

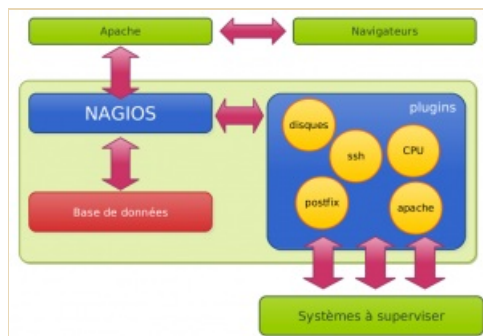


## Supervision domestique avec Nagios

Le 30 octobre 2008 à 00:42.

Lorsque l'on gère un petit réseau domestique avec une ou deux machines, on se retrouve malgré tout avec des problèmes de "grands" comme la nécessité d'être prévenu le plus vite lorsqu'un service tombe, et ce de la manière la plus automatisée possible. C'est à cette problématique que répond Nagios .

### Présentation



Nagios [1], qui s'appelait précédemment NetSaint, est un outil de supervision pour parc de serveurs. Il permet d'auditer en permanence des machines, des services sur ces machines, de recevoir des alertes en cas de problème et de disposer d'un tableau de bord de l'état du système à un moment donné.

Nagios s'architecture autour d'un moteur écrit en C chargé de la planification des différents audits à lancer à travers de le réseau et de l'agrégation des résultats dans une base de donnée. L'acquisition en elle-même est déléguée à une impressionnante librairie de greffons qui répondent à tous les besoins ou presque. Des modules qui sont souvent écrits en perl ou en bash, très simple à modifier et tout autant à imiter.

Le tout est contrôlable à travers une frontal Web basée sur le protocole CGI et accessible via n'importe quel serveur HTTP comme Apache.

### Installation

Sous Mandriva il suffit d'installer le paquet `nagios-www` pour que le reste vienne avec par jeu de dépendance. Si vous avez gardé la configuration standard de Mandriva côté apache, l'installation a ajouté un fichier de configuration spécifique en `/etc/httpd/conf/webapp.d`. Pour les autres ce qui est à rajouter dans votre apache doit ressembler à cela :

```
ScriptAlias /nagios/cgi-bin /usr/lib/nagios/cgi
```

```
<Directory /usr/lib/nagios/cgi>
```

```
Options ExecCGI
order deny,allow
deny from all
allow from 127.0.0.1
```

```
</Directory>
```

```
Alias /nagios /usr/share/nagios
```

```
<Directory /usr/share/nagios>
```

```
Options None
order deny,allow
deny from all
allow from 127.0.0.1
```

```
</Directory>
```

Le `ScriptAlias` permet d'accéder au CGI de Nagios, et l'`Alias` au dossier des éléments WEB. Tel quel l'`Alias` ne laisse passer que les utilisateurs locaux. A vous de modifier cela pour ajouter de l'authentification [2], du SSL, etc.

Ensuite pour que cela fonctionne, nous allons faire une très vilaine chose, à savoir virer l'authentification du côté CGI de nagios. Pourquoi ? Disons qu'il n'y a pas grand intérêt à authentifier un service qui n'est accessible que localement ou alors est encrypté et authentifié. Du moins pour un usage domestique aucun. Pour faire cela, il faut simplement modifier la configuration `/etc/nagios/cgi.cfg` et mettre à 0 la valeur de `use_authentication` .



Ceci fait, nous pouvons redémarrer apache, puis démarrer le service `nagios` et enfin aller faire un tour sur le serveur à l'url `/nagios` pour voir la plus horrible interface que le monde ait connu depuis le WEB 0.1. Heureusement il est possible d'arranger cela en installant un look un peu plus moins-pire avec le paquet `nagios-theme-nuvola` .

Comme vous le voyez, nagios fonctionne "out of the box" avec une série de service (cpu, disques) en écoute de la machine locale. Voyons maintenant comment aller plus loin.

## Configuration

Alors je vous préviens, le paramétrage de cet outil peut se révéler être une véritable expérience Kafkaïenne mais une fois réalisé, on dispose de quelque chose de fonctionnel, et pour longtemps.

Pour débiter, prenons un exemple simple. Imaginons que nous ayons un serveur nommé `gaston` et que nous voulions auditer la bonne marche de son service HTTP.

Nous allons commencer par créer un fichier `/etc/nagios/servers/gaston.cfg` chargé de décrire le serveur

```
define host{
    use                linux-server
    host_name          gaston
    alias              Le serveur Gaston
    address            192.168.0.155
}
```

*/etc/nagios/servers/gaston.cfg*

Notre `host` est de type `linux-server` qui un template nagios qui porte bien son nom. `Alias` est le nom "humain" du serveur, il peut contenir des espaces mais c'est à peu près tout. Pas d'accents, pas de parenthèses, bref un véritable ascétisme syntaxique tendance ethno-centré US.

Autre détail, ne pensez même pas à formater le code à votre convenance. Par exemple, mettre la première accolade à la ligne entraînera un refus brutal de démarrage...

Pour tester notre configuration, plutôt que de relancer tout de suite le service `nagios`, nous pouvons d'abord la tester à la main :

```
root# nagios -v /etc/nagios/nagios.cfg
Nagios 3.0.3
Copyright (c) 1999-2008 Ethan Galstad (http://www.nagios.org)
Last Modified: 06-25-2008
License: GPL

Reading configuration data...

Running pre-flight check on configuration data...

Checking services...
Checked 8 services.
Checking hosts...
Checked 1 hosts.
Checking host groups...
Checked 1 host groups.
...
Total Warnings: 0
Total Errors: 0

Things look okay - No serious problems were detected during the pre-flight check
root#
```

Là ça passe où ça casse mais au moins cela donne la ligne où ça plante. Une fois que tout est correcte, on peut redémarrer proprement le service `nagios`.

Il suffit ensuite d'aller sur l'URL de nagios, de faire un refresh pour voir un nouveau host apparaître. Dans un premier temps il apparaît en gris car la mise à jour des status est encore en file d'attente. Mais au bout de quelque temps cela devrait passer au vert.

Deuxième étape, ajouter un service à `gaston` en éditant à nouveau notre fichier `gaston.cfg` :

```
define service{
    use                local-service
    host_name          gaston
    service_description  Audit du site web de Gaston
    check_command      check_http_text!Ceci est le site Web de Gaston
}
```

*/etc/nagios/servers/gaston.cfg*

L'idée de ce service est de vérifier si le serveur HTTP fonctionne sur `gaston` et que la page renvoyé contient la chaîne de caractère `Ceci est le site Web de Gaston`. En l'état cela ne marchera pas car il va encore falloir créer la commande Nagios `check_http_text`.

En effet Nagios est fournit avec une foultitude de plugins permettant de tester un grand nombre de services. Ces plugins, qui sont de simple exécutables stockés dans `/usr/lib/nagios/plugins`, ne sont pas reconnus directement. Il faut préalablement les déclarer dans le fichier `/etc/nagios/objects/commands.cfg`.

Ceci dit, là, nous cherchons la difficulté car une grande partie des plugins de base sont déjà configurés en commandes directement utilisables et nous aurions pu par exemple utiliser la commande `check_http`, qui ne teste que la présence d'un serveur web, à la place `check_http_text`.

Mais pour nous faire la main, disons que nous avons absolument besoin de quelque chose de plus spécifique. Pour ajouter cette nouvelle commande, nous allons donc étendre le fichier `commands.cfg` :

```
define command{
    command_name    check_http_text
    command_line    $USER1$/check_http -H $HOSTADDRESS$ -R "$ARG1$"
}
```

à la fin de `/etc/nagios/objects/commands.cfg`

La commande `check_http_text` va invoquer le plugin `check_http` avec comme premier paramètre l'adresse IP du serveur, et en second paramètre une expression régulière à vérifier dans la page de garde. Ce paramètre correspond à ce qui se trouve après le point d'exclamation dans le service que l'on a écrit un peu plus haut (`check_http_text!Ceci est le site web de Gaston`). A noter que si nous avons créé une commande avec plusieurs paramètres, nous aurions rajouté des `$ARG2$` ou `$ARG3$` et aussi rajouté dans le service des paramètres précédés de leur point d'exclamation. Oui je sais, c'est totalement idiot comme formalisme...

Maintenant, on redémarre le service `nagios` et là sur la page WEB devrait apparaître notre nouveau service.

A ce stade, vous avez presque toutes les billes pour configurer Nagios. C'est long, c'est fastidieux, on se trompe souvent mais on y arrive, du moins pour les services locaux et les services distants mais publics (ex. `http`, `ssh`, etc.). Reste le cas des services locaux, d'un serveur distant.

## Audit de services internes à distance

Imaginons que cette fois nous voulions connaître l'espace disque sur le serveur `gaston`. Pour réaliser une telle chose nous allons devoir passer par une couche de transport prise en charge par le paquet `nrpe`.

NRPE est un petit serveur qui va se mettre en écoute sur un port sur la machine distante et répondre aux requêtes de votre serveur Nagios. Lorsqu'il reçoit une demande, il va exécuter un plugin local et transmettre sa réponse à Nagios. Conséquence directe, `nrpe` a son propre fichier de configuration qui n'a rien à voir avec celui de `nagios`. Donc si vous pensiez pouvoir paramétrer cela à distance c'est raté.

Pour auditer à distance l'état des disques du serveur `gaston`, nous allons devoir commencer par y installer le service `nrpe` avec le paquet `nrpe-plugin`. Et sur la machine qui exécute Nagios, nous allons installer le plugin ET la commande `nrpe` provenant du paquet `nagios-check_nrpe`.

Sur la machine `gaston`, nous allons éditer le fichier `/etc/nagios/nrpe.cfg` et localiser la ligne contenant `command[check_disk1]`. Vous commencez à comprendre, cette ligne publiée via `nrpe` un paramétrage spécifique du plugin `check_disk` qui pour l'instant utilise la partition `/dev/hda1`. Changer la référence de la partition pour, par exemple, auditer le répertoire `/var` en `/dev/sda3`. On remplace donc `hda1` par `sda3`. Le reste ne change pas. Ensuite il faut sauvegarder et redémarrer `nrpe`.

Maintenant nous allons ajouter un nouveau service à `/etc/nagios/servers/gaston.cfg` :

```
define service{
    use                local-service
    host_name          gaston
    service_description Etat de la partition /var
    check_command      check_nrpe!check_disk1
}
```

`/etc/nagios/servers/gaston.cfg`

Notez le paramètre `!check_disk1`, il va être transmis à la commande et remplacer l'argument `$ARG1$` avant que le plugin `nrpe` soit exécuté. Le serveur `gaston` va recevoir la demande de plugin `check_disk1` et va donc exécuter en local le plugin `check_disk` sur la partition `/dev/sda3`. Le résultat est renvoyé à `nrpe`, qui le renvoie au plugin `nrpe` sur la machine `nagios` qui va le rendre à `nagios` lui-même. ouf ! Dans la série pourquoi faire simple lorsque l'on peut faire compliqué...

## Passage par SSH

Une manière de se passer de NRPE est d'utiliser [SSH et une authentification par clef pour les deux utilisateurs nagios des deux machines](#) [3]. L'idée est assez simple, il s'agit de lancer, via `ssh` le plugin `nagios` distant par le biais d'une commande `nagios` conçue pour cela. Il faut toujours configurer de nouvelles commandes mais au moins cette configuration reste centralisée sur le serveur `nagios` et ne demande pas l'ouverture de ports supplémentaires.

Pour commencer, nous devons créer un plugin capable de relayer une commande sur une machine distante

```
#!/bin/sh

host=$1
shift
command=$*;
command=/usr/lib/nagios/plugins/$plugins$command
ssh [4] $host "$command"

/usr/lib/nagios/plugins/ssh_check
```

Pas très sorcier. Ensuite disons que nous voulions auditer l'espace disque, nous allons ajouter une nouvelle commande :

```
define command{
  command_name    check_disk
  command_line    $USER1$/ssh_check $HOSTADDRESS$ /check_disk -w 10% -c 5% -p /var -
  p /tmp -p /storage -p / home -p /
}

objects/commands.cfg
```

Ceci fait, la commande s'utilise comme les autres et renvoie à Nagios les résultats distants sans broncher.

## conclusion

Nagios n'est clairement pas un outil pour newbies. Mais une fois le paramétrage en main il se révèle être d'une grande souplesse. De plus son système de greffons permet de l'étendre très simplement. Si vous comptez aller plus loin avec Nagios, je vous conseille d'aller regarder par [ici](#) [5].

<http://artisan.karma-lab.net/node/1157>  
(C) artisan numerique - CC BY-SA

### Liens:

- [1] <http://www.nagios.org>
- [2] <http://artisan.karma-lab.net/node/22>
- [3] <http://artisan.karma-lab.net/node/82>
- [4] <http://pwet.fr/man/linux/commandes/ssh>
- [5] <http://blog.nicolargo.com/>