Nagios-Ökosystem mit Icinga

von Mike Liebsch,
DG-i Dembach Goo Informatik, Köln

Nagios ist ein weitverbreitetes Monitoring-Programm aus dem Open Source, das wegen seiner vielen Module und seiner Flexibilität sehr beliebt ist. Wird das System um intelligente Tools erweitert, kann es in ein richtiggehendes Ökosystem verwandelt werden.

Das bekannte Netzwerküberwachungssystem Nagios, das von einer internationalen Gruppe entwickelt wird, ist weit verbreitet und trotz Konkurrenz durch freie und kommerzielle Software mit gleicher Funktionalität zum Quasi-Standard geworden. Nagios ist kostenlos, bietet viele Zusatzprogramme und kann einfach angepaßt und erweitert werden. Im Jahr 2009 wurde der Fork Icinga abgespaltet. Icingas erklärtes Ziel ist es, neue Features und Erweiterungen schneller zu integrieren und das Monitoringsystem so zu erweitern und zu verbessern, gleichzeitig soll aber die Kompatibilität zu Nagios gewahrt bleiben.

Ein Nagios- oder Icinga-Server alleine ist nicht weiter reizvoll, interessant wird es dann, wenn man das System erweitert, denn das Icinga-Setup läßt sich mit ein paar Addons und Tools in ein komplexes Monitoring-Ökosystem verwandeln. Diese Addons und Tools sollen hier vorgestellt werden.

Konfigurationsmanagement

Normalerweise wird Nagios in den Konfigurationsdateien konfiguriert. Dies sind reine Textdateien, die gesamte Konfiguration kann auch leicht in mehrere Dateien aufgeteilt werden. Aber gerade in größeren, gewachsenen Setups geht die Übersicht über die Dateien schnell verloren und die Strukturierung ist unter Umständen schwierig. Analysiert man nur diese Konfigurationsdateien, kann man deshalb kaum wirklich exakt sagen, wie das Monitoring-System arbeitet und was wie, wann und wie oft geprüft wird. Eine Konfiguration läßt sich aber auch skriptgesteuert aus einer Datenbank oder einer anderen Datenbasis erzeugen. Eine bewährte Vorgehensweise ist das Speichern der kompletten Konfiguration in einem LDAP-Baum. Diese Art der Darstellung reflektiert die Hierarchie der Netze und erlaubt sogar eine Vererbung der Daten. Vorteilhaft ist, daß für LDAP viele Frontends zur Verfügung stehen, unter anderem ldapvi, Apache Directory Studio und phpLDAPadmin. LDAP-Bibliotheken gibt es außerdem für fast jede Programmiersprache, sie können weiterhin leicht in Skripte eingebunden werden.

Problematisch ist, daß Nagios immer seine Konfigurationsdateien braucht. Mit LConf wird deshalb aus einem LDAP-Server heraus automatisch die Konfiguration generiert und in die benötigten Konfigurationsdateien geschrieben. Dieses Perl-Tool bringt sein eigenes, erweiterbares Schema mit, kann periodisch aufgerufen werden und erzeugt einen Ordner *lconf*, in dem die Konfiguration der auszuführenden Befehle, Kontakte und Kontaktgruppen, Dienste und Dienstegruppen sowie zu überprüfenden Hosts strukturiert untergebracht wird.

Diese Konfiguration kann automatisch verteilt und über SVN oder Git versioniert werden. Da der Exportordner nur über die Anweisung *cfg_dir=/path/to/lconf* eingebunden wird, ist eine eigene, nicht generierte Konfiguration weiterhin möglich.

Autoinventarisierung

Für die Autoinventarisierung können die zu überwachenden Hosts und Netzwerkendgeräte mit dem Python-Programm *check_mk* durchsucht werden. Es besitzt eine spezielle Metasprache für die Konfiguration und stellt einen Agenten bereit, der den NRPE-Agenten auf den Servern ersetzt. Auch bei der anderen Netzwerkhardware, die über SNMP gesteuert wird, sucht check_mk automatisch nach überwachbaren Geräten. Ähnlich wie LConf erzeugt check_mk fertige Nagios-Dateien und bietet mit

dem Programm WATO außerdem ein eigenes Frontend, über das auch ein Großteil der administrativen Arbeiten erledigt werden kann.

Webbasierte Konfiguration

Damit auch Nicht-Techniker schnell und einfach ihre Arbeiten durchführen können, wird für ein kleines und vielleicht schon funktionierendes Setup eine webbasierte Konfiguration benötigt. Die Lösung heißt Nagios QL, mit diesem Frontend können Nagios-Objekte angelegt, gelöscht und einfach verwaltet werden. Dabei handelt es sich um eine webbasierte Anwendung mit eigener Datenbank. Das Frontend ist in vielen Sprachen verfügbar und richtet sich an Betreuer kleiner und mittelgroßer Installationen, denen eine Integration anderer Möglichkeiten zu aufwendig erscheint.

Graphing und Reporting

Ein funktionierendes Monitoring mit einfacher Konfiguration ist zwar gut, aber nicht das einzige, was benötigt wird. Weil es auch beim Troubleshooting unterstützen soll, ist eine grafische Auswertung sehr wichtig. Sie muß Trends aufzeigen und die Fehlersuche vereinfachen. Weil diese Option aber häufig in externe Tools wie Munin, Ganglia, Torrus oder mrtg ausgelagert wird, wird zusätzliche Last erzeugt. Speziell auf Netzwerkhardware ist das aber extrem ungeschickt, denn die Geräte werden doppelt oder gar noch öfter geprüft, was die Ergebnisse möglicherweise verfälscht und gerade älteres Netzwerk-Equipment sogar zum Stillstand bringen kann.

Nagios kann die Performancedaten direkt mitliefern, die Geräte müssen nur einmal geprüft werden und die gewonnenen Daten lassen sich direkt weiterverarbeiten. Sie können ausgewertet, visualisiert und in einer Datenbank oder einem Ringpuffer (zum Beispiel RRD) gespeichert werden. Natürlich ist auch eine Ablage als CSV-Datei möglich. pnp4nagios ist ein hübsches, PHP-basiertes Frontend, das Performancedaten übernimmt und in RRD ablegt. Eine Bulk-Verarbeitung, die gerade in großen Setups sinnvoll ist, ist ebenfalls über verschiedene Modi möglich. Das Tool bietet nicht nur eine gute Interface-Integration mit Vorschau-Popups und zoombaren Graphen für ein besseres Troubleshooting, sondern auch verschiedene Export-Funktionen direkt aus dem Webfrontend heraus. Die Konfiguration von pnp4nagios ist relativ einfach und für die meisten Tests werden bereits Graphing-Templates angeboten.

Der NETWAYS Grapher hieß früher Nagiosgrapher und bietet ein AppKit-Frontend (PHP und Adobe Flex) sowie einen Backend-Daemon. Die Daten werden in einer MySQL-Datenbank gespeichert. NETWAYS Grapher erkennt neue Dienste automatisch und kann komplett über das Webfrontend konfiguriert werden.

Die beste Netzwerküberwachung ist wertlos, wenn sie keine Berichte generiert. Sie werden für die Geschäftsführung, das Kapazitätsmanagement und für die Servicepartner benötigt, zudem erwartet das Controlling Verfügbarkeitsreports. Icinga Reporting hilft dem Administrator einer Monitoring-Umgebung beim Schreiben und Verwalten der Reports, wobei die Performancedaten in einer Datenbank abgelegt werden. Das Programm bringt bereits Templates für gängige Reports mit, das Backend beruht auf Jasper. Als Ausgabeformate werden standardmäßig PDF, HTML, Excel, CSV und RTF angeboten. Aufgrund der Speicherung in einer Datenbank können einzelne Werte auch leicht mit einem Skript oder von anderen Werkzeugen wie zum Beispiel OpenOffice.org direkt ausgelesen werden.

```
# echo 'GET hosts' | unixcat /var/lib/
nagios/rw/live
acknowledged;action_
url;address;alias;check_command;check_
period;checks_enabled;contacts;in_
check_period;in_notification_
period;is_flapping;last_check;last_
state_change;name;notes;notes_
url;notification_period;scheduled_
downtime_depth;state;total_services
0;/nagios/pnp/index.php?host=$HOSTNAME$;
127.0.0.1;Acht;check-mk-ping;;1;check_mk
,hh;1;1;0;1256194120;1255301430;Acht;;;2
4X7;0;0;7
0;/nagios/pnp/index.php?host=$HOSTNAME$;
127.0.0.1;DREI;check-mk-ping;;1;check_mk
,hh;1;1;0;1256194120;1255301431;DREI;;;2
4X7;0;0;1
0;/nagios/pnp/index.php?host=$HOSTNAME$;
127.0.0.1;Drei;check-mk-ping;;1;check_mk
,hh;1;1;0;1256194120;1255301435;Drei;;;2
4X7;0;0;4
```

Listing 1: Abfrage aller zu überwachender Hosts

```
# cat query
GET services
Stats: state = 0
Stats: state = 1
Stats: state = 2
Stats: state = 3
# unixcat /var/lib/nagios/rw/live <query
4297;13;9;0
```

Listing 2: Eine in einer Textdatei gespeicherte Abfrage

Core-Erweiterungen

Nicht nur das Konfigurationsmanagement und die grafische Darstellung beziehungsweise das Reporting lassen sich erweitern, auch im Kern kann Icinga durch verschiedene Module ergänzt werden.

Eine Kernelerweiterung einer Icinga- oder Nagios-Installation ist der Nagios Event Broker (NEB). Ist er vorhanden, können core-interne Funktionen aus externen Bibliotheken abgerufen werden. Der Event Broker bietet hierfür Callback-Routinen an, die beim Eintreten spezieller Events im Core aufgerufen werden und sehr vielseitig sind.

Ein weiterer Broker ist Livestatus. Er stellt einen Unix-Socket bereit, über den Informationen direkt aus dem Nagios- oder Icinga-Core ausgelesen werden können, auf Wunsch auch gruppiert und gefiltert. Der Socket kann über den (x)inetd als Netzwerksocket oder über SSH remote angesprochen werden. Die Daten werden wahlweise in den Formaten CSV oder JSON ausgegeben. Der Broker kann auch von alternativen Frontends als Informationsquelle genutzt werden, wobei der Geschwindigkeitsgewinn durch die direkte Kommunikation mit dem Core besonders deutlich wird.

Load Balancing und High Availability

Wird viel mit dem Monitoring-System gearbeitet, reicht irgendwann ein Host

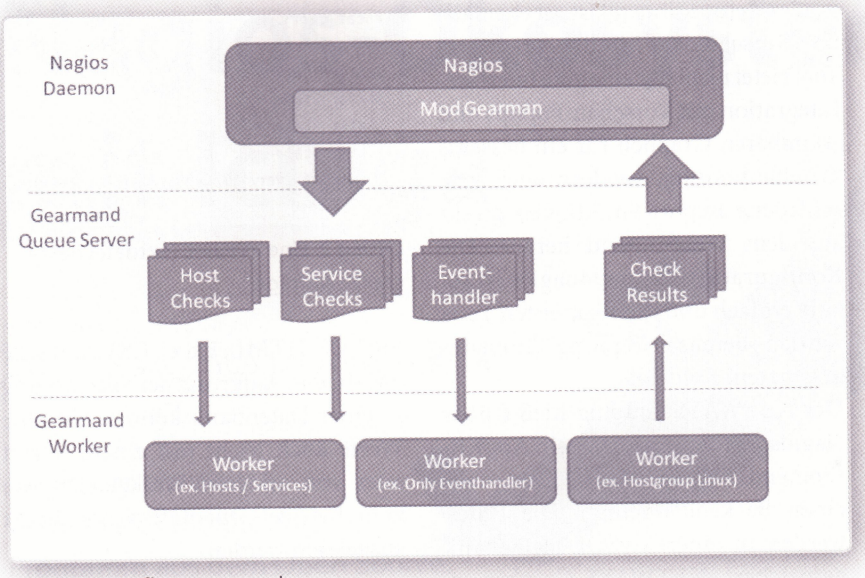
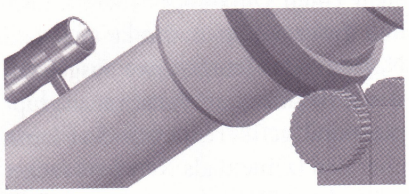


Bild 1: Der Aufbau von mod_gearman

nicht mehr aus. Es wächst die Gefahr, daß die Ausfallsicherheit nicht mehr gewährleistet ist, zu viel Last das System lähmt oder Tests nicht mehr in der gewünschten Zeit ausgeführt werden können. Außerdem kann es sein, daß das Monitoring nicht den Vollzugriff auf alle Netzwerksegmente erhalten soll oder kann. Aus diesem Grund kann eine Trennung in Prüfserver, Frontend-Server und verteilte Poller in besonders schützenswerten oder abgespaltenen Netzen (etwa bei anderen Firmenstandorten) sinnvoll sein. Die Poller sind hierbei Prüfserver, die jedoch nur bestimmte Prüfungen in einem bestimmten Netzwerksegment



vornehmen und die Ergebnisse anschließend gebündelt an die eigentlichen Prüfserver weiterleiten.

Mit *mod_gearman* wird der Core um einen Broker erweitert. Hinzu kommt ein Daemon auf dem Server, der die Aufgaben-Queue hält und verwaltet. mod_gearman erzeugt aus jeder anstehenden Host- oder Diensteprüfung einen sogenannten Workjob, der dynamisch oder fest den konfigurierten Workern zugeteilt wird (Worker sind Prozesse auf einem oder mehreren Servern, die Jobs annehmen, sie analysieren und ausführen. Die Ergebnisse werden dann dem Monitoring Core zurückgemeldet). Die Worker arbeiten asynchron und parallel, was gerade auf Multi-Core-Maschinen für eine bessere Hardwareausnutzung sorgt. Bild 1 verdeutlicht den beschriebenen Aufbau von mod_gearman.

Auch das Graphing kann einem eigenen pnp4nagios-Worker zugeteilt werden. Die Pakete werden AES-256-verschlüsselt zwischen den verschiedenen Komponenten versandt.

Ein weiterer Broker ist Merlin, er bringt seinen eigenen Daemon merlind mit. Aufgrund seines internen Protokolls werden die Checks von einem oder mehreren Masterservern, der oder die für die Kommunikation zwischen den Peers – das sind Server, die selbst prüfen und die Ergebnisse gesammelt an den Master und damit auch wieder an das Monitoring-System selbst zurückschicken – und dem Core zuständig sind, verteilt. Verteilt werden die Checks automatisch, die Verteilung kann aber auch durch Hostgroups gesteuert werden.

Merlin ermöglicht das Abbilden von Redundanz und Load Balancing. Aktuelle Statusinformationen können in einer Datenbank gespeichert werden. Merlin ist aber vor allem für große und besonders lastintensive Setups gedacht und daher nicht unbedingt trivial einzurichten.

Für Nagios und Icinga gibt es mittlerweile eine Vielzahl von Frontends für die Darstellung und die Steuerung des Monitorings, denn das Standardfrontend Nagios/Icinga Classic, das als CGI-Skript in C geschrieben und implementiert wird, ist unhandlich und schwer zu warten und erinnert stark an die neunziger Jahre. Auch läßt seine Geschwindigkeit gerade in größeren Umgebungen zu wünschen übrig. Icinga Web ist ein multisite-fähiges und datenbankbasiertes PHP-Frontend. Leider ist es noch nicht ausgereift und noch nicht für den produktiven Einsatz geeignet. Für viele Setups ist es unzureichend und speziell bei Fehlern eher schwierig zu handhaben. Gerade bei einem größeren Ausfall ist aber ein zuverlässiges, robustes und schnell wieder anlaufendes Monitoring wichtig. Im Falle von Icinga Web müssen hierfür zuerst Monitoring, Webserver und Datenbank(-Cluster)-Setup wieder zuverlässig funktionieren – vorher ist der Administrator blind.

Monitoring in Echtzeit

Das Frontend Ninja ist in PHP geschrieben, sein Backend basiert auf einer Datenbank. Mit Merlin können Distributed Setups leicht integriert werden. Module wie Google Maps und ein Scripting-API sind verfügbar. Allerdings zeigt Ninja die gleichen Probleme wie Icinga Web und wirkt sogar noch etwas träger als die Konkurrenz.

Das in Perl Catalyst geschriebene Frontend Thruk kommuniziert mit den Monitoring-Servern live und liefert daher eine Echtzeitansicht. Es hat viele Sonderfunktionen und ist sehr übersichtlich. Im Vordergrund stehen die Suche, das Filtern und das seitenweise Anzeigen der Informationen (pagenize). Weil Thruk multisite-fähig ist, können verschiedene Backends eingebunden werden, die einzeln oder zusammen darstellbar sind, allerdings nicht korreliert. Thruk unterstützt Nagios, Icinga und Shinken und benötigt keine Datenbank.

Multisite ist eine Bestandteil der check_mk-Suite und ebenfalls in Python geschrieben. Es erlaubt konfigurierbare Ansichten und ist multisite-fähig. Es kommuniziert live mit den Backends

und kann pnp4nagios integrieren. Multisite stellt Performancedaten grafisch dar und verfügt über ein rollenbasiertes Berechtigungssystem. Nach Aussage des Autors ist Multisite noch nicht ausgereift, weshalb es noch nicht als einziges Frontend in Produktivinstallationen taugt.

NagVis visualisiert die eigene Monitoring-Struktur und ist ein leichtes, aber auch komplex erweiterbares Frontend, mit dem sich Schaubilder anfertigen lassen. Es ist extrem flexibel, von Landkarten für Standorte über Visio-Grafiken bis hin zu Fotos von Serverschränken kann alles integriert werden. Dabei werden einzelne Widgets mit einem Nagios-Objekt verknüpft.

Auch Clustersysteme können in Nagios abgebildet werden. Das ist wichtig um zu erkennen, ob ein Dienst – zum Beispiel ein Webshop – überhaupt noch läuft, und wenn ja, wie schnell. Das Funktionieren des Systems ist schließlich entscheidender für den Geschäftsbetrieb als der Ausfall einer einzelnen, redundant ausgelegten Komponente. Mit dem Programm Business Process View können Ausfälle simuliert und die Auswirkungen auf das Gesamtsystem geprüft werden. Das hilft beim Planen der Verfügbarkeit, der Verbesserung des Risiko-Managements und der Überwachung der Service Level Agreements (SLAs) gegenüber Kunden und Zulieferern.

Fazit

Die Bedeutung von Monitoring für Unternehmen wird häufig unterschätzt: Ohne Monitoring gibt es kein Risikomanagement und keine Notfallplanung. Monitoring ist auch die Grundlage jedes Verfügbarkeitsmanagements und vor allem wichtig für das Troubleshooting. Das entscheidende Kriterium für die Bewertung eines Monitoring-Systems ist, welche Komponenten geplant und getestet sind.

Insgesamt gesehen ist Nagios/Icinga ein sehr leistungsfähiges Monitoring-System und es gibt für fast jeden Verwendungszweck eine Erweiterung oder ein Modul, von denen Unternehmen profitieren können. Hinzu kommt, daß sich bestehende Lösungen wie etwa Data-Warehouse-Datenbanken, Management- und Ticket-Systeme anbinden lassen, um Monitoring-Daten sowohl zu liefern als auch zu nutzen, wodurch ein wirkliches Ökosystem entsteht. ◆

Links

Nagios: http://www.nagios.org
Icinga: http://www.icinga.org
NagiosQL: http://www.nagiosql.org
mod_gearman: http://labs.consol.de/lang/de/nagios/mod-gearman/
Netways: http://www.netways.org (LConf, Netways Grapher v2)
Mathias Kettner: http://www.mathias-kettner.de (Livestatus, check_mk, Multisite, wato)
op5: http://www.op5.com (Ninja, Merlin, Plugins)
pnp4nagios: http://www.pnp4nagios.org
Thruk: http://www.thruk.org