

Tuesday, March 17, 2009

iTerm: Backspace in Screens

Seit kurzem nutze ich iTerm unter MacOS um meine Command-Line Tasks zu machen. Es gefällt mir sehr gut; Fullscreen, recht gut erweiterbar. Was mich aber störte war dass die Backspace-Taste nicht funktionierte wenn auf der Gegenseite in einem screen gearbeitet wurde. Warum auch immer.

Eine kurze Suche fand diesen Artikel: Einfach in den Properties Backspace auf Hex Code 8 umsetzen und schon tuts

Posted by rince in Tutorials at 08:14

Nagios: Testen verschiedener Netzwerk-Interface am selben Rechner

Wir haben eine recht große Webserver-Infrastruktur aufgebaut, wo jeder Rechner mehrere Interface hat - eines für administrative Zwecke, eines für Webserver und so weiter. Mit Nagios kann man so etwas schlecht abbilden; es kennt "nur" eine IP-Adresse pro Rechner. Da ich aber nicht mehrere Rechner definieren wollte um zum Beispiel HTTP und SSH checken zu können bin ich auf einen Trick verfallen.

Bei unseren Rechnern sind der erste, zweite und der vierte Teil des Quadrupels immer gleich - nur das dritte Teil ändert sich, je nachdem in welchem Netz das Interface hängt.

Also hat zum Beispiel ein Webserver die IP 10.10.5.20 im Admin-Netz und 10.10.10.20 im Webserver-Netz. Nagios wurde definiert mit der IP-Adresse des Webserver-Bereiches (da sind einfach mehrere Checks nötig :), so dass ein check_ssh auch auf 10.10.10.20 gehen würde - wo aber gar kein sshd lauscht.

Also habe ich einen Wrapper geschrieben, der die IP-Adresse bekommt, umschreibt und dann den Check selbst ausführt. Das geht recht einfach:

```
#!/bin/bash
```

```
HOST=$1
```

```
NEWHOST=`echo $HOST | /bin/awk -F"." '{print $1 "." $2 ".10." $4 }`
```

```
/usr/local/nagios/libexec/check_ssh -H $NEWHOST
```

Nun geht der ssh-Check auf das administrative Interface und es wird sauber dargestellt.

Posted by rince in Tutorials at 08:06

Saturday, April 26. 2008

Nagios: Nicht nur Monitoring sondern auch Reporting...

So, der letzte Teil dieses Tutorials ist hier: Reporting.

Nagios selbst ist hauptsächlich für Monitoring da. Reporting ist aber etwas ganz anderes. Reporting ist dafür da, über einen längeren Zeitraum einen Dienst zu beobachten und Trends zu erkennen, zu sehen ob zum Beispiel ein Dienst immer langsamer wird (zuwenig Ressourcen vorhanden?) oder ein Rechner regelmäßig ausfällt im Gegensatz zu anderen Maschinen.

Nagios bietet sowas auch an, wenn auch "relativ" rudimentär. Zusammen mit einer MySQL-Datenbank als Backend und/oder munin kann man vermutlich noch mehr bauen; aber dafür muss ich mich selbst erstmal schlau machen ob nagios an Munin was schicken kann

Was Nagios aber kann sind Trends oder Verfügbarkheiten einzelner Dienste und Rechner (oder entsprechender Gruppen) anzuzeigen. Das ist nicht allzu komfortabel und auch in der Form nicht Management-geeignet, aber daraus kann man "echte" Reports basteln. Es wird eine Übersicht gegeben wie oft(oder lange) der Host/Dienst im jeweiligen Status war und auch eine Durchschnitt pro Host/Service. Eine gute Möglichkeit einen Überblick zu bekommen. Man kann sich auch die letzten x Alarme anzeigen lassen; auch gut um Trends erkennen zu können.

Alles in allem gibt es kein Reporting für Management, aber zum Nachschauen und überprüfen seit dem letzten Programmstart von Nagios ist es durchaus ausreichend. Wobei Programmstart wirklich "Start von Nagios" meint und nicht ein HUP zum Ändern der Konfiguration.

Posted by rince in Tutorials at 22:51

Friday, April 25. 2008

Nagios: Darstellung der einzelnen Tasks?

Wenn man Nagios soweit installiert und konfiguriert hat dass man viele Dienste monitored kommt man irgendwann dahin dass man feststellt dass der Monitor bzw. die Auflistung zu klein ist um alles zu sehen.

Das ist soweit korrekt.

Mit der richtigen Konfiguration kann man allerdings viele Services zusammenfassen und nach dem Umbrella-Prinzip sich den "generellen" Überblick geben lassen.

Die Rede ist von Service- und Hostgruppen. Für mich sind die Servicegruppen wichtiger, weil die Hostgruppen zwar für Sysadmins interessant sind, aber für Applikationsadministratoren die Services.

Das Prinzip ist wie folgt: Man fasst Rechner oder Dienste zu Gruppen zusammen, weil sie entweder dasselbe monitoren oder dieselbe Zielrichtung haben. Pro Applikation haben wir zum Beispiel zwei Rechner in der Produktionsumgebung und zwei Rechner in der Testumgebung. Ich kann dann einerseits Hostgruppen bauen pro Produktionsumgebung, aber auch pro Applikation. Dann sehe ich den Gesamtstatus der Rechner im Summary; wieviele Services es pro Rechner gibt die gemonitored werden aber auch ob alle Services OK sind. Auf einen Blick habe ich so einen Überblick über die Rechner.

Bei Servicegruppen geht es genauso: Wenn ich weiss dass ich bei einer Applikation 40 Schnittstellen habe die ich überwache will ich als Überblick nicht alle Schnittstellen sehen - ich will die Aussage "alles ist OK".

Dies geht mit dem Servicegruppen. Als members muss ich da den Host und den Check angeben der dazugehört; daraus generiert dann das Web-Frontend eine Übersicht. Der Vorteil ist dass ich dann verschiedenen Nutzern auch nur die Sachen zeigen kann die sie sehen sollen - die Webadmins sollen nicht unbedingt sehen ob der Mailservice tut. So bleibt das Frontend letztendlich auch gut übersichtlich. Ich finde es hervorragend weil ich so auch für eine Fröhschicht aufbereiten kann dass sie mit einem Blick sieht ob alles in Ordnung ist oder nicht - und wenn eine Servicegruppe rot sein sollte kann man auf sie klicken und sieht dann die einzelnen Services und kann sehr schnell erkennen was Probleme bereitet.

Ich bin mir momentan unsicher ob ich dazu Beispielgrafiken machen soll - besteht daran Interesse? Dann würde ich diese noch bauen und einfügen...

<http://wiki.debian.de//doku.php?id=nagios:group>

Posted by rince in Tutorials at 22:51

Thursday, April 24, 2008

Nagios - Aktiv oder passive Checks?

Nachdem ich jetzt endlich weiss welche Hosts ich monitoren will und welche Dienste ist die Frage auf welchem Wege das stattfinden soll.

Generell gibt es drei "generische" Möglichkeiten:

- Der Nagios-Server selbst führt selbst Tests aus
- Der Nagios-Server "bittet" den entfernten Rechner einen Test auszuführen
- Ein Test wird ausgeführt und Nagios "nur" das Ergebnis mitgeteilt

Die ersten beiden Methoden sind sogenannte Aktive Checks - Nagios selbst führt die Checks aus oder stellt einen Task an der den Check ausführt.

Der letzte Test ist ein sogenannter Passiver Check; Nagios selbst ist nicht für den Test verantwortlich sondern verarbeitet nur das Ergebnis.

Gerade in Netzwerken wo es Firewalls gibt oder Sicherheitsbestimmungen ist die Wahl der Checks durchaus interessant. Nicht alle Checks kann der Nagios-Server selbst machen; ob auf entfernten Rechnern die Festplatte überprüft zum Beispiel kann er nicht einfach sehen.

Daher gibt es diese zweite Kategorie - für Router oder andere Systeme kann man snmp einsetzen, aber auch den Befehl via ssh auf dem entfernten Rechner auszuführen ist eine Option. Für Windows-Rechner gibt es auch ein entsprechendes Programm welches auch als Service installiert werden kann.

In allen Fällen bekommt Nagios vier wichtige Angaben gesagt:

- einen Zeitstempel (Epoch)
- einen Nagios-internen Befehl
- Den Host von dem der Check kommt
- Der Status des Dienstes (0 = OK, 1 = Warning, 2 = Critical, 3 = Unknown)
- Ein freier Text den der Check ausgeben kann

Mit Hilfe des Status hat Nagios seine Aktionen definiert - bei Critical-Meldungen im Melde-Zeitrahmen gibt es zum Beispiel eine Mail an die Beteiligten; wenn der Dienst wieder OK ist, ebenfalls. Es gibt auch die Möglichkeit dass Dienste eine Zeitlang nicht überprüft werden - wenn sie für längere Zeit ausgefallen sind zum Beispiel.

Zu wissen, welche Daten Nagios dringend braucht ist wichtig wenn man eigene Plugins schreibt (weil die Standard-Plugins nicht helfen oder weil man alles für sich anpassen möchte); aber auch wenn man passive Checks nutzen will. Bei passiven Checks werden Der Host, der Service, die Statusnummer und der Freitext üblicherweise mit Hilfe des programm NSCA an den Nagios-Server geleitet. Die Checks müssen natürlich definiert sein für den Host, damit die Nachrichten korrekt zugeordnet werden können.

Passive Checks haben aber auch einen weiteren Vorteil: Man kann nagios anweisen zu überprüfen wann der letzte passive Check eines Dienstes passiert ist - und wenn diese Zeit zuende ist wird ein aktiver Check durchgeführt. Diesen aktiven Check kann man dann so konfigurieren dass der Dienst sofort auf Critical gesetzt wird. Ich nutze diese Methode gerade um einen Monitor zu überwachen - wenn nach 125 Sekunden keine neuen Daten kamen meldet sich nagios und sagt mir dass der Monitor nicht mehr tut. Sehr sinnvoll, besonders abends wenn keiner mehr vor Ort sein möchte.

Passive Checks haben aber auch noch einen anderen Vorteil; gerade bei uns. Unsere Netze sind sehr sauber voneinander getrennt - Produktions, Test, Entwicklungsumgebungen, Internet usw. Und unsere Monitoring-Station muss von allen Netzen auch die Daten ja bekommen. Das könnte durchaus zu Problemen mit Firewalls führen.

Was aber unkritisch ist ist wenn die Rechner zum Monitoring hin eine Verbindung aufbauen und nicht andersrum. Daher

sind die passiven Checks sehr angenehm - mit Hilfe von nsca macht der zu Ä¼berprÄ¼fende Rechner auf und nicht der Nagios-Server.

Posted by rince in Tutorials at 20:50

Monday, April 21. 2008

Nagios - die Idee und Implementierung (auch Konfigurationsübersicht genannt)

Die Konfiguration von nagios ist durchaus komplex, aber wenn man es richtig anstellt ist es einfach, nach und nach alles einzurichten.

Es gibt verschiedene Sektionen die wichtig sind:

- Hosts
- Dienste(Services)
- Check-Kommandos
- Zeitperioden
- Kontakte, an den Benachrichtigungen gehen sollen.

Es gibt noch mehr, aber davon erzähle ich später.

Hosts: das sind die Geräte die überwacht werden sollen - das können Rechner sein (PCs, Unix, Windows...) aber auch Router oder Switches; oder auch Thermometer, sofern sie via tcp/ip abfragbar sind; eventuell snmp sprechen oder etwas ähnliches. Bei der Host-Konfiguration muss man mindestens eine IP-Adresse und/oder Hostnamen angeben, damit eindeutig geklärt ist welcher Rechner da gemonitored wird.

Dienste: sind die Programme oder Prozesse die von nagios überwacht werden. Jeder Dienst braucht als Mindestangabe den Host auf dem er laufen soll - sonst hätte die Definition auch keinen Sinn. Innerhalb eines Dienstes wird der Name (und Alias) definiert und auch welches Check-Kommando ausgeführt werden soll um den Dienst zu überwachen.

Check-Kommandos: Das sind die Kommandos die ausgeführt werden um den Dienst zu überwachen. Während beim Dienst noch nach Nagiosart Variablen übergeben werden (wie der Hostname des zu prüfenden Rechners) wird hier das Kommando definiert, inklusive dem Aufruf. Quasi eine Übersetzung von Nagios-Konfiguration in Überprüfungs-Konfiguration.

Zeitperioden: Manche Dienste sollen nur zu einer bestimmten Zeit oder an bestimmten Tagen überhaupt geprüft werden. Diese können definiert werden - zum Beispiel kann gesagt werden dass es unkritisch ist, wenn die Temperatur am Wochenende auf 40°C steigt; das könnte ja in Ordnung sein.

Kontakte: Falls es mal Probleme gibt, muss natürlich auch jemand benachrichtigt werden. Diese Personen werden in den Kontakten definiert - inklusive der Möglichkeit, wie die Nachrichten weitergegeben werden.

Für Rechner, Dienste und Kontakte gibt es auch noch Gruppierungsmöglichkeiten; so dass man zum Beispiel 10 verschiedene Rechner zu einer Gruppe zusammenfassen kann (Kunde A, Kunde B...). Genauso kann man Dienste in einer Gruppe zusammenfassen - Dienste, die nicht gleich sind bzw. nicht dieselben Checks haben, aber dasselbe Ziel. Und Kontaktgruppen ermöglichen es, Nutzer zusammenzufassen, wenn jemand benachrichtigt werden soll.

Für alle diese Konfigurationsmöglichkeiten gibt es sogenannte Templates - Vorlagen. In den von Nagios mitgelieferten Vorlagen sind alle möglichen Parameter bereits gesetzt und meistens auch sinnvoll. Ich habe allerdings festgestellt dass es für mich sinnvoller ist zusätzliche Gruppen (zum Beispiel bei den Diensten) zu bauen. Ich habe dann für den Dienst ein Template gebaut und musste dann pro Schnittstelle die ich überwachen wollte nur noch sagen wie sie heißt und welcher Port überwacht werden soll. Alles andere wurde durch den Gebrauch einer Vorlage dann standardisiert.

Ganz wichtig: in der Konfiguration kommt immer wieder der Parameter "Alias" vor. Das ist nicht nur der Alias für den Dienst/Rechner/was auch immer, sondern die Kurzbezeichnung des Dienstes wie er auf der Webseite angezeigt wird.

Wie hängt das nun miteinander alles zusammen? Ich fange mal oben an.

Ich habe eine Anzahl von Rechnern, von denen will ich einerseits wissen ob sie leben und ob sie Webserver spielen. Diese Überprüfungsart ist für mich aber nur in den Standard-Arbeitsstunden (9-17 Uhr) wichtig. Und auch nur ich soll

dabei erstmal informiert werden.

Die ganzen Konfigurationen habe ich mal ins Wiki gelegt; das ist übersichtlicher.

Generell gilt: Wo die einzelnen Definitionen stehen ist fast unabhängig; in der nagios.cfg kann man ganze Verzeichnisse angeben in dem die Konfigurationsdateien zu finden sind. Diese müssen nur die Endung .cfg haben. Ob man jetzt sich das ganze nach Diensten, Rechnern oder anderer Logik folgend strukturiert bleibt jedem Selbst überlassen.

Ich habe für einige Applikationen einzelne Config-Dateien weil ich genau weiss dass ich nur dort dann ändern muss. Andere dienste (Web) sind bei den Hosts selbst definiert.

Was muss ich also definieren? Wenn ich ganz von vorne anfangen will: Eine Vorlage für die Rechner. in dem schreibe ich (weil ich faul bin nur hinein, wie er heisst (host_name und alias) und die IP-Adresse (address)). Ich kann noch viel mehr definieren, aber anfangs will ich das gar nicht - ich will ja erstmal sehen was daraus entsteht.

Als nächstes frage ich mich: Was will ich auf den Rechner eigentlich monitoren? Also, erst einmal will ich generell ein Ping absetzen können, dann will ich einmal Web und einmal Mail testen können. Also muss ich dafür jeweils ein Kommando definieren - pro Rechner kann ich dann einen Dienst generieren der dieses Kommando nutzt. Wenn man sich die commands.cfg im Wiki genauer anschaut, sieht man das es dort Variablen gibt - \$HOSTADDRESS\$ zum Beispiel. Genau deswegen gibt es die Trennung zwischen Dienste und Check-Kommandos: beim Aufruf des Dienstes wird die Variable \$HOSTADDRESS\$ mit Inhalt gefüllt. Damit kann dieselbe Check-Kommandodefinition für verschiedene Dienste genommen werden; bei Pings zum Beispiel kann es interessanter sein; LAN-Strecken anders zu monitoren als WAN-Strecken.

Als nächstes definiere ich die Dienste (Services) pro Rechner, die ich abfragen will. Wie man in der Beschreibung sieht, soll serverA nur Web machen, serverB Web und Mail, serverC nur Mail.

Da mehr als ein Rechner die Dienste hat baue ich Service-Gruppen. In diesen Gruppen definiere ich den Namen der Gruppe und die Mitglieder - wenn ich weitere Rechner dazunehme zu dem Dienst, muss ich sie nur der Gruppe hinzufügen, mehr nicht.

Dementsprechend müssen auch die Dienste geschrieben werden. Wobei ich hier nur Beispiele mache und deswegen die eigene service.cfg dafür nehme.

Jetzt könnte auch klarwerden, wie bei den Checks das \$HOSTADDRESS\$ gefüllt wird und wie dann die Checks aufgerufen werden. \$USER1\$ ist übrigens ein Makro das vorher definiert wurde - da muss man den langen Pfad bis zum Check-Kommando nicht immer ausschreiben.

Zu guter Letzt habe ich ja gesagt die Dienste sollen nur zu bestimmten Zeiten getestet werden. das mache ich in der timeperiod.cfg

Auch wenn es unübersichtlich aussieht anfangs - wenn man sich einmal klargemacht hat wie man monitoren muss und wie, ist diese Konfiguration recht übersichtlich.

Posted by rince in Tutorials at 18:50

Sunday, April 20. 2008

Nagios - die Installation unter *nix

Die Installation von Nagios ist eigentlich recht einfach:

Entweder nutzt man eine Linux-Distribution die Nagios in seinem Paket-Management-System hat, das heisst entweder vorcompilierte Programmpakete (wo auch die Installation in die richtigen Verzeichnisse gleich stattfindet und die notwendigen User angelegt werden) oder eine Möglichkeit wie bei gentoo, die Programme selbst zu compilieren auf der eigenen Maschine. Das können sein:

- Debian
- RedHat / Fedora Core
- Gentoo
- Novell Linux

Oder man geht auf die Download-Seite von Nagios und lädt den Sourcecode herunter.

Bei beiden Methoden ist allerdings wichtig zu wissen dass man mindestens zwei Pakete braucht: Nagios (den Server) und Nagios-Plugins. Der Server selbst ist nur dazu da, das Framework dazustellen welches die Plugins nutzt um das Monitoring zu machen.

Die Plugins müssen auf jedem System installiert werden welches über sie gemonitored werden soll - zum Monitoren selbst gibt es mehrere Möglichkeiten, auf die ich später eingehe (snmp, ssh, nsca, nrpe). Es macht Sinn, sie auch auf dem Rechner zu haben der den Server betreibt, aber es ist kein muss.

Es macht Sinn, für Nagios einen eigenen User anzulegen mit seiner eigenen Gruppe; dann bleibt alles was nagios macht auch unter seiner Verantwortung und in seinen Verzeichnissen. Nach dem Auspacken (ich werde eine Beispielinstallation in ein Wiki legen zum Zeigen) nutzt man "configure" um die wichtigsten Einstellungen zu machen - unter welchem User soll nagios laufen, unter welchem User soll das Kommandointerface laufen (der Web-User meistens), wo liegt die Apache-Konfigurationsdatei für die Webseiten und so weiter. Ich setze da das Prefix (/usr/local), den User und Gruppe (beides nagios) und dann den Webuser für die Kommandos. Ein "make" und ein "sudo make install-all" installiert dann nagios in /usr/local, so dass der Server an sich fertig ist.

Was dann noch fehlt, ist die Konfiguration des Webservers (Apache) und die Init-Scripte. Dies werde ich morgen updaten

Auf dieser Seite werde ich die Installation Schritt für Schritt zeigen. Ich hoffe es ist dann auch verständlich

Posted by rince in Tutorials at 16:49

Nagios - eine Einführung

Ich habe dieses Blog etwas schleifen lassen. Aber das soll sich wieder ändern; ich werde es wieder mit Leben füllen. Anfangen will ich dabei mit einem neuen (und gleichzeitig alten) Projekt - Systemmonitoring.

In der Firma haben wir ein Monitoringsystem, welches inzwischen recht alt ist. Es funktioniert wunderbar, wir haben es auf unsere Bedürfnisse angepasst. Aber in den letzten Monaten haben wir ein paar Probleme festgestellt. Wir könnten natürlich den Hersteller bitten sich die Probleme anzuschauen - aber deren Reaktion wird sein uns zu sagen, wir sollen auf eine aktuelle Version upgraden. Dafür müssten wir auch alle unsere Scripte anpassen. Dann können wir aber auch gleich was neues nehmen...

Da ich im vorigen Job schonmal nagios ausprobiert hatte bin ich der Meinung es hat nach ca. 6 Jahren eine neue Chance verdient. Dann gleich nagios3 und schauen was die Neuigkeiten sind.

Warum Nagios? Nun, es gibt mehrere Gründe für mich dafür:

Ich kenne einige Leute die nagios nutzen, und zwar auch große Installationen. Das heisst es scheint performant

genug zu sein

Es ist - wenn man es einmal verstanden hat - recht einfach, auch komplexe Aufgaben damit abzubilden

Es ist Open Source. Das heisst ich kann notfalls jemanden fragen der C/C++ kann ob der Sourcecode das tut was ich will Alternativ kann ich meine eigenen Plugins schreiben um Checks zu machen

Ich kann es nicht nur unter Unix nutzen sondern auch unter Windows. Zumindest die Checks können dort laufen, was für uns durchaus wichtig ist.

Die Weboberfläche ist - wenn man es richtig macht - übersichtlich und auch Management-geeignet. Gerade in den letzten Tagen habe ich schätzen gelernt mit einem Blick sehen zu können ob und welche Services kaputt sein könnten

Ich kann Abhängigkeiten bauen - wenn der Switch kaputt ist sind logischerweise alle Services dahinter auch nicht erreichbar; also brauche ich dafür keine Meldungen extra; maximal auf der Webseite aber bitte nicht per Mail.

Die Config ist am anfang zwar verwirrend, aber je mehr man mit Templates arbeitet/arbeiten kann umso einfacher werden Spezialanforderungen (bei uns heisst das Gefälligkeit

Ein paar Sachen fehlen mir selbst oder ich habe sie noch nicht richtig gefunden - Anbindung an Munin in einer Weise wo man nicht von munin die Daten bekommt sondern Nagios sie munin gibt, aber auch sowas werde ich in den nächsten Wochen evaluieren weil man dann wunderbare Grafiken bekommt über Langzeitverhalten. Durchaus ein spannendes Thema.

Bei uns gab/gibt es mit diesem System zwei Ziele:

Ein Monitoringsystem für alle Welten haben: Bisher haben wir das alte Monitoringsystem einmal in der Windows- und einmal in der Unix-Welt. Da sich bei uns die Struktur ändert (ein Servicedesk aka First Line of Defence vor allen anderen) soll alles auf einem System zu sehen sein.

Wir brauchen etwas wo man mit einem Blick erkennt ob es eine Störung gibt und wo sie liegt. Und erst wenn man den "kaputten" Host oder Service anklickt soll man sehen welcher Teil genau kaputt ist.

In den nächsten Tagen oder Stunden werde ich dazu mehr schreiben.

Posted by rince in Tutorials at 15:47