sie Ihr System gerade erst frisch installiert haben, sollte bereits automatisch die lokale Maschine (`localhost`) überwacht und dabei auch das Plugin `check_ping` (siehe Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*) eingesetzt werden. Dieses gibt Performancedaten zurück, weshalb wir im Folgenden zunächst von diesem Plugin ausgehen.

Wie sich die Umwandlung von Performancedaten in Graphen genau vollzieht, wird in PNP4Nagios über Vorlagen, die sogenannten `templates`, festgelegt. Mit der Installation wurde bereits eine Auswahl an Standard-Vorlagen installiert, so dass Sie in der Regel automatisch einen Graphen erhalten, wenn ein Plugin Performancedaten zurückgibt. Für das angesprochene `check_ping` enthält PNP4Nagios dann auch bereits eine entsprechend angepasste Vorlage. Wir gehen zunächst von der Nutzung dieses Dienstes mit der entsprechenden Standard-Vorlage aus und beschreiben Ihnen zunächst allgemein, wie Sie auf entsprechende Graphen zugreifen können.

Die PNP4Nagios-Oberfläche

PNP4Nagios bietet Ihnen zunächst eine eigene Weboberfläche für den Zugriff auf die generierten Graphen. Sie erreichen diese bei einer Standard-Installation über den URL */pnp4nagios* auf Ihrer Monitoring-Maschine. Beim Aufruf dieses URLs wird Ihnen PNP4Nagios automatisch die erste verfügbare Maschine mit den dazu vorhandenen Graphen anzeigen. Abbildung 9-1 zeigt Ihnen diese Einstiegsseite.

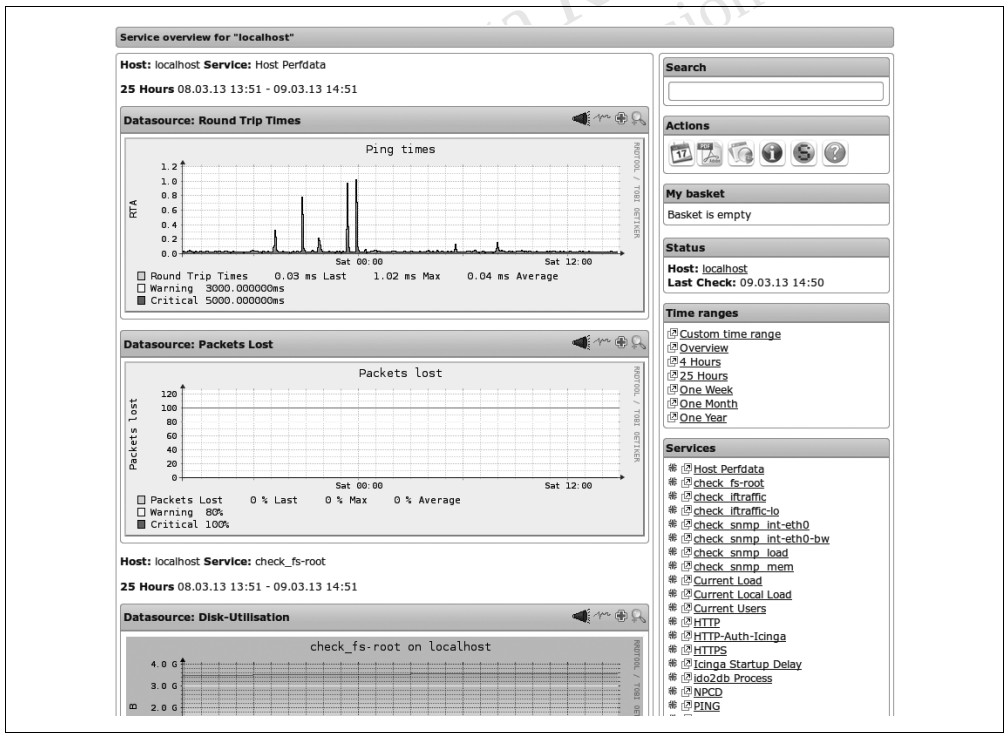


Abbildung 9-1: Die Weboberfläche von PNP4Nagios

Allerdings zeigen wir Ihnen hier eine Ansicht von einer Installation, in der bereits längere Zeit Daten gesammelt wurden. Entsprechend können wir hier umfangreichere Graphen und auch mehr Dienste zeigen, als Sie direkt nach der Installation sehen werden.

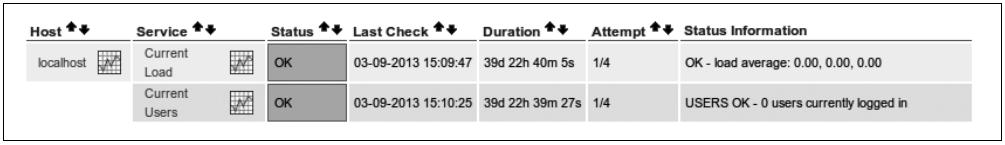
Sie können in dieser Oberfläche zu den Graphen anderen Maschinen springen, indem Sie den Namen in die Suche rechts oben eingeben (`Search`). Für die angezeigte Maschine können Sie unten rechts die verfügbaren Dienste (nur die mit Graphen) auswählen (`Services`). Über die angebotenen Zeit-Intervalle bei `Time ranges` können Sie sich den gerade angezeigten Graphen für entsprechende Zeitfenster anzeigen lassen. Über die Symbole unter `Actions` können Sie weiter zwischen den folgenden Optionen wählen (die zur Verfügung stehende Auswahl hängt von der aktuellen Anzeige und der genutzten Konfiguration ab):

- Das blaue `i` zeigt Ihnen eine Informationsseite zu den internen Daten von PNP4Nagios nebst entsprechenden Graphen.
- Das gelbe `?` ruft die (lokale, zu Ihrer Version gehörende) Hilfe zu PNP4Nagios auf.
- Die mit einem Kalenderblatt hinterlegte `17` blendet zu 2Graphen ein Fenster ein, in dem Sie ein eigenes Zeitfenster für die Anzeige angeben können.
- Das PDF-Symbol ermöglicht Ihnen, die aktuelle Ansicht als Dokument im Portable Document Format (PDF) abzurufen.
- Das Symbol mit den stilisierten Graphen ermöglicht Ihnen den Zugriff auf benutzerdefinierte Seiten (Pages, siehe Rezept 9.5, *Benutzerdefinierte Zusammenfassung von PNP4Nagios-Graphen*). Dieses Symbol steht Ihnen nur zur Verfügung, wenn Sie bereits entsprechende Seiten definiert haben.
- Das Symbol mit dem `S` erlaubt Ihnen den Zugriff auf zusammengefasste Graphen (Special Templates, siehe Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*). Auch diese Option steht Ihnen nur zur Verfügung, wenn Sie bereits entsprechende Seiten definiert haben.
- Wenn Sie die im Voranstehenden beschriebene Informationsseite zu den internen Daten von PNP4Nagios betrachten, steht Ihnen zusätzlich ein Symbol `XML` zur Verfügung, mit dem Sie die hinterlegte XML-Datei direkt einsehen können.

Neben diesem Zugriff über die PNP4Nagios-Oberfläche können Sie gegebenenfalls auch über die Integration mit den verschiedenen Weboberflächen des Monitoring-Systems auf Graphen zugreifen. Bevor wir weiter auf die Konfiguration von PNP4Nagios eingehen, stellen wir Ihnen diese Möglichkeiten kurz dar.

Integration mit Nagios und Icinga-Classic

In der Oberfläche von Nagios (siehe Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*) erzeugt die Integration ein Graph-Symbol. Abbildung 9-2 zeigt Ihnen einen Ausschnitt der Detailansicht für Dienste.

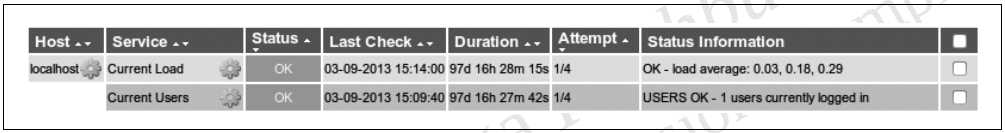


Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Current Load	OK	03-09-2013 15:09:47	39d 22h 40m 5s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	03-09-2013 15:10:25	39d 22h 39m 27s	1/4	USERS OK - 0 users currently logged in

Abbildung 9-2: PNP4Nagios-Integration mit der Weboberfläche von Nagios

Hier sehen Sie gleich drei dieser Symbole, einmal für den auf der Maschine ausgeführten Host-Check (siehe Rezept 4.1, *Maschinen einbinden (host)*) sowie jeweils eines zu den beiden gezeigten Diensten. Wenn Sie mit der Maus über einem dieser Symbole verweilen (ein sogenannter Mouse-Over), wird Ihnen der Graph in einem kleinem Fenster angezeigt. Wenn Sie auf eines der Symbole klicken, öffnet sich hingegen ein neues Browserfenster mit der PNP4Nagios-Oberfläche für den entsprechenden Graphen.

Bei der Oberfläche von Icinga-Classic erfolgt die Integration analog zu der in der Nagios-Oberfläche. Anstelle der Graph-Symbole wird Ihnen hier standardmäßig allerdings ein Rädchen angezeigt. Abbildung 9-3 zeigt Ihnen einen entsprechenden Ausschnitt, wobei Ihnen die gleichen Dienste wie voranstehend für Nagios angezeigt werden.



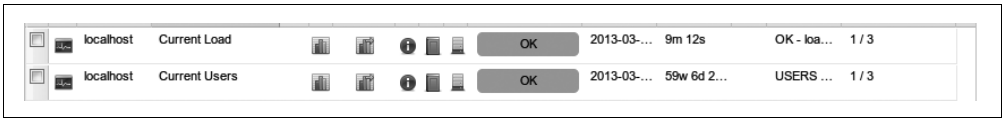
Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Current Load	OK	03-09-2013 15:14:00	97d 16h 28m 15s	1/4	OK - load average: 0.03, 0.18, 0.29
	Current Users	OK	03-09-2013 15:09:40	97d 16h 27m 42s	1/4	USERS OK - 1 users currently logged in

Abbildung 9-3: PNP4Nagios-Integration in Icinga-Classic

Auch hier können Sie sich den Graphen in einer kleinen Version ansehen, indem Sie mit der Maus über dem Symbol verweilen. Durch einen Klick öffnet sich die PNP4Nagios-Oberfläche hier standardmäßig im gleichen Fenster.

Integration mit Icinga-Web

In Icinga-Web hat sich die Integration zwischen Version 1.6.x und 1.8.x geändert. Bei den älteren Versionen wurden Ihnen analog zur eben gezeigten Integration in den Standard-Oberflächen Symbole zu den Maschinen beziehungsweise Diensten angezeigt. Abbildung 9-4 zeigt Ihnen dieses für die bereits eben angezeigten Dienste.



localhost	Current Load				OK	2013-03-...	9m 12s	OK - loa...	1 / 3
localhost	Current Users				OK	2013-03-...	59w 6d 2...	USERS ...	1 / 3

Abbildung 9-4: PNP4Nagios-Integration mit Icinga-Web (alt)

Wie Sie sehen, gab es hierbei gleich zwei Graph-Symbole. Das linke zeigt Ihnen nach einem Klick die kleine Version des Graphen im aktuellen Fenster, während Ihnen das rechte die entsprechende PNP4Nagios-Ansicht in einem neuen Tabulator innerhalb von Icinga-Web öffnet.

In den neueren Versionen wurden diese direkt verfügbaren Symbole abgeschafft und in die Details der Maschine beziehungsweise des Dienstes verschoben, die Sie zunächst mit dem Pfeilsymbol auf der linken Seite der Maschine beziehungsweise des Dienstes öffnen müssen (siehe auch Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*). Abbildung 9-5 zeigt Ihnen die dann verfügbaren Graph-Symbole für den Dienst »Current Load« nach dem entsprechenden Klick auf das Pfeilsymbol.

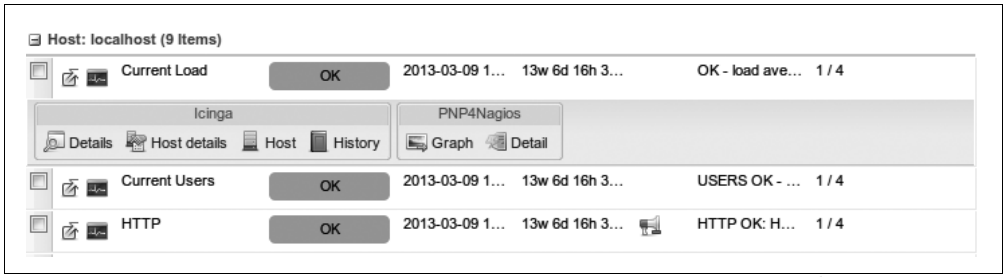


Abbildung 9-5: PNP4Nagios-Integration mit Icinga-Web (neu)

In der durch das Pfeil-Symbol geöffneten Detail-Leiste haben Sie nun Zugriff auf diverse Details. Die beiden Schaltflächen in der Kategorie PNP4Nagios erfüllen die Funktionalität der in der alten Version direkt verfügbaren Schaltflächen: Durch einen Klick auf `Graph` öffnen Sie den Graphen in einer kleinen Version und durch Klick auf `Detail` öffnen Sie die entsprechende PNP4Nagios-Seite in einem neuen Tabulator innerhalb von Icinga-Web.

Templates – Benutzerdefinierte Vorlagen für Graphen

Wie PNP4Nagios aus den Performancedaten einen Graphen generiert, wird wie bereits erwähnt über Vorlagen, die sogenannten Templates, festgelegt. Abhängig von der Art und Weise, wie Sie PNP4Nagios installiert haben, kann der Ablageort für diese Vorlagen variieren. Tabelle 9-1 zeigt Ihnen die im Rahmen der in diesem Buch beschriebenen Installationsarten aufgetretenen Pfade auf. Dabei gibt es neben `templates.dist`, in dem sich die mitgelieferten Vorlagen befinden, noch die Verzeichnisse `templates` für eigene Templates und `templates.special` mit einigen speziellen Beispielen.

Tabelle 9-1: Installationsverzeichnis der PNP4Nagios-Vorlagen

Distribution	Pfad
Installation auf Quellen	/usr/local/pnp4nagios/share/templates.dist
Debian 7.0 / Ubuntu 12.04	/usr/share/pnp4nagios/html/templates.dist
OpenSUSE 12.2 + server:monitoring + server:php:applications Repositories	/usr/share/pnp4nagios/templates.dist
CentOS 6.3 (Nagios Installation) + EPEL Repository	/usr/share/nagios/html/pnp4nagios/templates.dist

Bei der von uns verwendeten Installation aus Quellen wurden die mitgelieferten Vorlagen im Verzeichnis */usr/local/pnp4nagios/share/templates.dist* abgelegt. Die Vorlagen bestehen dabei aus Quellcode in der Sprache PHP. Dies erlaubt eine hohe Flexibilität, da es sich hier faktisch jeweils um Programme handelt, die die Daten aufbereiten und den Aufruf von RRDTool zusammensetzen. Allerdings führt dies auch zu einer hohen Komplexität, weshalb wir in diesem Buch nicht weiter auf benutzerdefinierte Graphen eingehen werden.

Diskussion

PNP4Nagios bietet Ihnen von Hause aus bereits eine gute Möglichkeit, auf einfache Weise Graphen zu erzeugen. Darüber hinaus haben Sie vielfältige Anpassungsmöglichkeiten. Benutzerdefinierte Vorlagen ermöglichen Ihnen, für unbekannte Formate Graphen nachzurüsten und die Darstellung (Linien- und Hintergrundfarben, Raster und Weiteres mehr) anzupassen. Leider können wir hier nicht alle Aspekte detailliert behandeln. Sie sollten gegebenenfalls die vom PNP4Nagios-Projekt gepflegten Seiten zu den Vorlagen lesen (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/tpl*). In den folgenden Rezepten werden wir Ihnen wichtige Schritte bei der Arbeit mit PNP4Nagios und zumindest auch Teile dieser Anpassungsmöglichkeiten vorstellen.

Wenn Sie Maschinen und/oder Dienste umbenennen, wird PNP4Nagios die unter dem alten Namen gespeicherten Daten im Normalfall nicht mehr zuordnen können. Damit Ihnen solche vorhandenen Daten nicht verlorengehen, sollten Sie in diesen Fällen entsprechende Anpassungen in PNP4Nagios vornehmen. Die betreffenden Schritte beschreiben wir Ihnen in Rezept 9.2, *Migration von Performancedaten bei Umbenennungen*. Ansonsten entstehen Ihnen bei Umbenennungen Alt-Lasten, wie sie auch bei der Entfernung von Maschinen/Diensten auftreten. Wie Sie solche Alt-Lasten aus PNP4Nagios entfernen, zeigen wir Ihnen in Rezept 9.3, *Alte Graphen und Daten aus PNP4Nagios entfernen*. Mit den sogenannten `Pages` (siehe Rezept 9.5, *Benutzerdefinierte Zusammenfassung von PNP4Nagios-Graphen*) können Sie mehrere Graphen auf einer Seite zusammenfassen, was sich insbesondere für Graphen unterschiedlicher Daten anbietet. Gleichartige Daten können Sie mit den sogenannten `Special Templates` (siehe Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*) in einem Graphen zusammenfassen.

Siehe auch

- Installation von PNP4Nagios im Installationskapitel mit Rezepten zur Installation von Nagios/Icinga auf unterschiedlichen Systemen: Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*
- Rezept zu den Rückgabewerten von Plugins inklusive Erläuterung des Formats der Performancedaten: Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*
- Rezept zu einem Wrapper-Script, um Performancedaten nachzurüsten, wenn ein Plugin solche nicht liefert: Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*

- Rezept zur klassischen Oberfläche von Nagios/Icinga und das zur neuen Oberfläche Icinga-Web: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic* und Rezept 3.4, *Nutzung der Weboberfläche Icinga-Web*
- Rezept zur Maschineneinbindung und zur Nutzung von `host-checks`: Rezept 4.1, *Maschinen einbinden (host)*
- Rezepte zu den in den Beispielen verwendeten Plugins: Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)* und Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*
- Weiterführende Rezepte zur Arbeit mit PNP4Nagios: Rezept 9.2, *Migration von Performancedaten bei Umbenennungen*, Rezept 9.3, *Alte Graphen und Daten aus PNP4Nagios entfernen*, Rezept 9.5, *Benutzerdefinierte Zusammenfassung von PNP4Nagios-Graphen* und Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*
- Dokumentationsseiten des PNP4Nagios-Projekts zu den Vorlagen: *http://docs.pnp4nagios.org/de/pnp-0.6/tpl*

9.2 Migration von Performancedaten bei Umbenennungen

Problem

Sie möchten einen Service, zu dem mit PNP4Nagios Performancedaten aufgezeichnet und Graphen erstellt wurden, umbenennen, ohne die vorhandenen Daten (und damit Graphen) zu verlieren.

Lösung

PNP4Nagios speichert zu jedem Dienst, der Performancedaten liefert, Daten in sein Datenverzeichnis. Bei unserem Referenzsystem ist dies das Verzeichnis */usr/local/pnp4nagios/var/perfdata*; bei Ihrem System wurde hier je nach Installationsart und System unter Umständen ein abweichender Pfad genutzt (siehe Tabelle in Rezept 9.3, *Alte Graphen und Daten aus PNP4Nagios entfernen*). In diesem Verzeichnis legt PNP4Nagios für jedes Gerät einen Unterordner mit dem in Nagios/Icinga genutzten Namen des Gerätes an. In diesem Ordner erstellt PNP4Nagios dann für jeden Dienst dieses Gerätes zwei Dateien mit Performancedaten. Es handelt sich dabei um Dateien mit den Endungen `.xml` und `.rrd`. Die XML-Datei enthält dabei Metadaten, die anhand der verwendeten PNP4Nagios-Vorlage erstellt wurden, und verweist auf die RRD-Datei, die die eigentlichen Daten enthält.

Wenn Sie einen Dienst in Nagios/Icinga umbenennen, finden Sie die alten Daten in den Dateien, die nach dem ursprünglichen Dienstnamen benannt wurden. Die aktuellen Daten werden dann allerdings in die Dateien mit dem neuen Namen geschrieben. Wenn sich das Format der Daten nicht geändert hat, diese also von demselben Plugin generiert

werden uns Sie beispielsweise lediglich einen Dienst umbenannt haben, können Sie diese Altdaten übernehmen. Hierzu verschieben Sie die Datei mit den alten Daten so, dass sie als Daten für den neuen Namen behandelt werden. Sehen Sie sich zur Veranschaulichung das folgende Beispiel an.

Beispiel: Migration einer Netzwerkschnittstelle

Ein Plugin prüft wie in Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins* beschrieben die Auslastung einer Netzwerkschnittstelle `br0`. Mit Hilfe der Performancedaten wurden über Monate Graphen gezeichnet. Durch Änderungen in der Netzwerkkonfiguration wurde `br0` nun auf dem entsprechenden System durch eine Schnittstelle `virbr0` ersetzt.

Die Folge dieser Änderung ist bezüglich der Überwachung die, dass der unter dem Namen `check_iftraffic-br0` eingerichtete Dienst, der die Schnittstelle über ihren Namen `br0` abfragt, nicht mehr funktioniert. Sofern für diesen Dienst Mitteilungen konfiguriert sind, erhalten Sie dann also auch entsprechende Fehler-Meldungen. Gleichzeitig wird die neu eingerichtete Schnittstelle `virbr0` zunächst nicht überwacht.

Um diese Änderung in Nagios/Icinga nachzupflegen, könne Sie nun einfach die bestehende Servicedefinition (siehe allgemein Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)* und spezifisch Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*) modifizieren. Hierzu passen Sie den Namen und die abzufragende Schnittstelle an. In unserem Beispiel wird so aus der ursprünglichen Definition

```
define service{
       service_description                    check_iftraffic-br0
       use                                    template-srv-hcksp
       check_command                          check_snmp_iftraffic!-i br0
       host_name                              hg-hcksp
}
```

die angepasste Definition:

```
define service{
       service_description                    check_iftraffic-virbr0
       use                                    template-srv-hcksp
       check_command                          check_snmp_iftraffic!-i virbr0
       host_name                              hg-hcksp
}
```

Nach einem Neustart von Nagios/Icinga verschwinden so die Fehler wegen der nicht mehr vorhandenen Schnittstelle und stattdessen wird die neue Schnittstelle geprüft. Aber dabei tritt eben der Effekt auf, dass die bereits vorhandenen Performancedaten nicht automatisch übernommen wurden. Diese befinden sich wie beschrieben unter */usr/local/pnp4nagios/var/perfdata* und wurden im vorliegenden Fall hier abgelegt:

- */usr/local/pnp4nagios/var/perfdata/hcksp0/check_iftraffic-br0.xml* und
- */usr/local/pnp4nagios/var/perfdata/hcksp0/check_iftraffic-br0.rrd*

Mit der obigen Änderung der Servicedefinition werden von PNP4Nagios nun die folgenden, neuen Dateien erstellt:

- */usr/local/pnp4nagios/var/perfdata/hcksp0/check_iftraffic-virbr0.xml* und
- */usr/local/pnp4nagios/var/perfdata/hcksp0/check_iftraffic-virbr0.rrd*

Um die Alt-Daten zu übernehmen, sollten Sie parallel oder direkt im Anschluss die folgenden Schritte durchführen:

1. Verschieben Sie die alte RRD-Datei an die Stelle der neu erstellten Datei.
2. Löschen Sie die alte XML-Datei.

Sicherungskopien: Für den Fall, dass Ihnen bei den folgenden Schritten ein Fehler unterläuft, empfehlen wir Ihnen sich zunächst Sicherungskopien der genannten Dateien anzulegen.

Damit haben Sie die beiden veralteten Dateien entsorgt und die Altdaten unter dem neuen Namen verfügbar gemacht. Überprüfen Sie den Erfolg Ihrer Operation, indem Sie die Graphen des Services unter seinem neuen Namen aufrufen. Sofern Sie die Graphen mit den bereits bestehenden Daten angezeigt bekommen, waren die Schritte erfolgreich.

Diskussion

Sofern sich ausschließlich bezeichnende Namen (also Name des Services oder des Gerätes) ändern, können Sie Alt-Daten vergleichsweise unkompliziert übernehmen. Mittels der von PNP4Nagios vorgenommenen Trennung in Daten-Datei (Endung `rrd`) und Metadaten-Datei (Endung `xml`) können Sie dies durch die beschriebenen Schritte erreichen. Alternativ könnten Sie den Verweis auf die Datendatei in der XML-Datei ändern. Wir raten Ihnen jedoch zunächst davon ab, da Sie in diesem Fall die übersichtliche Ablage mit identischen Namen für die beiden Dateien aufgäben.

Sie müssen jedoch beachten, dass eine solche Datenübernahme nicht mehr funktioniert, wenn Sie außer dem Namen auch irgendwie die Daten verändern. Selbst wenn zwei unterschiedliche Plugins das gleiche Datum messen, so sind Ihre Ausgaben häufig unterschiedlich benannt. Verwendet etwa bei Bandbreitenmessungen ein Plugin »inUsage« als Bezeichner und ein anderes »inBandwidth«, so funktioniert der beschriebene Mechanismus schon nicht mehr.

Wenn die Messungen und dazugehörigen Formate völlig gleich bleiben, können Sie unter Umständen Umbenennungen innerhalb der XML-Datei vornehmen, um entsprechend anderslautende Bezeichner einzupflegen. Dabei müssen Sie aber sehr genau beachten, dass es sich nicht etwa um unterschiedliche Maßeinheiten handelt oder Ähnliches, da Sie ansonsten Graphen erhalten, die falsche Daten wiedergeben. Wir gehen auf weitergehende Möglichkeiten der Modifizierung bereits gespeicherter Daten aufgrund der damit verbundenen Komplexität hier nicht weiter ein. Es gibt aber zum RRD-Format entsprechende Hilfsanwendungen, die vielfältige Änderungen an den Datenbanken ermöglichen.

Sofern Sie die entsprechenden Dateien wie in Rezept 7.4, *Einbinden des PNP4Nagios-Plugins (check_pnp_rrds)* beschrieben überwachen, wird Nagios/Icinga Ihnen zu den nicht mehr genutzten Dateien auch eine Warnung ausgegeben (Vorgabewert: nach 7 Tagen). Wenn dieses Plugin Ihnen eine Warnung ausgibt, liegen alte oder nicht kompatible Daten vor, und Sie müssen gegebenenfalls manuelle Korrekturen vornehmen.

Siehe auch

- Rezept zum angesprochenen Plugin: Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*
- Rezept zur Einbindung von Diensten: Rezept 4.2, *Dienste: Befehle mit Maschinen verknüpfen (service)*
- Rezept zur Einbindung des von PNP4Nagios zur Verfügung gestellten Plugins: Rezept 7.4, *Einbinden des PNP4Nagios-Plugins (check_pnp_rrds)*

9.3 Alte Graphen und Daten aus PNP4Nagios entfernen

Problem

Sie haben einen Dienst gelöscht und möchten nun auch den dazugehörigen Graphen und die dazu gespeicherten Daten in PNP4Nagios entfernen.

Lösung

PNP4Nagios legt für jeden Dienst mit Performancedaten automatisch eine XML-Datei mit der Beschreibung dieser Daten an. Je nachdem, wie Sie PNP4Nagios installiert haben, kann der Verzeichnispfad hierfür variieren. Tabelle 9-2 listet die im Rahmen dieses Buches behandelten Installationen und den dabei entsprechend verwendeten Pfad auf.

Tabelle 9-2: Speicherpfade von PNP4Nagios bei unterschiedlichen Installationsarten

Distribution	Pfad
Quellcode-Installation	/usr/local/pnp4nagios/var/perfdata
Debian 7.0 / Ubuntu 12.04	/var/lib/pnp4nagios/perfdata
OpenSUSE 12.2 + server:monitoring + server:php:applications Repositories	/var/lib/pnp4nagios/perfdata
CentOS 6.3 (Nagios Installation) + EPEL Repository	/var/lib/pnp4nagios

In diesem Verzeichnis legt PNP4Nagios zunächst einen Ordner für jede Maschine (und mit deren Namen) und darin zwei Dateien für jeden Dienst, der Performancedaten zurückgibt, an. Die Dateien haben dabei den Namen des jeweiligen Dienstes und die

Endungen `.xml` und `.rrd`. Die XML-Datei beschreibt in diesem Fall, welche Daten ursprünglich übergeben wurden und wie diese in die RRD-Dateien geschrieben werden.

Löschvorgang

Wenn Sie nun also die zu einem Dienst auf einer Maschine gehörenden Daten löschen möchten, entfernen Sie einfach die beiden entsprechenden Dateien. Nehmen wir einmal an, Sie hatten auf der Maschine `localhost` einen Service `TESTPING` definiert, den Sie nun gelöscht haben. Um die automatisch von PNP4Nagios angelegten Dateien wieder zu löschen, müssten Sie nun also den folgenden Befehl ausführen:

```
root@moni:~# rm /usr/local/pnp4nagios/var/perfdata/localhost/TESTPING.*
```

Dabei würden die zwei in dem Fall vorhandenen Dateien *TESTPING.xml* und *TESTPING.rrd* entfernt.

Alternativ können Sie den kompletten entsprechenden Ordner löschen, falls Sie die ganze Maschine entfernt haben und die Daten nicht mehr benötigen. Dabei entsorgen Sie dann ganz automatisch alle zu dieser Maschine gehörigen Dateien.

Diskussion

Die beschriebenen manuellen Schritte des Löschens scheinen auf den ersten Blick vielleicht automatisierbar. Tatsächlich aber tritt relativ häufig den Fall ein, dass ein zuvor definierter Dienst nicht mehr vorhanden ist, ohne dass Sie diese Daten gleich verlieren möchten. So kann PNP4Nagios etwa auch keine Umbenennung eines Dienstes erkennen und behandelt danach stattdessen zwei getrennte Dienste. Sie müssen in diesen Fällen manuell eingreifen, um diese Daten weiter verwenden zu können (siehe hierzu Rezept 9.2, *Migration von Performancedaten bei Umbenennungen*). Bei Umbenennungen ist dies durch entsprechende Umbenennungen der Dateien noch relativ einfach. Bei der Änderung von Datenformaten hingegen ist dann die manuelle Modifikation der Datenbanken nötig, auf die wir aufgrund der damit verbundenen Komplexität nicht weiter eingehen.

Sie können die XML- und RRD-Dateien mit Hilfe des bei PNP4Nagios mitgelieferten Plugins `check_pnp_rrds.pl` (siehe Rezept 7.4, *Einbinden des PNP4Nagios-Plugins (check_pnp_rrds)*) auf alte Graphen und Daten überwachen. Über entsprechende Warnschwellen können Sie sich dann erinnern lassen, wenn Sie Umbenennungen oder Löschungen einmal vergessen haben. Sie können das Plugin auf der Kommandozeile aufrufen, um die entsprechenden Dateien anzuzeigen. Der folgende Aufruf zeigt Ihnen ein Beispiel dafür, wie entsprechende Dateien angegeben werden:

```
icinga@moni:~$ /usr/local/pnp4nagios/libexec/check_pnp_rrds.pl
OK: 67 XML Files checked. 0 RRD Errors found. 0 old XML Files found | total=67
errors=0;1;10;0;67 old=0;1;10;0;67
```

Dazu muss das Plugin nicht dauerhaft eingebunden sein. Wenn Sie es eingebunden haben, ist Sie dieser manuellen Aufruf nicht erforderlich, da das Plugin die betroffenen Dateien auch an Nagios/Icinga zurückgibt. Die Liste betroffener Dateien wird allerdings

nicht in der kurzen Ausgabe angezeigt, so dass Sie sich zu dem Service im Zweifel gegebenenfalls die Details anzeigen lassen müssen. Abbildung 9-6 zeigt Ihnen, wie dies aussieht.

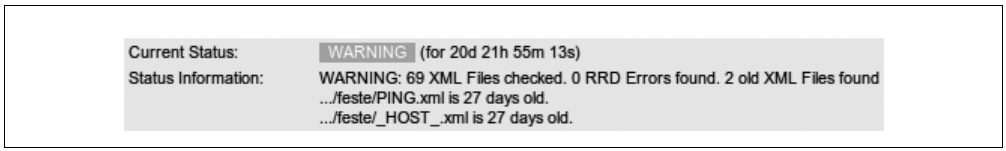


Abbildung 9-6: Detaillierte Ausgabe des Plugins check_pnp_rrd.pl

Massen-Löschung: Fall bei Ihnen möglicherweise bereits viele solcher Überreste angefallen und bisher noch nicht aufgefallen sind, können Sie dennoch einen Automatismus einsetzen. Lassen Sie dazu von dem Kommando find alle Dateien in diesem Verzeichnis suchen, die längere Zeit nicht modifiziert wurden (hier 1440 Minuten entsprechend 1 Tag):

```
root@moni:~# find /usr/local/pnp4nagios/var/perfdata/ -type f
-mmin +1440
```

Überprüfen Sie das Ergebnis. Wenn Sie möchten, können Sie die so gefundenen Dateien auch direkt löschen lassen. Seien Sie mit der folgenden Anweisung aber unbedingt vorsichtig und setzen Sie sie nur nach dem obigen Test ein, um nicht versehentlich die falschen Graphen zu löschen. Die folgende Erweiterung der Anweisung sorgt dafür, dass die gefundenen Dateien direkt gelöscht werden:

```
root@moni:~# find /usr/local/pnp4nagios/var/perfdata/ -type f
-mmin +1440 | xargs -n 1 rm
```


Siehe auch

- Rezept zur Migration von Daten bei Umbenennungen: Rezept 9.2, *Migration von Performancedaten bei Umbenennungen*
- Rezept zur Einbindung des von PNP4Nagios zur Verfügung gestellten Plugins: Rezept 7.4, *Einbinden des PNP4Nagios-Plugins (check_pnp_rrds)*

9.4 Vereinen von PNP4Nagios-Graphen über Special Templates

Problem

Sie möchten in PNP4Nagios mehrere Graphen zu einem Graphen vereinen.

Lösung

Neben der Möglichkeit, einzelne Graphen auf einer Seite zusammenzufassen (siehe hierzu Rezept 9.5, *Benutzerdefinierte Zusammenfassung von PNP4Nagios-Graphen*), kön-

nen Sie auch mehrere Graphen zu einem einzigen zusammenfassen, sie also vereinen. Dies erfolgt wieder mittels Vorlagen, den sogenannten Special Templates, die damit aus PHP-Code bestehen. Entsprechend können wir in diesem Buch nicht ausführlich auf diese eingehen. Da die mit dieser Funktion ermöglichten Effekte allerdings groß sind, möchten wir Ihnen zumindest ein Beispiel und Hinweise für eine Anpassung an die Hand geben. In den Dokumentationsseiten des PNP4Nagios-Projekts ist eine eigene Seite zu diesen Vorlagen enthalten (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/ tpl_special*). Dort finden Sie im Zweifel weiterführende Informationen.

Das folgende Beispiel ist eine Zusammenfassung aller mit Hilfe des Plugins `check_snmp_load` (siehe hierzu Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*) erzeugten CPU-Graphen von auf einem Hypervisor laufenden virtuellen Maschinen. Auf dieStellen, die Sie gegebenenfalls anpassen müssen, werden wir im Folgenden noch etwas genauer eingehen. Definitionen für entsprechende Vorlagen hinterlegen Sie im gesonderten Verzeichnis `templates`. Im Rezept 9.1, *PNP4Nagios verwenden* finden Sie eine Tabelle mit den distributionsspezifischen Installationsorten der mitgelieferten Vorlagen im Verzeichnis `templates.dist`. Die Verzeichnisse `templates.special` mit einigen speziellen Beispielen und `templates` für eigene Templates finden Sie auf der gleichen Verzeichnisebene.

Die bei unserer Installation unter */usr/local/pnp4nagios/share/templates.special* gespeicherte Vorlage `cpu.php` für die Zusammenfassung sieht folgendermaßen aus:

```
<?php

$this->MACRO['TITLE']   = "CPU-Auslastung aller virtuellen Maschinen";
$this->MACRO['COMMENT'] = "Zeigt die Auslastung der Maschinen im Vergleich.";

$services = $this-
>tplGetServices("(buckap|conf|db|docu|eldub|localhost)","check_snmp_load");

$ds_name[0] = "CPU-Auslastung";
$opt[0]     = "--title \"CPU-Auslastung\"";
$def[0]     = "";

foreach($services as $key=>$val){
    $data      = $this->tplGetData($val['host'],$val['service']);
    $hostname  = rrd::cut($data['MACRO']['HOSTNAME'], 15);
    $def[0]   .= rrd::def("var$key" , $data['DS'][0]['RRDFILE'], $data['DS'][0]['DS']);
    $def[0]   .= rrd::line1("var$key", rrd::color($key), $hostname, 1);
    $def[0]   .= rrd::gprint("var$key", array("MAX", "AVERAGE"));
}

?>
```

Die beiden Zeilen am Anfang sind Titel und Kommentar des erzeugten Graphen. Die direkt darauf folgende Zeile ist die wirklich entscheidende zur Auswahl der zusammenzufassenden Graphen. Die Funktion `tplGetServices` erwartet zwei Parameter: erstens einen regulären Ausdruck zur Beschreibung der zu berücksichtigenden Maschinen und zweitens einen regulären Ausdruck zur Auswahl der auf diesen zu berücksichtigenden

Dienste. In unserem Beispiel wählt der reguläre Ausdruck (`buckap|conf|db|docu|eldub|localhost`) eine Liste von Maschinen aus und `check_snmp_load` gibt eben an, dass von diesen die über das gleichnamige Plugin ermittelte CPU-Last berücksichtigt werden soll. Bei den beiden darauf folgenden Zeilen handelt es sich um Namen und Titel der verwendeten Werte, die in den Graphen dargestellt werden. Danach folgt eine Schleife, die bei Graphen dieser Art immer dieselbe ist. Wir gehen hier aus Platzgründen nicht auf die einzelnen Teile dieser Schleife ein. Sie können Sie einfach in der angegebenen Form verwenden.

Sobald Sie eine entsprechende Definition für Ihr System eingepflegt haben, wird Ihnen die PNP4Nagios-Oberfläche das entsprechende Symbol mit dem `S` (für Special Templates) einblenden. Dabei wird Ihnen wieder automatisch die (alphabetisch) erste Vorlage angezeigt und Sie erhalten am rechten Rand eine Liste aller verfügbaren Vorlagen. Abbildung 9-7 zeigt Ihnen diese Ansicht bei der Anzeige der eben dargestellten Definition.

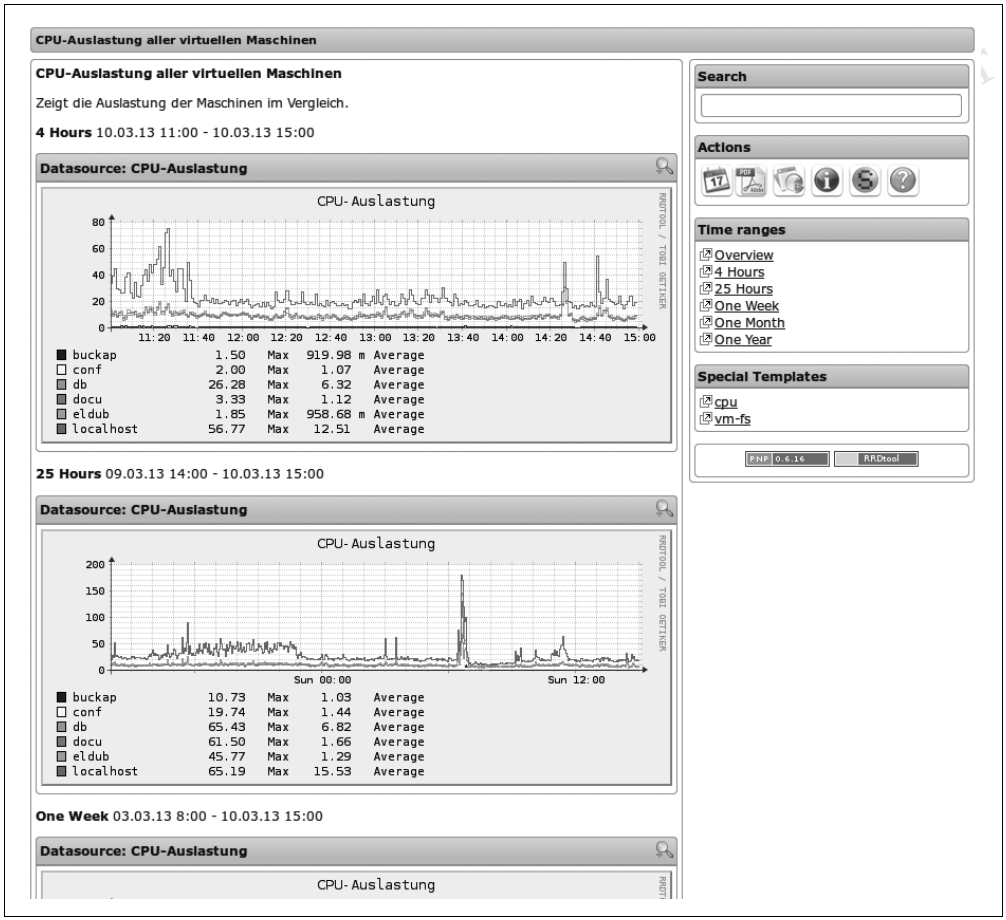


Abbildung 9-7: PNP4Nagios-Seite mit Special Templates

Diese Variante der Zusammenfassung ist sehr gut geeignet, um Graphen direkt miteinander zur vergleichen. Allerdings sollten die Daten hier die gleichen Einheiten verwenden, damit die direkte Vergleichbarkeit auch gegeben ist.

Diskussion

Die hier beschriebene Möglichkeit zur Zusammenfassung von gleichartigen Graphen ist besonders anschaulich und damit hilfreich. Neben der gezeigten Zusammenfassung der CPU-Auslastung dient vielleicht zusätzlich noch ein weiteres Beispiel zur Verdeutlichung: Die über das Plugin `check_snmp_storage.pl` (siehe hierzu Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*) ermittelte Plattenauslastung der selben virtuellen Maschinen haben wir ebenfalls zusammengefasst. Die Definition der entsprechenden Vorlage erfolgt analog zu der voranstehenden:

```
<?php

$this->MACRO['TITLE'] = "Dateisysteme virtueller Maschinen";
$this->MACRO['COMMENT'] = "Zeigt die Auslastung der Maschinen im Vergleich.";

$services = $this->tplGetServices("(buckap|conf|db|docu|eldub|localhost)","check_fs");


#throw new Kohana_exception(print_r($services,TRUE));

$ds_name[0] = "Dateisystem-Auslastung";
$opt[0] = "--title \"Dateisystem-Auslastung\"";
$def[0] = "";

foreach($services as $key=>$val){
 $data = $this->tplGetData($val['host'],$val['service']);
 $hostname = rrd::cut($data['MACRO']['HOSTNAME'], 15);
 $def[0] .= rrd::def("var$key" , $data['DS'][0]['RRDFILE'], $data['DS'][0]['DS'] );
 $def[0] .= rrd::line1("var$key", rrd::color($key), $hostname, 1);
 $def[0] .= rrd::gprint("var$key", array("MAX", "AVERAGE"));
}

?>
```

Beachten Sie, dass hier zwar die selben Maschinen berücksichtigt werden, nun aber nur der Dienst `check_fs` ausgewertet wird. Den über das Jahr akkumulierten entsprechenden Graphen sehen Sie in Abbildung 9-8.

Wenn Sie auf den Hypervisor selbst schauten, würden Sie nur ein beständig wachsendes Datenvolumen sehen. In diesem zusammengefassten Graphen aller virtuellen Maschinen können Sie hingegen sehr schön erkennen, dass dies durch das Backup-System (von uns `buckap` genannt) verursacht wird, für das wir keine automatische Löschung vorgesehen haben. Entsprechend sammeln sich die Backups immer an, bis wir diese manuell wieder entfernen, während die anderen Systeme eine vergleichsweise konstante Größe aufweisen (wobei das System `eldub` eben erst Ende letzten Jahres hinzugekommen ist). Die Möglichkeiten der Anpassung gehen dabei noch weit über die hier konkret gezeigten

hinaus, sind aber eben auch mit der Programmierung in PHP verbunden. Wenn Sie diese nicht scheuen, können Sie die Darstellung in vielerlei Hinsicht weiter anpassen.

Die hier beschriebene Form an Erkenntnissen wäre aus den einzelnen Daten eher schwer zu gewinnen, auch wenn dies bei der in diesem Fall verwendeten Anzahl von Maschinen durchaus noch möglich wäre. Aber vielleicht haben Sie ja Anwendungsfälle, in denen Sie sehr viel mehr Daten vergleichen müssen? Wir gehen jedenfalls davon aus, dass Sie diese Funktionalität zu schätzen lernen werden. Für Berichte oder E-Mails können Sie alle Seiten der PNP4Nagios-Oberfläche auch als PDF exportieren und so anderweitig verwenden. Insgesamt ist PNP4Nagios damit ein wichtiges Hilfsmittel zur Erschließung von Zusammenhängen und für ein Berichtswesen.

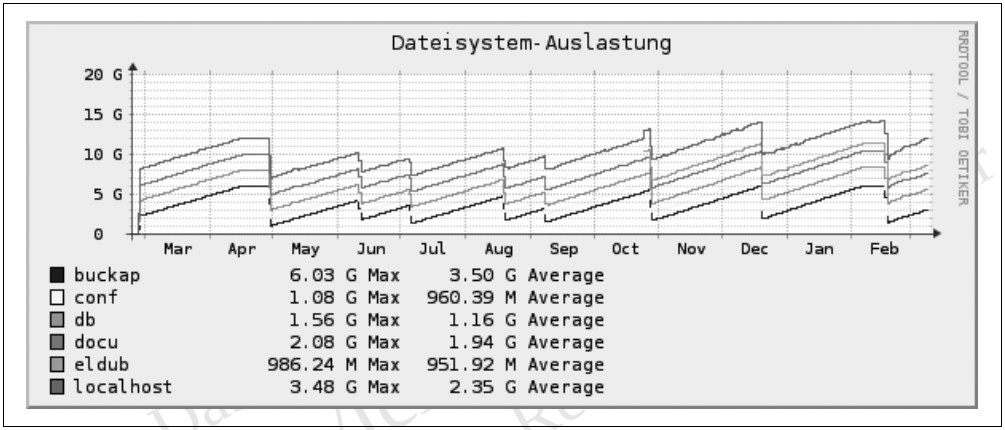


Abbildung 9-8: Zusammenfassender PNP4Nagios-Graph

Manchmal entstehen Synergien allerdings auch durch die Kombination von Graphen, die von ganz unterschiedlicher Art sind. Diese lassen sich dann nicht oder zumindest nur schlecht wie hier beschrieben zusammenfassen. Als Lösung bieten sich dann die `Pages` (im Gegensatz zu den hier beschriebenen `SpecialTemplates`) an. Diese haben wir im Rezept 9.5, *Benutzerdefinierte Zusammenfassung von PNP4Nagios-Graphen* beschrieben.

Siehe auch

- Seite des PNP4Nagios-Projekts zu den Special Templates: *http://docs.pnp4nagios.org/de/pnp-0.6/tpl_special*
- Rezept zur Anzeige unterschiedlicher Graphen auf einer Seite: Rezept 9.5, *Benutzerdefinierte Zusammenfassung von PNP4Nagios-Graphen*
- Rezept zu den angesprochenen Plugins: Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*
- Rezept zur Verwendung von PNP4Nagios: Rezept 9.1, *PNP4Nagios verwenden*

9.5 Benutzerdefinierte Zusammenfassung von PNP4Nagios-Graphen

Problem

Sie möchten Graphen in PNP4Nagios auf einer Seite darstellen, um nicht zwischen den einzelnen Ansichten wechseln zu müssen.

Lösung

Neben der Möglichkeit, die Graphen zu vereinen (siehe hierzu Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*), können Sie mit PNP4Nagios auch mehrere Einzelgraphen auf einer Seite darstellen. Diese Seiten werden Pages genannt, wohl da sie zusätzliche Seiten in PNP4Nagios erzeugen. Sie bieten sich an, wenn eine Vereinigung nicht möglich ist, weil die Graphen etwa völlig unterschiedliche Werte darstellen. Das Symbol zum Aufruf dieser Seiten (siehe Abbildung 9-9, der Knopf mit dem Kreisdiagramm) wird Ihnen erst angezeigt, wenn Sie die erste Seite angelegt haben.

Sie können solche Seiten über recht einfache Konfigurationsdateien definieren. In den Dokumentationsseiten des PNP4Nagios-Projekts findet sich hierzu eine eigene Seite (siehe *http://docs.pnp4nagios.org/de/pnp-0.6/pages*). Da es sich bei den entsprechenden Konfigurationsdateien nicht um PHP-Dateien handelt, liegen sie nicht im Vorlagen-Verzeichnis, sondern in einem eigenen Verzeichnis. Bei unserer Referenzinstallation finden Sie sie unter */usr/local/pnp4nagios/etc/pages* beziehungsweise bei einer paketbasierten Installation typischerweise unter */etc/pnp4nagios/pages*.

Die Syntax ist dabei der von Nagios/Icinga nicht unähnlich. Zur Demonstrationszwecken erstellen wir hier eine Seite, die von allen Maschinen den Graphen für den Dienst `PING` einbindet. Legen Sie in dem genannten Verzeichnis dazu zunächst eine Datei mit der Endung `.cfg` an und definieren Sie mit Anweisungen wie den folgenden zunächst die Seite an sich:

```
define page {
       use_regex                       1
       page_name                       Alle PINGs
}
```

Mit der Option `use_regex` geben Sie dabei an, ob Sie in den folgenden Definitionen für die zu nutzenden Graphen reguläre Ausdrücke verwenden möchten (`1`) oder nicht (`0`). Über die Option `page_name` geben Sie der Seite einen Namen, der dann im Auswahl-Menü für die Seiten verwendet wird.

Darauf folgend müssen Sie nun nur noch angeben, welche Graphen auf der Seite angezeigt werden sollen. Für unser Beispiel sollen dafür alle Maschinen und eben der Dienst `check_ping` (siehe hierzu Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*) berücksichtigt werden. Fügen Sie Ihrer Datei dazu den folgenden Abschnitt hinzu:

```
define graph {
       host_name                       .*
       service_desc                    check_ping
       source                          0
}
```

Da wir im Voranstehenden die Verwendung von regulären Ausdrücken gewählt haben, geben wir hier den Ausdruck `.*` für alle Maschinennamen an. Wenn Sie keine regulären Ausdrücke verwenden (also weiter vorne für `use_regex` den Wert `0` setzen), können Sie an dieser Stelle stattdessen eine durch Kommata separierte Liste von Maschinennamen angeben. Mit der Option `service_desc` definieren Sie die zu berücksichtigen Dienste, hier eben `check_ping`.

Das Plugin `check_ping` hat die Besonderheit, dass es zwei verschiedene Werte zurückgibt: Zunächst die mittlere Antwortzeit und dann die dabei verlorengegangenen Pakete. Die Option `source` ermöglicht Ihnen, bei mehreren Werten auszuwählen, welcher dieser Werte auf der Seite angezeigt werden soll. Um auf der Seite also nur die Antwortzeiten darzustellen, haben wir diese Option hier auf den Wert `0` für den ersten Wert gesetzt. Sie können ihn aber auch auf `1` setzen und damit stattdessen eben die verlorenen Pakete anzuzeigen oder die Option ganz weglassen, um beide Werte anzuzeigen.

Abbildung 9-9 zeigt Ihnen als Beispiel die entsprechend der voanstehenden Definitionen erzeugte Seite. Sie erreichen diese bei Ihrem System, indem Sie nach der Definition der Seite das Symbol mit dem abgebildeten Torten-Diagramm wählen. Bei Vorhandensein mehrere Graphen zeigt Ihnen PNP4Nagios dann zunächst die (alphabetisch) erste Seite. Über den rechts unten verfügbaren Menüpunkt werden alle definierten Seiten aufgeführt und Sie können zu der von Ihnen gewünschten Seite springen, hier (wie weiter vorne definiert) die Seite `Alle PINGs`.

Auf diese Weise können Sie sich nach Bedarf Übersichten von Graphen zusammenstellen. Wenn Ihre Maschinen viele Graphen erzeugen, wird es für Sie ohne dieses Hilfsmittel recht mühsam, einen bestimmten Graphen aller Maschinen zu vergleichen. Entsprechend sollten Sie mit dieser Funktion vertraut sein, insbesondere, da sich entsprechende Seiten wie gezeigt auf einfache Weise definieren lassen.

Diskussion

Der Mechanismus, unterschiedliche Graphen untereinander darzustellen, ergänzt die durch die Vereinigung (siehe hierzu Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*) gegebenen Möglichkeiten. Während es die Vereinigung allerdings ermöglicht, mehrere Graphen in nur einem einzigen zusammenzufassen, reiht dieser Mechanismus ausgewählte Graphen untereinander auf. In der Oberfläche von Nagios/Icinga können Sie sich zwar Gruppen von Maschinen und Diensten anzeigen lassen und die Graphen zu den jeweiligen Maschinen oder Diensten über entsprechende Symbole erreichen. Sie sind aber eben nicht gleichzeitig sichtbar. Dies ist deshalb der Mehrwert, den die gemeinsame Darstellung der Graphen zu bieten hat. Da der Aufwand

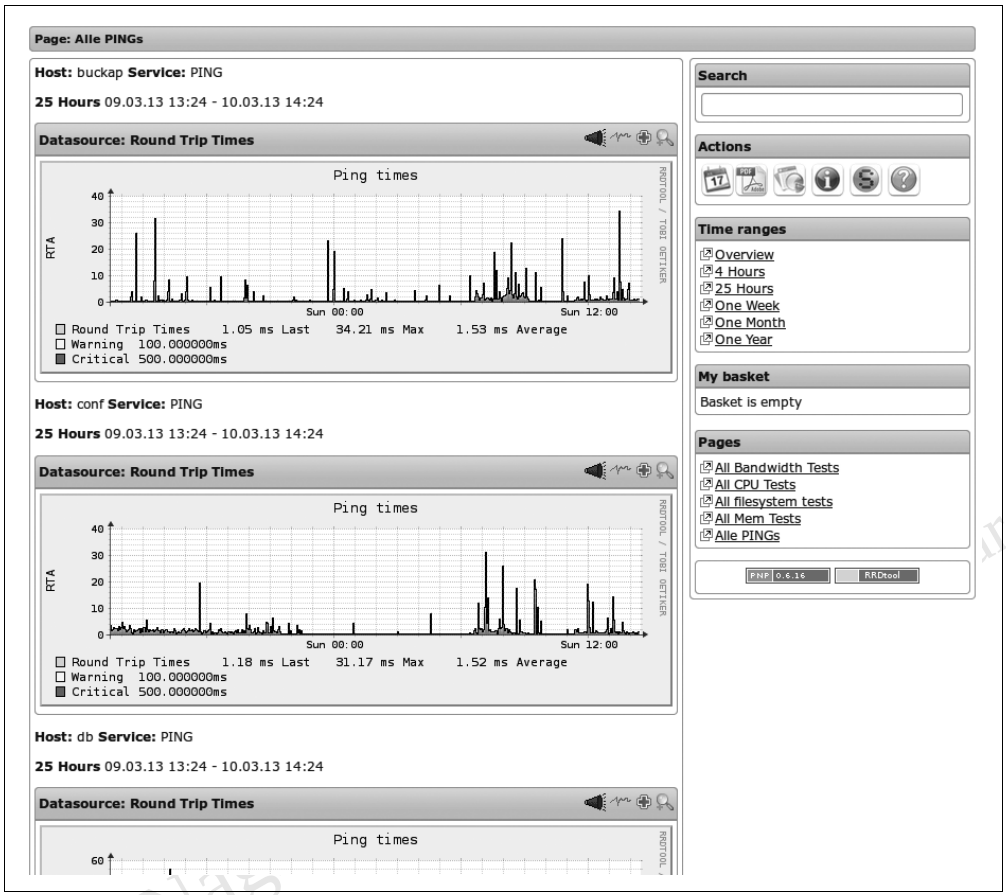


Abbildung 9-9: PNP4Nagios: Seite aller PINGs

für die Definition recht klein ist und die Daten der einzelnen Graphen ja bereits vorliegen, können Sie entsprechende Übersichten nach Bedarf erstellen und verwenden. Wenn Sie sie nicht mehr benötigen, können Sie die Dateien mit den entsprechenden Definitionen einfach löschen – die einzelnen Graphen sind dadurch nicht betroffen.

Siehe auch

- Dokumentation des PNP4Nagios-Projekts zu den Pages: *http://docs.pnp4nagios.org/de/pnp-0.6/pages*
- Rezept zur Vereinigung von Graphen: Rezept 9.4, *Vereinen von PNP4Nagios-Graphen über Special Templates*
- Rezept zum hier angesprochenen Plugin: Rezept 6.1, *Erreichbarkeit überwachen mit Ping (check_ping)*

9.6 Verwendung von NagVis und Erstellung von Karten

Problem

Sie möchten Maschinen und/oder Dienste mit NagVis nach Bedarf anordnen und sich so maßgeschneiderte Ansichten erstellen.

Lösung

NagVis ermöglicht es Ihnen Symbole für Maschinen und Dienste frei auf beliebigen Hintergründen anzuordnen. Die damit erreichbaren Visualisierungen können einen erstaunlichen Unterschied im Vergleich zu der rein technischen Darstellung in den Standard-Oberflächen von Nagios/Icinga ausmachen. Die Möglichkeiten, die NagVis Ihnen hierzu in der aktuellen Version 1.7 bietet, sind umfangreich und beinhalten unter anderem auch eine Programmierschnittstelle zur Programmierung dynamischer Karten. Aus Platzgründen können wir hier nur auf die wichtigsten Funktionen eingehen, verweisen Sie aber zumindest auf die weiterführenden Möglichkeiten.

Anmeldung und Menü

Wir gehen im Folgenden von einem installierten NagVis (siehe hierzu Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*) aus. Sie erreichen die Oberfläche dazu über den URL */nagvis* auf Ihrem Server, also etwa *http://localhost/nagvis* bei einem lokalen Zugriff. Zur erstmaligen Anmeldung verwenden Sie die vorgegebenen Zugangsdaten für das Konto `admin` mit dem Passwort `admin`. Am oberen Rand bietet NagVis Ihnen nach der Anmeldung ein Menü zur Auswahl zentraler Funktionen. Abbildung 9-10 zeigt Ihnen dieses Menü.

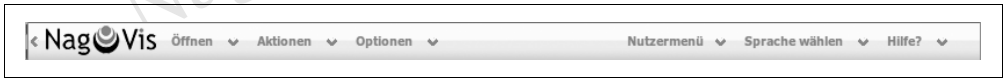


Abbildung 9-10: Das Menü von NagVis

Die Ihnen hier zur Verfügung stehenden Funktionen erläutern wir Ihnen kurz:

- `Öffnen`

  Hier bietet Ihnen NagVis die verfügbaren Karten sowie eine entsprechende Übersicht an. Mit einem Klick auf den entsprechenden Kartennamen können Sie die entsprechende Karte gezielt öffnen. Über die Auswahl `Übersicht` gelangen Sie hingegen zur Übersichtsseite, auf der Ihnen die Karten angezeigt werden.

- `Aktionen`

  In diesem Menü bietet NagVis Ihnen Aktionen passend zur jeweiligen Ansicht an. Auf der Übersichtsseite ist das Menü zunächst leer. Beim Ansehen einer Karte können Sie hier eine Suche aufrufen, mit deren Hilfe Sie die zu einer Maschine beziehungsweise einem Dienst gehörenden Symbole auf einer Karte hervorheben lassen können.

- `Karte bearbeiten`

  Dieses Menü wird Ihnen nur angezeigt, wenn Sie sich aktuell eine spezifische Karte ansehen. Wir werden im Nachfolgenden auf wichtige Punkte dieses Menüs eingehen.

- `Optionen`

  In diesem Menü können Sie die Konfigurations-Dialoge: `Hauptkonfiguration`, `Datenquellen verwalten`, `Hintergründe verwalten`, `Karten verwalten` und `Bilder verwalten` aufrufen. Wir werden im Folgenden nur in dem Rahmen auf diese Optionen eingehen, in dem sie für die von uns gezeigten Aktionen notwendig sind.

- `Nutzermenü`

  Hier zeigt Ihnen NagVis, als welcher Benutzer Sie angemeldet sind, und ermöglicht Ihnen eine Abmeldung. Je nach Berechtigung werden ihnen hier darüber hinaus die Optionen `Passwort ändern`, `Benutzer verwalten` und `Rollen verwalten` angezeigt. Wir gehen hier nicht weiter auf die Benutzerverwaltung von NagVis ein. Das Passwort für das Benutzerkonto `admin` sollten Sie aber in jedem Falle ändern. Überprüfen Sie gegebenenfalls auch das voreingerichtete Konto `guest` und passen Sie es an Ihre Anforderungen an.

- `Sprache wählen`

  Unter diesem Menüpunkt können Sie gegebenenfalls andere Sprachen für die Oberfläche auswählen.

- `Hilfe`

  Hier bietet Ihnen NagVis einen schnellen Zugriff auf die `Dokumentation` und das `Forum`. Der Eintrag `Dokumentation` wird dabei die zur Ihrer Version mitinstallierte Dokumentation anzeigen. Über `NagVis-Support-Informationen` können Sie darüber hinaus die Eckdaten Ihrer NagVis-Installation abrufen, um bei einer Support-Anfrage entsprechende Informationen zur Verfügung zu stellen.

Die Verwendung der einzelnen Funktionen für die Erstellung und Nutzung von Karten erläutern wir im Rahmen der folgenden Ausführungen und Rezepte.

Einfache Karten mit Hintergrund und Symbolen

Die einfache Art der Visualisierung erreichen Sie über die Platzierung von Symbolen auf entsprechenden Hintergründen. NagVis bietet Ihnen von Hause aus für jede Maschine und jeden Dienst eine Symboldarstellung, so dass Sie hier nur einen geeigneten Hintergrund aussuchen beziehungsweise erstellen und die passenden Symbole entsprechend platzieren müssen. Betrachten Sie als Beispiel Abbildung 9-11 .

Abbildung 9-11: NagVis-Karte für einen Campus-Überblick

Hier wurde eine Campus-Karte (das Beispiel stammt aus OpenStreetMap) als Bild für den Hintergrund erstellt und Symbole (die Kreise mit dem Häkchen) für WLAN-Stationen entsprechend der Anbringungsorte an den Gebäuden platziert. Vom Inhalt her entspricht die Darstellung etwa einer entsprechenden Maschinen-Gruppe in der klassischen Oberfläche: Die Symbole zeigen für jede Maschine den Status an. Alle weiteren verfügbaren Informationen sind dabei leicht über einen Klick zugänglich, der Sie direkt zu der entsprechenden Detail-Ansicht in der Standard-Oberfläche (siehe hierzu Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*) bringt.

Die Erstellung der in Abbildung 9-12 gezeigten Darstellung von Serverschränken erfolgt ähnlich. Hier wurden die Serverschränke zunächst als Bild erstellt. Dieses Bild wurde dann in Nagvis als Hintergrund für eine neue Karte gewählt. Auf dieser wurden dann die Geräte- und Dienst-Icons hinzugefügt.

Hintergründe erstellen

Um eine entsprechende Karte zu erstellen, benötigen Sie zunächst ein entsprechendes Bild als Hintergrund. Für eine Karte wie im gezeigten Campus-Überblick können Sie beispielsweise auf Karten des Projekts OpenStreetMap zurückgreifen.

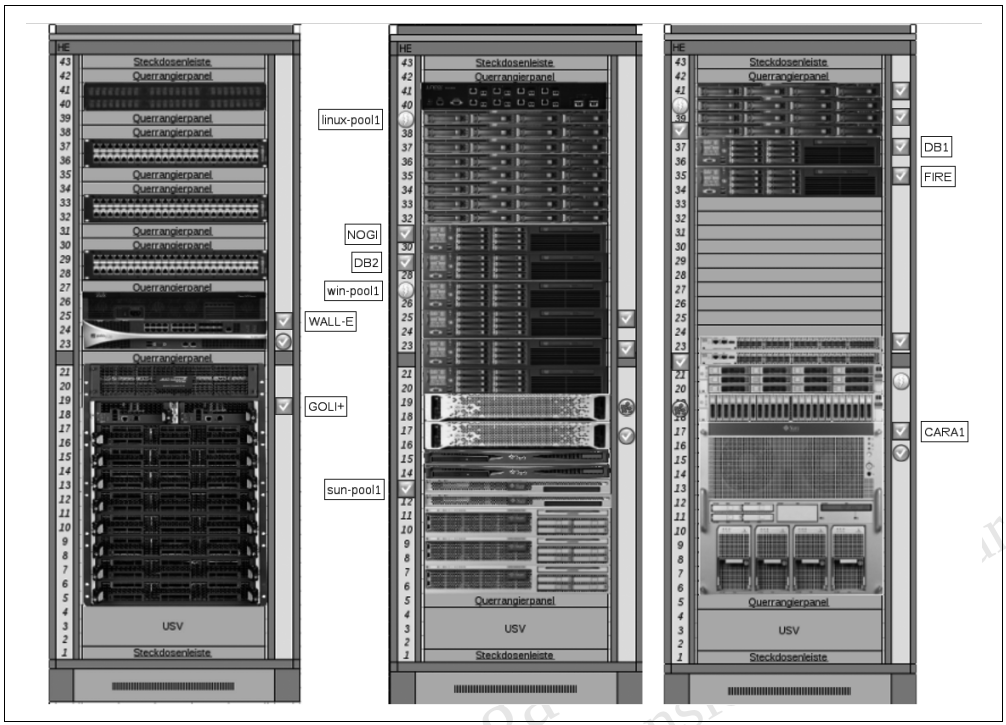


Abbildung 9-12: NagVis-Karte für einen Schrank-Überblick

NagVis Geomap: Seit Version 1.7 bietet NagVis eine integrierte Unterstützung für OpenStreetMap-Karten. Das Feature Geomap ermöglichte zunächst, die Koordinaten über eine Textdatei anzugeben. Seit Version 1.7.4 gibt es eine weiterführende Integration, die die Angabe der Koordinaten über benutzerdefinierte Variablen in der Nagios- beziehungsweise Icinga-Konfiguration vorsieht. Wir haben mit diesem Feature noch nicht gearbeitet, gehen aber davon aus, dass sich hiermit einfacher Karten erzeugen lassen, die allerdings entsprechend generisch bleiben. Wir gehen im Folgenden nur auf die manuelle Erzeugung ein. Bei Interesse finden Sie weiterführende Informationen zu diesem neuen Feature auf der entsprechenden Dokumentationsseite des Projektes (unter *http://docs.nagvis.org/1.7/de_DE/geomap.html*).

Ihr als Hintergrund zu verwendendes Bild pflegen Sie in NagVis dann zunächst über die Funktion `Optionen / Hintergründe verwalten` ein. In dem sich öffenenden Fenster bietet Ihnen NagVis drei Abschnitte. Mit dem ersten können Sie generisch Hintergründe mit einer einfachen Füllfarbe erstellen. Mit dem zweiten können Sie Bilder zur Verwendung als Hitnergrund hinterlegen. Mit dem dritten können Sie vorhandene Hintergründe löschen. Wählen Sie im zweiten Abschnitt das von Ihnen erstellte Bild aus und laden Sie es hoch.

Karten erstellen

Jetzt können Sie eine entsprechende Karte erstellen. Wählen Sie hierzu `Optionen / Karten verwalten`. In dem sich öffnenden Dialogfenster bietet Ihnen NagVis wieder verschiedene Abschnitte an. Der erste Abschnitt ermöglicht Ihnen, eine neue Karte zu erstellen. Geben Sie zunächst einen Namen für die neue Ansicht an. Bei der Auswahl `Karten-Iconset` können Sie unter den vorgegebenen Icons auswählen. Die Ihnen hier angebotene Auswahl enthält zunächst einen Satz von Symbolen in unterschiedlich großen Darstellungen (klein, mittel und groß). Wir haben in der dargestellten Karte die kleine Darstellung gewählt. Sie können die Auswahl der Symbole nachträglich ändern. Wenn Sie unsicher sind, beginnen Sie also einfach mit der mittleren Größe und schalten Sie gegebenenfalls später auf eine andere Größe um.. Schließlich wählen Sie noch den von Ihnen angelegten Hintergrund aus.

Symbole hinzufügen

Nach diesen Schritten öffnen Sie Ihre neue Karte über das Menü `Öffnen` und Auswahl des von Ihnen angegebenen Kartennamens. Sie werden dann zunächst nur den Hintergrund sehen. Beachten Sie, dass das zusätzliche Menü `Karte bearbeiten` nun angezeigt wird. Für jede Maschine beziehungsweise jeden Dienst, für den Sie auf der Karte ein Symbol platzieren möchten, führen Sie nun die folgenden Schritte durch:

1. Wählen Sie `Karte bearbeiten / Icon hinzufügen` und dann Maschine, Dienst oder eine entsprechende Gruppe aus.
2. Klicken Sie an der Stelle auf dem Hintergrund, an dem das Symbol angezeigt werden soll. Beachten Sie dabei, dass Sie mit dem Klick die Position für die linke obere Ecke des Symbols spezifizieren. Entsprechend wird das Icon also anschließend rechts darunter angezeigt.
3. In dem sich öffnenden Fenster können Sie dann entsprechend Ihrer vorherigen Auswahl die Maschine, den Dienst oder die Gruppe auswählen, die das Icon repräsentieren soll. NagVis wird Ihnen alle in Frage kommenden Einheiten in dem entsprechenden Menü anbieten.

Nach diesen Schritten wählen Sie innerhalb des Dialogs den Button `Speichern`. NagVis wird die Karte nach einigen Sekunden neu laden und den Status für das hinzugefügte Objekt mit dem entsprechenden Symbol anzeigen.

Texte, Bilder und Linien hinzufügen

Neben den besprochenen Symbolen für die Darstellung der entsprechenden Status von Maschinen, Diensten oder Gruppen, können Sie Karten auch statuslose Objekte hinzufügen. NagVis bietet Ihnen hier die Wahl zwischen Texten, Bildern und Linien. Um Bilder einzubinden, müssen Sie diese allerdings zunächst auf dem Server hinterlegen. Hierzu stellt NagVis einen entsprechenden Dialog unter `Optionen / Bilder verwalten` bereit. Im oberen Teil des darüber angebotenen Bildschirmdialogs können Sie Bilder hinterlegen und so für die Einbindung verfügbar machen. Im unteren Teil können Sie nicht mehr benötigte Bilder wieder vom Server löschen.

Um ein statusloses Objekt zu einer Karte hinzuzufügen, führen Sie folgende Schritten durch:

1. Wählen Sie `Karte bearbeiten / Spezial-Objekt hinzufügen` und dann den entsprechenden Objekt-Typ.
2. Klicken Sie an der Stelle auf dem Hintergrund, an dem Sie das Objekt platzieren möchten. Bei Linien haben Sie damit nur den Startpunkt der Linie markiert und klicken ein weiteres Mal, um den Endpunkt der Linie anzugeben.
3. In dem sich öffnenden Fenster können Sie dann entsprechend Ihrer vorherigen Auswahl weitere Optionen festlegen. Für Texte geben Sie hier den Text an, bei Bildern wählen Sie hier das Bild aus und für Linien können Sie die Darstellungsart der Linien angeben.

Anschließend wählen Sie in dem Dialog-Fenster den Button `Speichern`, um Ihre Änderungen zu sichern. NagVis wird dann das Dialog-Fenster schließen und die Karte neu laden, um Ihre Änderungen anzuzeigen.

Karten ändern

Im Ansichtsmodus, in dem Sie sich standardmäßig befinden, zeigt NagVis Ihnen Detail-Informationen an, sobald Sie die Maus über eines der Symbole bewegen. Ein Klick auf ein Symbol führt Sie auf die entsprechende Seite der Oberfläche von Nagios beziehungsweise Icinga. Wenn Sie Symbole verschieben oder ändern möchten, müssen Sie zunächst in den Bearbeitungsmodus wechseln. Wählen Sie hierzu `Karte bearbeiten / Alle Sperren/Freigeben`. Dass Sie sich jetzt im Bearbeitungsmodus befinden, zeigt Ihnen NagVis über ein rot dargestelltes `Edit Mode!` neben dem Eintrag für das Nutzermenü an.

Damit ändert sich das Verhalten von NagVis dahingehend, dass Ihnen keine Informationen mehr angezeigt werden, wenn Sie mit der Maus über ein Symbol fahren und bei einem Klick keine Weiterleitung zu Nagios beziehungsweise Icinga mehr erfolgt. Sie können im Bearbeitungsmodus stattdessen alle Symbole auf der Karte verschieben. Über einen Klick mit der rechten Maustaste auf ein Symbol zeigt Ihnen NagVis dann weitere Bearbeitungsoptionen an. In diesem Menü können Sie über `Modify object` das Dialogfenster zu den Symboleigenschaften öffnen und beispielsweise das auszuwertende Objekt (Maschine beziehungsweise Dienst) ändern.

Symbol-Beschriftungen: Wenn Sie auf Karten unterschiedliche Dienste oder Gerätetypen anzeigen, sollten Sie Ihre Symbole gegebenenfalls beschriften. Setzen Sie dazu in den Eigenschaften des Symbols die Option `labels_show` auf `Ja`. Daraufhin wird Ihnen NagVis weitere Optionen zu Inhalt und Positionierung der Beschriftung anbieten. Standardmäßig erhalten Sie zunächst die Möglichkeit, den Namen der jeweiligen Maschine beziehungsweise des jeweiligen Dienstes zu verwenden und das entsprechende Textfeld unter dem Icon zu positionieren. Sie können diese Optionen anpassen, um Inhalt und Position an Ihre Bedürfnisse anzupassen. Seien Sie sich aber darüber bewusst, dass diese Art Anpassung relativ viel Arbeit bereitet und bei Änderungen gegebenenfalls einen entsprechenden Pflegeaufwand nach sich zieht.

Wenn Sie die Eigenschaften der Karte selbst ändern möchten, etwa um einen anderen Hintergrund auszuwählen oder das verwendete Icon-Set zu ändern, rufen Sie `Karte bearbeiten / Kartenkonfiguration` auf. So erhalten Sie einen Eigenschaften-Dialog zu der jeweiligen Karte. In diesem wird Ihnen eine Vielzahl von Optionen angeboten. Um den Hintergrund zu ändern, wählen Sie bei `map_image` einen abweichenden Hintergrund aus, und um die verwendeten Icons zu ändern, wählen Sie unter `iconset` entsprechend ein anderes Set.

Diskussion

Wir haben Ihnen hier die allgemeinen Schritte zur Arbeit mit NagVis inklusive der Erstellung einfacher Karten aufgezeigt. Mit diesen vergleichsweise einfachen Schritten können Sie bereits sehr effektvolle Darstellungen erreichen. Die mit Version 1.7 neu dazugekommene Möglichkeit, Symbole anhand einer im Rahmen der Nagios- beziehungsweise Icinga-Konfigurationsdateien angegebenen Position automatisch auf einer entsprechenden Karte anzuzeigen, haben wir selbst noch nicht genutzt. Wir gehen aber davon aus, dass diese vor allem für grobe Übersichten in sich ständig ändernden Szenarien hilfreich ist. Die Möglichkeiten durch NagVis sind insgesamt sehr umfangreich, weshalb wir hier auf viele Detail-Aspekte nicht eingehen können. Wir zeigen Ihnen im Rezept 9.7, *Erweiterte Karten in NagVis (Wetterlinien und Gadgets)* jedoch weitere Möglichkeiten der Darstellung unter Nutzung der von NagVis gebotenen Wetterlinien und Widgets auf. In Rezept 9.8, *Karten in NagVis zu Rotationspools zusammenfassen* zeigen wir Ihnen weiter, wie Sie in NagVis Rotationen von Karten definieren und verwenden. Ansonsten gehen wir davon aus, dass Ihnen die gezeigten Möglichkeiten Anreiz genug bieten, um sich mit NagVis zu beschäftigen. Die weitergehenden Optionen müssen Sie sich dann gegebenenfalls nach Bedarf erschließen.

Siehe auch

- Rezepte zur Installation von Nagios/Icinga und NagVis: Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*
- Dokumentation des NagVis-Projektes für Version 1.7 unter *http://docs.nagvis.org/1.7/de_DE/index.html*
- Rezept zur klassischen Oberfläche von Nagios/Icinga: Rezept 3.3, *Nutzung der Weboberflächen von Nagios und Icinga-Classic*
- Dokumentation des NagVis-Projektes für das neue Feature Geomap unter *http://docs.nagvis.org/1.7/de_DE/geomap.html*
- Weiterführende Rezepte zur Arbeit mit NagVis: Rezept 9.7, *Erweiterte Karten in NagVis (Wetterlinien und Gadgets)* und Rezept 9.8, *Karten in NagVis zu Rotationspools zusammenfassen*

9.7 Erweiterte Karten in NagVis (Wetterlinien und Gadgets)

Problem

Sie möchten Ihre NagVis-Karten erweitern und die Mechanismen sich anpassender Wetterlinien und Gadgets kennenlernen.

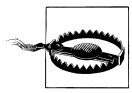
Lösung

Für die folgenden Schritte sollten Sie bereits eine Karte wie in Rezept 9.6, *Verwendung von NagVis und Erstellung von Karten* gezeigt angelegt haben. Wir zeigen Ihnen im Folgenden darauf aufbauend, wie Sie Visualisierungen mit Wetter-Linen und Widgets einbinden und was Sie dabei beachten müssen. An Beispielen führen wir Ihnen den Nutzen dieser Art von Erweiterungen vor Augen, so dass Sie überlegen können, inwiefern Ihnen diese Optionen von Vorteil sein können.

Visualisierungen mit Wetterlinien

Um den Status eines Netzwerks im Blick zu haben, bietet es sich an, Karten mit entsprechenden Geräten um die relevanten Verbindungen zu erweitern. Schauen Sie sich als Beispiel Abbildung 9-13 an, in der über Deutschland verteilte Standorte mit den sogenannten Wetterlinien verbunden sind, die die Auslastung der entsprechenden Verbindungen anzeigen.

Sie können Ihren Karten sehr leicht entsprechende Darstellungen hinzufügen, wenn die von Ihnen genutzten Plugins die Vorgaben für die Rückgabe von Performancedaten einhalten (siehe hierzu Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)* beziehungsweise die entsprechenden Seiten des NagVis-Projektes unter *http://docs.nagvis.org/1.7/de_DE/lines_weathermap_style.html*).



Performancedaten: Leider verstehen sich Linien von NagVis nicht auf die Ausgaben der mitgelieferten Plugins, obwohl diese durchaus die geforderten Daten liefern. Um diese Plugins verwenden zu können, müssen Sie deshalb also ein Wrapper-Script einsetzen, um die Ausgaben der mitgelieferten Plugins entsprechend umzugestalten (siehe hierzu Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*). Alternativ können Sie auf andere Plugins zurückgreifen, die von Hause aus eine entsprechende Ausgabe liefern.

Für die hier von uns gezeigten Graphen haben wir ein Wrapper-Script eingesetzt, das die Ausgabe des Plugins `check_snmp_int.pl` (siehe Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*) entsprechend anpasst. Wir haben dabei folgende Parameter verwendet (hier zunächst die Ausgabe ohne Wrapper):

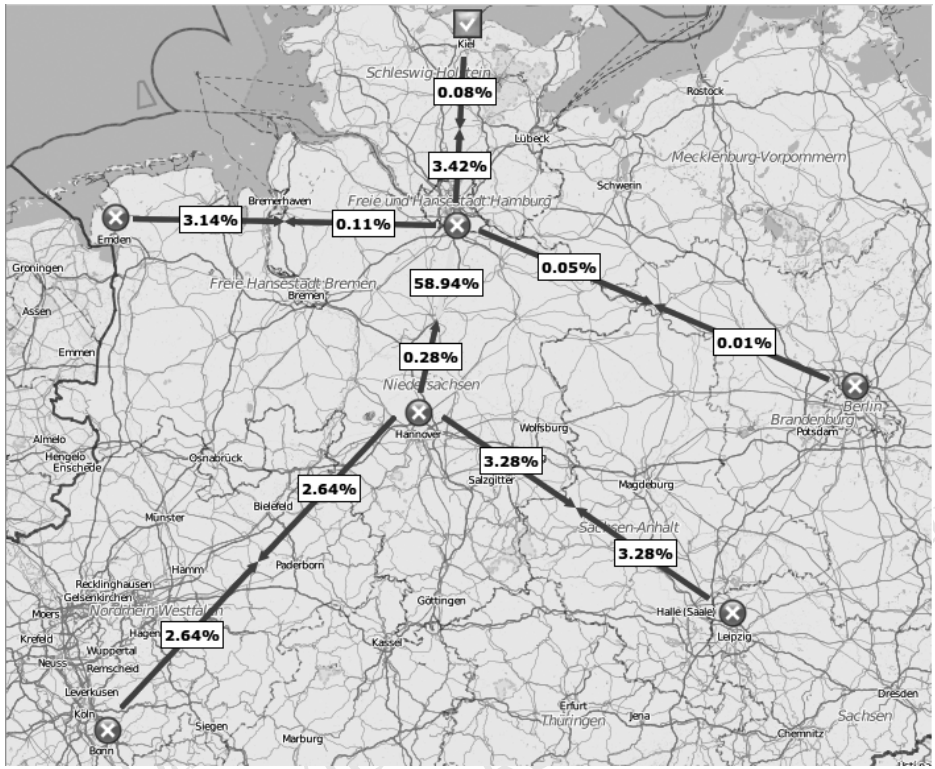


Abbildung 9-13: NagVis-Karte zur Standort-Visualisierung

```
icinga@moni:~$ /usr/local/icinga/libexec2/check_snmp_int.pl -H 10.85.58.1 -l icinga
-L sha,aes -x "Passphrase" -X "Passphrase" -n eth0 -k -f -Y -w 50,50 -c 90,90
eth0:UP (1.4KBps/3.5KBps):1 UP: OK | 'eth0_in_Bps'=1389;51200;92160;0;1000000000
'eth0_out_Bps'=3615;51200;92160;0;1000000000
```

NagVis erwartet genau die Werte, die das Plugin liefert, ist jedoch auf eine andere Benennung festgelegt, anstatt `'eth0_in_Bps'` erwartet NagVis ein `in` und anstelle von `'eth0_out_Bps'` ein `out`. Um die obige Ausgabe entsprechend zu ändern, können Sie ein Wrapper-Script wie in Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins* beschrieben verwenden, das Sie am Ende wie folgt abändern:

```
LINE=`echo $LINE | sed "s/eth0_in_Bps/in/"`
LINE=`echo $LINE | sed "s/eth0_out_Bps/out/"`
LINE=`echo $LINE | sed "s/'//g"`
echo $LINE
exit $RC
```

Die erste Zeile ersetzt in der zurückgegebenen Zeichenkette die Beschreibung für die eingehenden Daten, die zweite die für die ausgehenden und die dritte entfernt das Anführungszeichen. Die letzten beiden Zeilen geben die modifizierte Ausgabe sowie den in der Variablen `RC` gespeicherten Rückgabewert zurück.

Wenn Sie einen Dienst definiert haben, der die Werte im entsprechenden Format liefert, können Sie diesen mit den folgenden Schritten auf Ihrer Karte einbinden. Öffnen Sie die betreffende Karte über das Menü `Öffnen`. Um der Karte Wetterlinien hinzuzufügen, führen Sie nun jeweils die folgenden Schritte durch:

1. Wählen Sie `Karte bearbeiten / Linie hinzufügen` und dann `Dienst`.
2. Klicken Sie zunächst an der Stelle auf dem Hintergrund, an der die Linie starten soll. Klicken Sie anschließend auf die Stelle, an der die Linie enden soll.
3. In dem sich öffnenden Fenster können Sie nun zunächst die Maschine und dann den Dienst auswählen, dessen Daten für die Anzeige verwendet werden sollen.
4. Wählen Sie unter `line_type` aus, wie die Linie dargestellt werden soll. In unserem Beispiel hatten wir hier die Version mit der prozentualen Angabe ausgewählt. Es gibt auch eine Variante zur zusätzlichen Anzeige der tatsächlich genutzten Bandbreite, die Sie an dem enthaltenem `+BW` erkennen.

Nach diesen Schritten wählen Sie innerhalb des Dialogs den Button `Speichern`. NagVis wird die Karte nach einigen Sekunden neu laden und die Linie ab sofort entsprechend anzeigen.

Visualisierungen mit Widgets

Neben den Symbolen und Wetterlinien bietet NagVis sogenannte Widgets zur Visualisierung. Dies sind quasi Programme, die anhand der ihnen übergebenen Daten eine Darstellung berechnen und das Ergebnis als HTML-Code zurückgeben. Wir können hier nicht auf Details der Programmierung eingehen, sondern müssen Sie stattdessen auf die entsprechenden Seiten des Projektes unter *http://docs.nagvis.org/1.7/de_DE/gadgets.html* verweisen. Entsprechend gehen wir im Folgenden zur Veranschaulichung nur auf eines der mitgelieferten Gadgets ein, um deren generelle Funktion für Ihre Kartendarstellungen zu verdeutlichen. Es liegen NagVis jedoch neben dem hier gezeigten noch zusätzliche Gadgets bei und unter *http://exchange.nagvis.org/exchange/Gadgets/* enthalten die Seiten des NagVis-Projektes ein Verzeichnis mit weiteren Gadgets.

Ein recht vielseitig verwendbares Gadget ist das mitgelieferte Speed-o-Meter, das wir Ihnen im Folgenden vorstellen. Dieses kann insbesondere alle prozentualen Angaben, wie Sie sie beispielsweise von `check_snmp_load.pl` oder `check_snmp_storage.pl` erhalten, darstellen (zu den Plugins siehe Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*). Beim Speed-o-Meter handelt es sich dabei eine einem Tacho ähnelnde Anzeige, die die Warnwerte für die Status WARNING und CRITICAL farbig hervorhebt und den aktuellen Wert über eine Tacho-Nadel anzeigt. Betrachten Sie Abbildung 9-14 als Beispiel.

Die Abbildung zeigt zwei Tacho-Anzeigen: der linke Tacho stellt eine CPU-Auslastung dar, während der rechte eine Festplattenauslastung anzeigt. Für eine Berechnungsmaschine beziehungsweise einen Dateiserver sind dies die maßgeblichen Werte, die Sie auf diese Weise deutlich hervorheben können.

Abbildung 9-14: NagVis-Karte zur Schrank-Visualisierung mit Linien und Gadgets

Zur Nutzung von Gadgets öffnen Sie die entsprechende Karte (Menü `Öffnen`) und führen Sie die folgenden Schritten durch:

1. Wählen Sie `Karte bearbeiten / Icon hinzufügen` und dann `Dienst`.
2. Klicken Sie zunächst an der Stelle auf dem Hintergrund, an der das Widget angezeigt werden soll.
3. In dem sich öffnenden Fenster wählen Sie nun zunächst die Maschine und dann den Dienst aus, dessen Daten für die Anzeige verwendet werden sollen.
4. Ändern Sie die Eigenschaft `view_type` auf `Gadget`.
5. In den daraufhin zusätzlich angezeigten Option wählen Sie gadget_url aus und setzen es auf `std_speedometer2.php`.

Speichern Sie Ihre Änderungen durch Klick auf den Button `Speichern`. NagVis wird die Karte neu laden und das Widget anzeigen.

Größenänderungen: Wenn Ihnen ein Widget zu groß erscheint, können Sie es über die Option `gadget_scale` leicht anpassen. Geben Sie hierzu einfach den prozentualen Wert der eingangs angezeigten Größe an und NagVis wird die Anzeige entsprechend anpassen.

Diskussion

Mit Wetterlinien und Gadgets können Sie die einfachen Darstellungen (siehe Rezept 9.6, *Verwendung von NagVis und Erstellung von Karten*) bereits erheblich erweitern. Insbesondere können Sie sich aber auch eigene Gadgets nach Bedarf erstellen, wenn Sie spezifische Vorstellungen von einer Darstellung haben. Bei dieser Art Erweiterungen muss sich jedoch nicht Nagios/Icinga auf das Format der Rückgabe durch die Plugins verstehen, sondern auch die entsprechende Komponente von NagVis. Nicht alle Plugins sind damit grundsätzlich für solche Erweiterungen nutzbar, auch wenn Sie in Nagios/Icinga vielleicht durchaus anstandslos ihren Dienst verrichten. Entsprechend machen Erweiterungen dieser Art schnell viel Arbeit: Suche nach geeigneten Plugins, Einbindung und gegebenenfalls Anpassungen verursachen einen nicht zu vernachlässigenden Aufwand. Diesem Aufwand steht der Nutzen gegenüber: angepasste Darstellungen veranschaulichen eine Sachlage häufig sehr viel besser als die reine Datenanzeige in der Oberfläche von Nagios/Icinga.

Entsprechend müssen Sie zunächst abwägen, inwiefern sich der zusätzliche Aufwand lohnt. Typischerweise wird dies zunächst bei grundlegenden Sachverhalten der Fall sein, die sich selten ändern. Haben Sie dann aber passende Plugins im Einsatz, können Sie den Einsatz ohne weiteren Aufwand einfach ausdehnen.

Dynamische Ansichten: Seit Version 1.7 unterstützt NagVis über eine API dynamische Quellen für Ansichten, die sogenannten `Map Sources`. Diese ermöglichen es, mit Hilfe von PHP ganz eigene Ansichten zu programmieren und dabei etwa externe Datenquellen einzubinden oder Verarbeitungsschritte durchzuführen. Insbesondere können Sie damit die Ansicht selbst in Abhängigkeit von Zuständen ändern. Die Seite *http://docs.nagvis.org/1.7/de_DE/maps.html* enthält die Dokumentation des Projektes zu diesen neuen Möglichkeiten.

Siehe auch

- Rezept zu den einfachen Darstellungen mit NagVis: Rezept 9.6, *Verwendung von NagVis und Erstellung von Karten*
- Rezept zur Programmierschnittstelle von Nagios/Icinga: Rezept 7.1, *Einführung zur Plugin-Schnittstelle von Nagios/Icinga (API)*
- Seiten des NagVis-Projektes zu den Wetterlinien: *http://docs.nagvis.org/1.7/de_DE/lines_weathermap_style.html*
- Rezept zum Wrapper-Script: Rezept 7.2, *Ein einfaches Beispiel eines benutzerdefinierten Plugins*
- Seiten des NagVis-Projektes zu Gadgets: *http://docs.nagvis.org/1.7/de_DE/gadgets.html*
- Verzeichnis vorhandener Gadgets auf exchange.nagvis.org: *http://exchange.nagvis.org/exchange/Gadgets/*

- Rezept zu den angesprochenen Manubulon-Plugins: Rezept 7.5, *Einbinden externer Plugins am Beispiel von Manubulons SNMP-Plugins*
- Seiten des NagVis-Projektes zu den dynamischen Map Sources: *http://docs.nagvis.org/1.7/de_DE/maps.html*

9.8 Karten in NagVis zu Rotationspools zusammenfassen

Problem

Sie möchten bei der Anzeige von Karten einen automatischen Wechsel zwischen bestimmten Karten erreichen.

Lösung

Hierzu sieht NagVis sogenannte Karten-Rotationen vor, bei denen die enthaltenen Karten für eine bestimmte Zeit angezeigt werden, bevor NagVis automatisch zur nächsten enthaltenen Karte wechselt. Wir zeigen Ihnen im Folgenden, wie Sie Darstellungen zu gegebenenfalls unterschiedlichen Rotationspools zusammenfassen und verwenden können.

Die NagVis-Konfigurationsdatei

NagVis bietet für die Erstellung von Rotationspools keine Funktionalität in der Oberfläche. Stattdessen müssen Sie Rotationspools in der zentralen Konfigurationsdatei von NagVis anlegen. Je nachdem, wie Sie NagVis installiert haben (siehe hierzu Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*), finden Sie diese entweder in `/usr/local/nagvis/etc/nagvis.ini.php` (quellinstallation) oder in `/etc/nagvis/nagvis.ini.php` (paketbasiert). Wir gehen im Folgenden zunächst auf die allgemeinen Optionen für Rotationen ein, bevor wir Ihnen die Definition eigener Rotationen erläutern.

Allgemeine Optionen für alle Rotationen

Die Konfigurationsdatei enthält zunächst für alle Rotationen geltende Optionen. Dies ist zum einen die Angabe, ob Rotationen auf der Karten-Übersicht angezeigt werden (`0` für nein, `1` für ja):

```
; Enable/Disable rotation listing
showrotations=1
```

Desweiteren sieht NagVis einen Vorgabewert für die Dauer der Anzeige der einzelnen Karten in Sekunden vor. Diesen können Sie mit der folgenden Konfigurationsoption festlegen:

```
; Default rotation time of pages in rotations
refreshtime=60
```

Bei dem hier vorgesehenen Wert von 60 Sekunden würde jede Karte entsprechend eine Minute lang angezeigt, bevor zur nächsten Karte in der Rotation gewechselt wird.

Definition von Rotationen

In dieser Datei ist bereits die Demo-Rotation definiert, die die mitgelieferten Karten umfasst. In unserer Installation war dies der folgende Abschnitt:

```
[rotation_demo]
maps="demo-germany,demo-ham-racks,demo-load,demo-muc-srv1"
interval=15
```

Die hinter dieser Definition liegende Syntax umfasst zunächst die Angabe des Namens `demo` als Anhang zur Einleitung der Definition durch das in eckigen Klammern angegebene `rotation_`. Die zur Rotation gehörenden Karten werden durch das folgende `maps="name1,name2,..."` angegeben. Schließlich können Sie optional über `interval=anzahl-sekunden` noch ein nur für diese Rotation geltendes Anzeige-Intervall festlegen.

Eine eigene Rotation können Sie entsprechend einfach definieren. Die Definition einer Rotation mit dem Namen `kochbuch` für die Karten `Schrank`, `Campus` und `deutschland` ließe sich also beispielsweise mit einem Abschnitt wie dem folgenden realisieren:

```
[rotation_kochbuch]
maps="Schrank,Campus,deutschland"
```

Hierbei haben wir kein spezifisches Intervall für die Anzeige angegeben, so dass der weiter vorne angesprochene Wert für alle Rotationen verwendet wird.

Nutzung von Rotationen

Alle in der Konfigurationsdatei definierten Rotationen sind über einen entsprechenden URL aufrufbar. Die eben definierte Rotation mit dem Namen *buch* können Sie beispielsweise über den URL */nagvis/frontend/nagvis-js/index.php?rotation=buch* aufrufen. Alternativ finden Sie alle Rotationen auch auf der Übersichtsseite (in NagVis: `Öffnen / Übersicht`) und können Sie dort per Klick anwählen. Nach dem Aufruf einer Rotation wird NagVis Ihnen die erste betreffende Karte für die angegebene Dauer anzeigen. In der Menüzeile von NagVis wird dabei ein Zähler eingeblendet (links neben dem Nutzermenü), der die verbleibenden Sekunden bis zum nächsten Kartenwechsel anzeigt. Abbildung 9-15 zeigt Ihnen die Menüzeile von NagVis mit einem solchen Zähler.

Abbildung 9-15: NagVis Menü mit Zähler für die Rotation

Nach der letzten Karte einer Rotation wird NagVis dann wieder zur ersten Karte springen und die Rotation so erneut beginnen. Entsprechend laufen Rotationen durchgehend, also bis Sie sie manuell wieder stoppen. Dies geschieht automatisch, wenn Sie etwa eine einzelne Karte durch Auswahl im Menü öffnen.

Diskussion

Rotationen sind erst dann eine hilfreiche Funktion, wenn Sie mehrere Karten erstellt haben. Sie eigenen sich hervorragend zur automatischen Anzeige, etwa auf von für die Allgemeinheit zugänglichen Monitoren. Dort können Sie die Status-Übersicht von Nagios/Icinga ersetzen. Da Sie mehrere Rotationen erstellen können, lassen sich bei umfangreichen Installationen etwa für einzelne Abteilungen passende Ansichten zu Rotationspools zusammenfassen, die dann entsprechend den jeweiligen Aufgabenbereich abdecken.

Siehe auch

- Rezepte zur Installation von NagVis im Rahmen der Installation von Nagios/Icinga: Kapitel 1, *Nagios/Icinga installieren und Hostsystem vorbereiten*

9.9 Ausblick auf das kommende Icinga2

Problem

Sie möchten wissen, in welche Richtung sich Icinga entwickelt.

Lösung

Wir vermitteln Ihnen im Folgenden einen Eindruck von den Plänen des Icinga-Projektes und absehbaren Weiterentwicklungen. Icinga ist aus dem Nagios-Projekt hervorgegangen, als dieses scheinbar nicht mehr aktiv weiterentwickelt wurde. Tatsächlich war Icinga zunächst eine Kopie von Nagios, die im Gegensatz zu dem ursprünglichen Projekt eben aktiv weiterentwickelt wurde. Dabei sind die Icinga-Entwickler auf eine Vielzahl von Beschränkungen gestoßen, die mit dem Versionssprung auf Version 2 behoben werden sollen. Hierzu schreiben Sie das Programm derzeit von Grund auf neu, wohl auch wegen der Qualität des Quellcodes, der über Jahre »gewachsen« ist. Nach eigenen Angaben haben sie dabei versucht, für das nächste Jahrzehnt zu planen, also für ein durchaus in der IT-Welt langfristiges Zeitfenster.

In der neuen Version wird dabei die Kompatibilität zu Nagios beibehalten. Um jedoch bestehende Beschränkungen zu umgehen wird ebenfalls ein Icinga2-eigenes Format eingeführt. Damit ein Umstieg leicht fällt und die bisherigen Plugins weiter verwendet werden können, wird Icinga2 über einen Nagios-Kompatibilitätsmodus verfügen. Während dann über das neue Konfigurationsformat die Verbesserungen zur Verfügung stehen, wird sich das neue System alternativ also auch weiter mit dem alten Format verwenden lassen. Auf diese Weise steht eine zunächst einfache Migration in Aussicht, wobei sich die Vorteile allerdings wohl erst vollständig erschließen werden, wenn auch die Konfiguration migriert wurde.

Zu den geplanten Verbesserungen von Icinga2 gehört insbesondere die Geschwindigkeit bei der Verarbeitung. Diese ist bisher vor allem bei der Skalierung auf umfangreichen Installationen ein Problem. Grundlage der Verbesserung wird ein modularer Aufbau sein, der es ermöglicht, nur tatsächlich benötigte Module zu laden und andere, nicht benötigte Module außen vor zu lassen. Dabei berücksichtigen die Entwickler auch komplexe Szenarien, wie die Zusammenarbeit mehrerer Instanzen (Clustering beziehungsweise verteilte Überwachung) und Abbildung auf Geschäftsprozesse, die in Nagios bislang nur auf Umwegen umsetzbar und ursprünglich gar nicht vorgesehen waren. Das Gleiche ist bei der dynamischen Konfiguration der Fall, also bei der Änderung von Konfigurationsoptionen zur Laufzeit. Dies war bislang gar nicht möglich, wird aber von Icinga2 unterstützt werden.

Durch diese Änderungen werden dann auch Änderungen an den Benachrichtigungen zugelassen. Bestehende Redundanzen und Doppeldeutigkeiten, wie beispielsweise bei der Dualität zwischen Eltern-Kind-Beziehungen im Gegensatz zu Maschinen- beziehungsweise Dienstabhängigkeiten sollen dabei behoben werden. In diesem Zuge werden die host-checks wegfallen, so dass nur noch Dienste ausführbare Anweisungen enthalten werden. Daneben adressiert die Neuentwicklung aber auch das wichtige Thema Sicherheit (durch Unterstützung von SSL und X.509 Zertifikaten). Auch das Internet Protocol wird dann durchgängig in Version 6 unterstützt werden.

Neben den festgelegten Details spricht aber insbesondere der auf Schnittstellen basierende Ansatz für den kommenden Nachfolger. Diese werden auch Nutzungen erlauben, die vielleicht noch gar nicht absehbar sind. Daneben gibt es weitere, vielversprechende Details von den Dokumentationsplänen bis hin zu im Speicher ausgeführten Plugins.

Diskussion

An der kommenden Version arbeiten die Icinga-Entwickler nach eigenen Angaben bereits seit März 2012. Erste Prototypen wurden bereits auf Konferenzen präsentiert, so dass Icinga2 bereits weit über den Ideen-Status hinaus gediehen ist. Als Ausblick lässt sich zusammenfassen, dass die Entwicklungspläne sehr den Eindruck vermitteln, dass wünschenswerte Änderungen umgesetzt werden. In Anbetracht der bisherigen Erfolge des Icinga-Projektes (beispielsweise das Web-Interface Icinga-Web als besonders offensichtliche Weiterentwicklung) scheinen die Pläne auch durchaus glaubhaft. Inwiefern das Projekt seinen Zeitplan einhalten kann, wird natürlich wie bei jedem Software-Projekt spannend zu beobachten sein. Die erste Veröffentlichung von Icinga2 ist für Ende 2013 geplant.

Siehe auch

- Webseiten des Icinga-Projektes zu Icinga2: *https://www.icinga.org/about/icinga2/*

Index

Symbole

$USERn$-Makros 260

A

Abhängigkeiten zwischen Diensten 300
Abhängigkeiten zwischen Maschinen 299
accept_passive_host_checks 199
accept_passive_service_checks 199
access denied 35
ACKNOWLEDGED 204
Acknowledgements 265
Acknowledgment expiry 266
action_url 307
Active Directory (AD) überwachen 523
active_checks_enabled 272, 277
address 271
Administrator-Passwort für MySQL 13
AES-Verschlüsselung 158, 168, 392
aktive Infrastruktur 491
aktive Prüfung 199, 200, 202
Aktualisierung 133
Aktualisierungen überwachen 355
Alert Histogram (Bericht) 216
Alert History (Bericht) 216
Alert Summary (Bericht) 216
alias 271, 294, 295, 297
alias_up 475
Alias-Kommandos 342
angemeldete Nutzer 311
Antwortzeit 362, 365
API (Application Programming Interface) 433
apt 467
apt-get 30, 44, 93, 104, 144
apt-get autoremove 137, 138, 143
aptitude 30, 44, 93, 104, 144, 467
Arbeitsabläufe 267
Ausfallzeiten 264
Autostart 115, 125
Availability (Bericht) 216

B

Backports (Debian) 29
Backup 249
Backup *siehe* Datensicherung
Backup-Script 250
Bedienung 203
Befehlsdefinition 282
Benachrichtigungen 257
  siehe auch Berichte
  Befehlsdefinition 259
  Kontaktgruppen 258
  Notifications (Bericht) 217
  Rolle 258
  Zeitfenster 259
Benutzeranzahl überwachen 311
benutzerdefinierte Plugins 437
Benutzerkonten 195, 286
Benutzerrechte testen 343
Berichte 216
  Alert Histogram 216
  Alert History 216
  Alert Summary 216
  Availability 216
  Event Log 217
  Notifications 217
  Trends 216
  Verfügbarkeit 216
Bezeichner-Länge 435
BlueCoat 456
Brocade 455

C

can_submit_commands 287
cfg_file 238

cgi.cfg 197
CGI-Optionen 290
check_apache 507
check_apt 166, 355
check_by_ssh 350
check_clamd 367
check_command 270, 272, 277
check_dhcp 155, 380, 521
check_dig 380, 521
check_disk 322
check_dns 376, 519, 523
check_dummy 282, 283, 434
check_external_commands 36
check_freshness 279
check_ftp 367, 400, 500
check_http 403, 508
check_icmp 155, 359, 363, 458
check_ide_smart 327, 464
check_ifoperstatus 396, 459
check_ifstatus 391, 459
check_imap 367, 415
check_interval 272, 277
check_jabber 367, 373
check_ldap 418, 522, 523
check_load 314
check_log 336
check_logfiles 336
check_mysql 425, 515
check_mysql_health 518
check_mysql_query 425, 517
check_nntp 367
check_nntps 367
check_nrpe 175, 179, 339, 345
CHECK_NRPE, Error 340
check_nt 180, 344, 345
check_ntp 370, 461
check_ntp_peer 370, 461
check_ntp_time 370, 461
check_oracle_health 431, 517, 518
check_period 271, 277
check_ping 359, 458
check_pnp_rrds 445
check_pop 367, 415
check_postgres 431, 517, 518
check_procs 317, 438, 464, 506, 513
check_sensors 330, 466
check_simap 367, 415
check_smb 421, 498, 523
check_smtp 367, 412, 504
check_snmp 384, 507, 513
check_snmp_apt 467
check_snmp_env 455
check_snmp_environment 455
check_snmp_int 453, 465
check_snmp_load 454, 461
check_snmp_mem 452, 462
check_snmp_storage 452, 463
check_snmp_xyz (Manubulon) 450
check_spop 367, 415
check_ssh 409
check_ssmtp 367
check_tcp 367, 510
check_udp 367
check_updates 356, 467
check_users 312
check_zypper 356, 467
CIFS/SMB überwachen 421, 498, 523
Cisco 454, 455, 456
Cisco-Router 494
Citrix 456
CLAMD überwachen 367
command 259, 281
command dropdown 209
command_line 282
command_name 282
contact 286, 296
contact_groups 271, 277
contact_name 287
contactgroup_members 297
contactgroup_name 297
contacts 271, 277
CPU-Auslastung überwachen 314, 454, 461, 475, 492
CRITICAL 258
CRITICAL SMB anon access 423
cron 251
Cronks 220

D

Dateien durchsuchen 244
Dateiserver überwachen 498
  CIFS 498
  Festplattenbelegung 498
  FTP 500
  Netzwerk-Freigaben 498
  NFS 499
  SMB 498

Dateisysteme überwachen 322
Datenbank sichern 252
Datenbankserver 512
  Abfragen 517
  MariaDB 16, 58, 82, 515
  MySQL 31, 47, 82, 126, 427, 430, 515
  Oracle 5, 16, 517
  PostgreSQL 5, 16, 517
  Prozesse 513, 514
  überwachen 512
Datensicherung 249
Datenträgerauslastung *siehe* Speicherbelegung
Datenvisualisierung 525
DBPASSWD 34
Debian
  Icinga deinstallieren 137
  Icinga installieren 12, 29
  Icinga-Web 19, 33
  IDOUtils 16, 31
  Nagios deinstallieren 142
  Nagios installieren 78, 93
  NagVis 25, 40, 89, 100
  PNP4Nagios 21, 36, 85, 97
debmon repository 30
Debugging *siehe* Fehlersuche
Deinstallation *siehe* Icinga und Nagios
dependency_period 300, 301
dependent_host_name 299, 300, 301
dependent_hostgroup 299, 301
dependent_service_description 301
DES 158
devices 493
DHCP-Server überwachen 380, 521
Dienstdefinition 277
Dienste überwachen (Windows) 482
Dienste, Abhängigkeiten 269, 300
Dienstgruppen 295
display_name 273, 278
DNS cache poisioning 520
DNS-Server überwachen 371, 376, 519
DNS-Spoofing 379
DOWN 258
Dummy-Kommando 202

E

Eltern-Kind-Beziehung 272, 299
E-Mail-Server 412, 415, 501
  Authentifizierung 504
  überwachen 412, 415, 501
  Zertifikate 503
enable_flap_detection 262
Entscheidungshilfe 3
EPEL (CentOS) 124
Erreichbarkeit 46, 58, 359, 363, 458, 472, 492
  überwachen mit ping 359
  überwachen über ICMP 363
Escapes 428
Event Broker 18
Event Log (Bericht) 217
event_handler_enabled 273, 279
Execute bit 450
execution_failure_criteria 299, 301
Expander Icon 40
externe Kommandos verarbeiten 36
externe Plugins 432
Extreme 456

F

Fehlersuche 255
  debug_level 255
  Hauptkonfigurationsdatei 254
  Icinga-Web 255
  IDOUtils 256
  NDOUtils 256
  PNP4Nagios 256
  Protokolldateien 254
Fehlersuche_debug_verbosity 255
Festplatten *siehe* Speicherbelegung
Festplattenstatus überwachen 327
FibreAlliance 496
find 246, 537
Firewalls 493
Firmware 294
flap_detection_enabled 273, 279
FLAPPING 204
Flattern Status *siehe* Status, Flapping
Foundry 455, 456
freshness 201
Frische-Prüfung 201
FTP-Server 400, 500

G

Gadgets 552
Geomap 548
git 85, 89
Graphen entfernen 535

Graphen generieren 437, 526
Graphen vereinen 537, 542
grep 244
Gruppen-Freischaltung 198
gzip 250

H

HANDLED Status 204
Hardware-Anforderungen 9
Hauptkonfigurationsdatei 199, 238, 262
  siehe auch Icinga, icinga.cfg und Nagios, nagios.cfg
Hauptspeicher *siehe* Speicherbelegung
Hersteller MIB *siehe* MIB
Hintergründe erstellen 547
Hintergrundbild 548
host *siehe* Konfigurationsobjekte, host
host_name 271, 273, 276, 299, 301, 306
host_notification_commands 260, 287
host_notification_options 287
host_notification_period 259, 287
host_notifications_enabled 287
host-check 268, 270, 272
host-dependency 269
hostextinfo 305
hostgroup_members 294
hostgroup_name 276, 294, 299, 301
hostgroups 273
Hosts *siehe* Maschinen
HP ProCurve 455, 456
HTML-Tags 306
HTTP 403
http-auth 195
HTTPS 149, 403
Hypervisor überwachen 486

I

Icinga 284
  aus den Quellen installieren
    icinga.cfg 134, 135, 199, 238, 254, 262
    icingastats 10
    Icinga-Web 19
    IDOUtils 16
    Installation 12
    Nagios-Plugins (über Git) 14
    NagVis 25
    PNP4Nagios 21
Icinga PPAs 44
Icinga-Web 19, 220
  Cronks 220
  Datenzugriff
    Ausfallzeiten 228
    Benachrichtigungen 228
    Gruppendarstellungen 227
    Instances 229
    Maschinen- und Dienststatus 225
    Protokollansicht 229
    Statuskarte 229
    Unhandeled 224
  Konfigurationsoptionen 231
  Menüleiste 221
  Reporting 230
  taktische Übersicht
    TO Charts 222
    TO CustomVariables 223
    TO Hostgroups 223
ICMP 359, 363
ICMP Echo Request 361, 365, 472
icon_image 278, 308
icon_image_alt 308
ido2db 465
ido2db.cfg 134, 256
idomod 48
IDOUtils 16
IDOUtils sichern 252
IMAP 415
Infrastruktur 491
inherits_parent 299, 301
insserv 13, 31, 46, 94, 105
Installation *siehe* Icinga und Nagios
IPv4/6 362
IRC-Server überwachen 371
IronPort 456

J

Jabber-Server überwachen 372
Jasper-Server 217
Juniper 455, 456

K

Karten ändern 550
Karten erstellen 545, 549
Kerberos 523
Kommentare 265

Komplexität 240
Konfigurationsdateien 218, 237, 238, 268
  Abhängigkeiten 237
  Änderungen 243
  Aufbau 237
  Ordnung 233, 237
  sichern 249
  Struktur 239
  Suche in 244
  Vorlagen 233
Konfigurationsobjekt 268
  Änderungen 244
  command 237, 282
  contact 195, 286, 296
  contactgroup 296
  host 195, 237, 270
    address 271
    alias 271
    check_period 271
    contact_groups 271
    contacts 271
    host_name 271
    max_check_attempts 271
    notification_interval 271
    notification_period 271
  hostdependency 299
  hostextinfo 305
  hostgroup 293
  service 195, 276
  servicedependency 300
  serviceextinfo 305
  servicegroup 295
  Timeperiod 290
Konfigurationsoptionen 282
  command 282
  contact 284
  contactgroup 284
  host 284
  hostgroup 284
  service 284
  servicegroup 284
  timeperiod 287
Konfigurationsverzeichnis 239
Kontaktdefinition 286, 296
Kontaktgruppe 296
kritische Schwelle *siehe* CRITICAL

L

Latenzzeiten 10
Laufzeit 218, 254, 360, 446, 460, 474
LDAP-Server überwachen 418
Linux-Server 456
  Arbeitsspeicher 462
  CPU-Last 461
  Datenträgerauslastung 463
  Datenträgerstatus 464
  Erreichbarkeit 458
  Netzwerk-Schnittstellen 465
  Prozessanzahl 465
  Prozessor-Auslastung 461
  Sensordaten 466
  SMART 464
  Speicherauslastung 462
  überwachen 458
  Umweltparameter 466
  Update-Status 467
  Uptime 460
lm_sensors 330, 331, 466
Log-Dateien überwachen 336
Logik 204
lokale Plugins 312
Lüfter 493

M

mail 260
Makros 218, 237, 260
  $USER$-Makro 284
  Befehlsargument-Makro 284
  resource.cfg 284
Makros (Platzhalter) 283
managed 493
Managementnetzwerk 7
Manubulon 448
  C-Paket 456
MariaDB 16, 58, 82, 515
Maschinen 270
Maschinen gruppieren 293
Maschinen, Abhängigkeiten 269, 299
Maschinendefinition *siehe* Konfigurationsobjekte, host
Maschinenoptionen
  address 271
  alias 271
  check_period 271
  contact_groups 271

contacts 271
host_name 271
max_check_attempts 271
notification_interval 271
notification_period 271
max_check_attempts 271, 277
members 294, 295, 297
MIB 162
siehe auch SNMP, MIB
installieren 163
nachpflegen 164
OID 161
OID übersetzen 164
Standard-MIBs 163
Migration 133
Mumble-Server überwachen 369
MySQL 58, 425, 513, 517
Passwörter 32
Server überwachen 425

N

Nagios
aus den Quellen installieren
Installation 78
nagios.cfg 83, 86, 96, 98, 107, 108, 109, 117, 119, 127, 129, 134, 135, 254
Nagios-Plugins (aus tarball) 80
nagiosstats 10
NagVis 89
NDOUtils 82
PNP4Nagios 85
Nagios Service Check Acceptor (NSCA) 203
NagVis 545
Gadgets 552
Geomap 548
Hintergründe 547
Karten 545, 549, 550
sichern 250
siehe auch Icinga, Installation und Nagios, Installation
Symbole 549
Wetterlinien 552
Widgets 554
nagvis.ini.php 26, 557
Nameserver *siehe* DNS-Server
ndo2db.cfg 83, 117, 127, 134, 135, 256
NDOUtils sichern 252
NDOUtils *siehe* Nagios, Installation
NetBIOS 423
NetScreen 455, 456
network 493
Network Traffic 367, 465
Netzteile 492
Netzwerk
Firewalls 493
Infrastruktur 491
managed network devices 493
PoE 496
Router 493
SAN 496
Schnittstellen 453
überwachen 391, 396
Switches 493
USV 495
Verkehr 366, 465
VPN-Gateways 493
Netzwerkanbindung 7
Netzwerke überwachen 493
Netzwerkgeräte
überwachen 492
CPU-Last 492
Erreichbarkeit 492
Lüfter 493
Seriennummern 493
Speichernutzung 492
Stromversorgung 492
Temperatur 492
Versionsstände 493
Netzwerkschnittstellen 465
überwachen 391, 396, 453
Netzwerkverkehr 366, 465
überwachen 465
NFS-Server 499
NNTP überwachen 367
no data 35
Nokia 456
notes 278, 306
notes_url 307
notification_failure_criteria 299, 301
notification_interval 271, 277
notification_options 258, 273, 278
notification_period 259, 271, 277
Notifications (Bericht) 217
notifications_enabled 273, 278
notify-host-by-email 259
notify-service-by-email 259

NRPE 174, 179, 339, 343, 475
nrpe.cfg 175, 177
NRPE-Server
  unter Unix installieren 174
  unter Windows installieren 179
NSCA 180
NSClient 180, 339, 344
NSClient++ 179, 339, 344
  externe Scripte 182
  installieren (Windows) 179
  konfigurieren (Windows) 179
  timeout 184
NSClient-Server installieren (unter Windows) 179, 344
NTP-Server überwachen 370
Nur-Lese-Zugriff 198

O

Objekte 268
Objekte *siehe* Konfigurationsobjekte
OID *siehe* MIB und SNMP
OK *siehe* Status
OpenSolaris 458
Optionen 282
Optionen *siehe* Konfigurationsoptionen
Oracle 5, 16, 517
Oracle *siehe* Datenbankserver

P

Paketabhängigkeiten 36, 49
Paketmanager 467
parents 272
passive Prüfung 199, 200, 202, 203
  Addons 203
  Dummy-Kommando 202
passive_checks_enabled 272, 278
Passwortdefinition 134
PENDING 204
Performancedaten 437
  verarbeiten mit prcess_performance_data 22
  Wrapper-Script 439
Performancedaten migrieren 532
Ping 359
Plugin-API 433, 434, 435
Plugins 282
  auf einer entfernten Maschine ausführen 339
  ausführen 350
  check_apt 166
  check_by_ssh 178
  check_dhcp 155, 157
  check_icmp 155, 157
  check_ifoperstatus 161, 173
  check_ifstatus 161, 173
  check_nrpe 175, 178, 180
  check_nt 180
  check_snmp 161, 167, 173
  Daten übergeben 433
  eigene erstellen 433, 437
  externe einbinden 448
  Rückgabewerte 433
  Status-Codes 434
PNP4Nagios 525
  Altdaten übernehmen 533
  Daten löschen 536
  Graphen vereinen 537
  mehrere Einzelgraphen 542
  Nutzung 526
  Pages 542
  Plugin 445
  sichern 250, 252
  Special Templates 538
  Übersichten von Graphen 543
  Weboberfläche 527
  Wrapper-Script 439
PNP4Nagios *siehe* Icinga, Installation
PNP4Nagios *siehe* Nagios, Installation
pnp4nagios.conf 22, 38, 51
PoE 496
POP3 415
Port, TCP 366
Port, UDP 366
Postgres-Datenbankserver 5, 16, 517, 518
printf 260
process_perf_data 273, 279
process_performance_data 22
ProCurve 455
Protokolldatei überwachen 336
Prozesse 267
  überwachen 317, 464, 480
Prozessorauslastung überwachen 314, 454, 461, 475, 493
Prüfungen 199
  aktiv 199
  passiv 199
Prüfungsergebnisse zusammenfassen 440
purge 137, 139, 143, 145

Q

queue send error 97

R

RAM *siehe* Speicherbelegung
RECOVER 258
register 270
reguläre Ausdrücke 244
remove 137, 139, 143, 145
RepoForge (CentOS) 70
Reporting *siehe* Weboberfläche, Berichte
Repositories
  debmon (Debian) 30
  EPEL (CentOS) 124
  RepoForge (CentOS) 70
resource.cfg 134
retain_nonstatus_information 273, 278, 287
retain_status_information 273, 278, 287
retry_interval 272, 277
RHEL|CentOS|Fedora
  Icinga deinstallieren 141
  Icinga installieren 69
  Icinga-Web 74
  IDOUtils 72
  Nagios deinstallieren 147
  Nagios installieren 123
  NagVis 76, 131
  NDOUtils 126
  PNP4Nagios 76, 128
Rollen 296
Rollenkonzept 198
root-Rechte 155
  setuid 156
  setuid-Bit 156
  sudo 156
  visudo 156
Router überwachen 493
Routing 9
RRDs überwachen 445
Rückgabewert definieren 283, 284, 434

S

SAMBA-Server 422
SAN 496
SCHEDULED 258
Schwellwerte *siehe* Status
Sensor-Daten überwachen 330
Sensoren überwachen *siehe* lm-sensors
Seriennummern 493
Server
  AD-Server 523
  CIFS-Server 498, 523
  Dateiserver 498
  Datenbankserver 512
  DHCP-Server 380, 521
  DNS-Server 376, 519
  E-Mail-Server 367, 412, 415, 501
  FTP-Server 367, 400, 500
  IMAP-Server 415
  Jabber-Server 372
  LDAP-Server 418, 522
  Linux-Server 458
  Mumble-Server 369
  MySQL-Server 425, 513
  NFS-Server 499
  NRPE-Server 174, 179, 339
  NSClient-Server 179, 344
  NTP-Server 370
  POP3-Server 415
  SMB-Server 421, 498, 523
  SMTP-Server 412
  SNMP-Server 158, 166, 168, 384, 448
  SSh-Server 187, 350, 409
  Unix-Server 458
  virtuelle Maschinen 486
  Webserver 403
  Windows-Server 470
Server-Prozesse überwachen 506
service 195
service_description 276, 301, 306
service_notification_commands 260, 287
service_notification_options 287
service_notification_period 259, 287
service_notifications_enabled 287
service-dependency 269
servicedependency (Objekte) 269
serviceextinfo 305
servicegroup 295
servicegroup_members 295
servicegroup_name 295
servicegroups 278
Serviceport überwachen 366
setuid 156
show_all_services_host_is_authorized_for 195
Sicherung 249

SLES|openSUSE
Icinga deinstallieren 140
Icinga installieren 56
Icinga-Web 61
IDOUtils 58
Nagios deinstallieren 146
Nagios installieren 113
NagVis 67, 120
NDOUtils 116
PNP4Nagios 62, 119
SMART 327, 464
SMB-Dienst 421
SMB-Freigabe 422
SMB-Server überwachen 421, 498, 523
SMTP 412
SMTPs 412
SNMP 158, 166, 168, 384, 448, 449
Ausführen von Plugins (unter Linux) 166
Community 170
Daten abfragen 448
erweitern 166
MIB 163
Namensauflösung 163
OID 162
snmpd 158
Wert auslesen 384
SNMP, MIB *siehe* MIB
SNMP-Plugin, Manubulon 448
SNMP-Server
Installation 158
installieren (Windows) 168
Konfiguration 159
konfigurieren (Linux) 165
konfigurieren (Windows) 165
unter Linux installieren 158
unter Windows installieren 168
WMI 169
Solaris 458
Special Templates 491, 530, 537
Speicherauslastung *siehe* Speicherbelegung
Speicherbelegung
check_by_ssh 350
Hauptspeicher 452, 478
Medien (beliebig) 450
Medien (beliebige) 452
Swap-Speicher 452
SSH 187, 350, 409
authorized_keys 190
Fingerabdrücke 188
installieren (unter Linux) 187
known_hosts 188
Multiplexing 355
Schlüssel 189
Server
installieren (Linux) 187
konfigurieren (Linux) 192
Sicherheit 189
ssh-copy-id 190
ssh-keygen 189
SSH-Server überwachen 409
SSL 149
SSL, Absichern von Verbindungen 149
SSL-Verschlüsselung 509
Status
Acknowledged 204
Dienste 199
RECOVER 258
Flapping 204, 207, 261
GUI 204
Handled 204
Maschinen 204
CRITICAL 258
DOWN 258
FLAPPING 258
NONE 258
OK 258
RECOVER 258
SCHEDULED 258
UNREACHABLE 258
WARNING 258
PENDING 204
Service
FLAPPING 204
UNREACHABLE 199, 204
Status Map 215
Status-Entwicklung (Bericht) 216
Stromversorgung 492
Struktur 241
sudo 156, 178
Swap-Speicher 452
Switch 492
Systeminformationen
Ausfallzeiten 217
Kommentare 217
Systemlast 454

T

tactical overview 222
taktische Übersicht 222
tar 250
TCP-Port überwachen 366
Temperatur *siehe* Netzwerkgeräte, Temperatur
Temperatur überwachen 335, 492
Templates *siehe* Vorlagen
timeperiod 290
timeperiod_name 290
Transmode 456
Trends (Bericht) 216
Trunks 495

U

Ubuntu
  Icinga deinstallieren 138
  Icinga installieren 44
  Icinga-Web 49
  IDOUtils 46
  Nagios deinstallieren 144
  Nagios installieren 104
  NagVis 53, 111
  NDOUtils 106
  PNP4Nagios 49, 108
UDP-Port überwachen 366
Umweltsensoren überwachen 330, 466
Unix überwachen *siehe auch* Linux 458
UNREACHABLE 258
Updates überwachen 356, 467
UPTIME 475
use 270
USV 495

V

Verfügbarkeit (Bericht) 216
Versionsstände 493
virtuelle Maschinen
  KVM 487
  überwachen 486
Visualisierung 525
visudo 156
VNC 487
Vorlagen
  Icinga 23
  PNP4Nagios
    Special Templates 491, 530
  Vererbung 233
VPN 8, 493, 495
VPN-Gateways 493

W

WARNING 258
Warnschwelle *siehe* WARNING
Weboberfläche 195
  Benutzer anlegen/ändern 195
  Authentifizierung 194
  Berichte 216
  CGI-Optionen freischalten 196
  Detailansicht 208, 209
  Detailübersicht 208
  Dienstgruppenübersicht 213
  Gruppenfreischaltung 198
  Konfigurationsobjekte 218
  Maschinengruppenübersicht 211
  Menü 206
  Menüleiste 221
  Nur-Lese-Zugriff 198
  obere Statuszeile 204
  Problemansichten 214, 215
  Statuskarte 215
  Statuszeile 204
  Suche 206
  Systeminformationen 217
  taktische Übersicht 222, 207
  Übersichten zu Dienstgruppen 213
  Übersichten zu Maschinengruppen 211
  Übersichtsseiten 208
  Zugriffsrechte 194
Webserver
  Erreichbarkeit 508
  HTTP-Authentifikation 509
  Server-Prozesse 506
  SSL-Zertifikat 509
  überwachen 403, 505
Weiterleitung 511
Wetterlinien 552
Windows-Server
  CPU-Last 475
  Datenträgerauslastung 479
  Dienste 482
  Erreichbarkeit 472
  Laufzeit 474
  Prozesse 480
  Prozessor-Auslastung 475
  Speicherauslastung 477

überwachen 339, 344, 470
Update-Status 483
Wir 195
WLAN 496
WMI 170
Wrapper-Script 438, 440

X

XMPP 372

Y

yast 114, 140, 146
yum 141, 147

Z

Zeitfenster 290
zentrale Dienste 519
Active Directory 523
DHCP 521
DNS 519
LDAP 522
überwachen 518
Zombie-Prozesse 318, 465
Zusammenarbeit 264, 267
Zustand *siehe* Status
Zypper 114, 140, 146, 467