

Nagios



nicolargo

eBook
version 1.0

**eBook sous licence libre
Creative Common BY NC**

**Auteur:
Nicolas Hennion
aka Nicolargo**

Table des matières

[Pourquoi cet ebook ?](#)

[Nagios vs le reste du monde](#)

[Installation pas à pas de Nagios](#)

[Avant de commencer](#)

[Installation du système d'exploitation GNU/Linux](#)

[Installation de pré-requis système](#)

[Téléchargement des sources de Nagios](#)

[Compilation de Nagios depuis les sources](#)

[Compilation des plugins Nagios depuis les sources](#)

[Installation automatique depuis un script](#)

[Récupération du script d'installation automatique](#)

[Exécution du script d'installation automatique](#)

[Informations sur l'installation](#)

[Premier test de Nagios](#)

[Maintenir à jour son serveur Nagios, la méthode manuelle](#)

[Backup de l'ancienne version](#)

[Téléchargement de la dernière version de Nagios](#)

[Compilation et installation](#)

[On décompresse puis on lance la compilation:](#)

[Vérification de la configuration et redémarrage de Nagios](#)

[Mise à jour des plugins de Nagios](#)

[Téléchargement des plugins](#)

[Mise à jour des plugins](#)

[Redémarrage de Nagios](#)

[Maintenir à jour son serveur Nagios, la méthode automatique](#)

[Récupération du script](#)

[Lancement du script](#)

[Et si la mise à jour se passe mal ?](#)

[Informations sur la mise à jour](#)

[Configuration de Nagios](#)

[Arborescence des fichiers de configuration](#)

[Configuration des fichiers cgi.cfg, nagios.cfg et resource.cfg](#)

[Configuration de Nagios: nagios.cfg](#)

[Configuration des CGI: cgi.cfg](#)

[Configuration des ressources externes: resource.cfg](#)

[Configuration des objets](#)

[Définition des machines réseaux: network.cfg](#)

[Définition des machines serveurs: hostservers.cfg](#)

[Définition des machines clientes \(utilisateurs\): hostclients.cfg](#)

[Durées et fréquences des checks](#)

[Définition de la période d'activité d'un objet](#)

[Définition des intervalles de vérification d'un objet](#)

[Et les notifications ?](#)

[Nagios et les greffons \(aka plugins\)](#)

[Le protocole SNMP](#)

[Les plugins locaux](#)

[Les plugins actifs avec NRPE](#)

[Les plugins passifs avec NSCA](#)

[Et sous Windows ?](#)

[Quelques exemples de plugins Nagios](#)

[Superviser un serveur Web](#)

[Superviser un serveur Web sécurisé](#)

[Superviser un serveur de messagerie](#)

[Superviser un serveur Asterisk](#)

[Superviser un serveur de fichiers](#)

[Superviser un serveur LDAP](#)

[Superviser un serveur de base de données](#)

[Superviser un serveur réseau](#)

[Supervision de disques RAID hardware HP Proliant](#)

[Supervision de disques RAID 1 logiciel sous FreeBSD](#)

[Supervision d'un trunk IAX sur un serveur Asterisk](#)

[Superviser l'espace disque avec Nagios via SNMP](#)

[Configuration de la machine à surveiller](#)

[Configuration du serveur Nagios](#)

[Surveiller vos espaces disques SMB avec Nagios](#)

[Surveiller les interfaces de son Cisco avec Nagios](#)

[Installation du plugin](#)

[Test du plugin](#)

[Configuration de Nagios](#)

[Et si en plus je veux...](#)

[Détection des attaques DDOS avec Nagios](#)

[Installation du script](#)

[Configuration de Nagios](#)

[Surveiller vos serveurs Windows avec Nagios](#)

[Installation de check_nt](#)

[Installation de NSClient++](#)

[Configuration de Nagios pour surveiller vos machines Windows](#)

[Surveiller vos serveurs GNU/Linux avec Nagios](#)

[Installation de NRPE depuis les sources](#)

[Récupération des sources](#)

[Installation de NRPE](#)

[Installation des plugins Nagios standards](#)

[Correction des droits sur les fichiers](#)

[Lancement automatique au démarrage](#)

[Surveiller vos serveurs Linux avec Nagios et NRPE](#)

[Sur votre serveur Nagios...](#)

[Sur votre serveur Linux à surveiller...](#)

[On teste la communication...](#)

[On configure Nagios...](#)

[Surveiller la mémoire de vos serveurs avec Nagios](#)

[Installation de NRPE](#)

[Installation du plugin de supervision de la mémoire](#)

[Configuration de NRPE pour prendre en compte le script check_memory](#)

[Configuration du serveur Nagios](#)

Pourquoi cet ebook ?

Le blog de Nicolargo aborde de nombreux sujets gravitant dans la galaxie des logiciels libres mais il y en a un qui suscite un intérêt constant chez mes lecteurs: la supervision système et réseau. La complexité de ces logiciels et la faiblesse de la documentation Francophone (bien que ce soit en train de changer) y est sûrement pour quelque chose.

Cet eBook a pour objectif de réunir dans un seul fichier, facilement transportable ou imprimable, l'ensemble des articles que j'ai écrits sur le sujet depuis la création de mon blog.

Je vous souhaite une bonne lecture.

Nicolargo

Nagios vs le reste du monde

Un lecteur de mon blog m'a posé, par mail, la question suivante:

//

Je suis à la recherche d'un produit de supervision et je trouve des produits comme Patrol visualis, Spectrum, HP Open View... aux prix exorbitants..

- j'aurai à superviser: 20 switchs cisco, 10 routeurs cisco, des serveurs windows 2003, ISA, exchange et 60 cisco pix (en VPN ipsec et en extrémité un ISA) soit environ 100 équipements snmp.

- ma priorité étant les "cisco pix" afin de détecter les ruptures de ligne adsl et par exemple permettre d'envoyer un mail, sms avec une remontée d'alerte visuelle sur le produit de supervision...

- obtenir des graphiques sur la bande passante utilisée...

*- les équipements non snmp, peut-on les gérer par un ping à intervalle régulier ?
Ce produit est-il difficile à mettre en oeuvre ?*

//

Ce genre de question revient régulièrement quand on doit faire le choix entre une solution open-source et une solution payante. Dans ce cas précis et au vu du nombre relativement faible de machine à "monitorer", je pense qu'une solution à base de Nagios (pour le monitoring) et de Cacti (pour la génération des graphes de bande passante ou autres) est la meilleure.

En effet, une des qualités première de Nagios est son architecture facilement adaptable à l'environnement, notamment grâce à l'utilisation de plugins. Notre lecteur dispose d'un parc composé en majorité de machine compatible SNMP, il sera donc très facile d'adapter Nagios aux besoins des utilisateurs. Il existe même des sites comme [Nagios Exchange](#) permettant de trouver facilement des extensions spécifiques pour les équipements Cisco et les serveurs Windows. Pour les équipements non compatibles SNMP, il est bien sur possible de les surveiller grâce à des "ping". Pour être plus précis, on passe alors par un système de scripts permettant de faire un peu ce que l'on veut (faire une ping, émuler une requête HTTP, tester une connection telnet...). En ce qui concerne la remontée des alarmes, Nagios dispose de tout un panel de possibilité: message dans l'interface Web, message dans fichier syslog, mail, SMS (si vous disposez d'un serveur SMS).

Bien que l'[installation de Nagios](#) soit relativement simple, il faut avouer que les [taches d'administrations](#) (ajout/modification/suppression de machine) sont un peu lourdes. Elles peuvent cependant être faite par n'importe quel utilisateur maîtrisant les bases d'un OS Linux. Si vous souhaitez toutefois avoir un support professionnel, de nombreuses SSII offrent leurs services pour l'installation et la maintenance de telles solutions.

L'archivage des l'utilisation de la bande passante n'est par contre pas une fonction incluse dans Nagios, je conseille pour cela l'utilisation d'un autre logiciel: [Cacti](#). Ce dernier se présente lui aussi comme une interface Web et est bien plus simple à utiliser que des graphes MRTG.

Il ne faut pas limiter les solutions libres à des petites configurations comme celle de ce lecteur. En effet Nagios peut très bien superviser de très gros réseaux d'entreprise avec plusieurs milliers de machines (jusqu'à environ 3.000 serveurs). Cependant, sur le créneau des infrastructures de tailles importantes on trouve des projets parallèles plus optimisé. On peut notamment citer le projet [Shinken](#) qui est une ré-écriture complète en langage Python du coeur de Nagios. Développé par Jean Gabes (auteur d'un des ouvrage de référence sur Nagios), Shiken propose une optimisation des performances tout en restant compatible avec les fichiers de configurations et les plugins de Nagios. On peut ainsi superviser à partir d'un seul serveur de supervision plus de 10.000 machines.

Installation *pas à pas* de Nagios

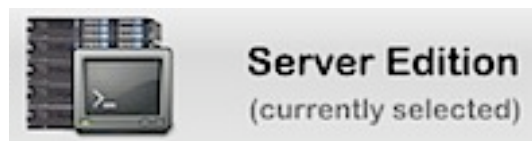
Trêve de blabla, entrons directement dans le vif du sujet avec l'installation du système d'exploitation et des pré-requis systèmes qui vont servir de base à notre serveur de supervision.

Avant de commencer

L'installation de Nagios est à réserver à des utilisateurs ayant des bases en système d'exploitation GNU/Linux. Si ce n'est pas le cas, je vous conseille de [vous tourner vers FAN](#) (Fully Automated Nagios), une distribution Linux avec les outils Nagios mais aussi Centreon pré-installés.

Installation du système d'exploitation GNU/Linux

J'ai choisi d'utiliser la distribution Ubuntu Server Edition 10.04 LTS. Sans juger de la qualité technique de cette distribution, je trouve que le support Francophone est très bien fait notamment par le biais de site comme [Ubuntu-fr](#). Comme je le dis souvent sur mon blog, il est important d'utiliser une distribution avec laquelle on est "à l'aise".



Il faut donc commencer par [télécharger](#) cette version sur un des serveurs. A moins d'être complètement allergique à l'éditeur de texte "vi", je vous conseille une installation standard, c'est à dire *sans interface graphique* Gnome/KDE/Xfce.

Installation de pré-requis système

On commence par mettre à jour le système en saisissant les commandes suivantes:

```
# sudo aptitude update
# sudo aptitude safe-upgrade
```

Dans cette série d'articles nous allons avoir besoin de compiler des sources de logiciels, il faut donc dans un premier temps installer le package "build-essential" qui comporte les bibliothèques de développement de bases:

```
# sudo aptitude install build-essential
```

Nagios offre aux utilisateurs une interface de type Web. Il faut donc installer un serveur HTTP sur notre machine de supervision. On ne va pas être très original... On va utiliser Apache 2. Bien qu'il soit possible d'utiliser des solutions plus légères et rapide comme [Nginx](#).

```
# sudo aptitude install apache2 wget
```

Certaines bibliothèques sont également nécessaires au bon fonctionnement de Nagios et de ces plugins :

```
# sudo apt-get install bind9-host dnsutils libbind9-60 libdns66
libisc60 libisccc60 libiscfg60 liblwres60 libradius1 qstat
radiusclient1 snmp snmpd
```

Pour tester votre serveur Web, il faut commencer par le lancer...

```
# sudo apache2ctl start
```

Puis on ouvre un navigateur Web sur un PC connecté en réseau à votre serveur et on saisie l'URL suivante: **http://<adresse_ip_serveur>**. Une page de bienvenue devrait s'afficher.

On installe les bibliothèques qui serviront à Nagios pour afficher de beaux diagrammes de votre réseau:

```
# sudo aptitude install libgd2-noxpm-dev libpng12-dev libjpeg62
libjpeg62-dev
```

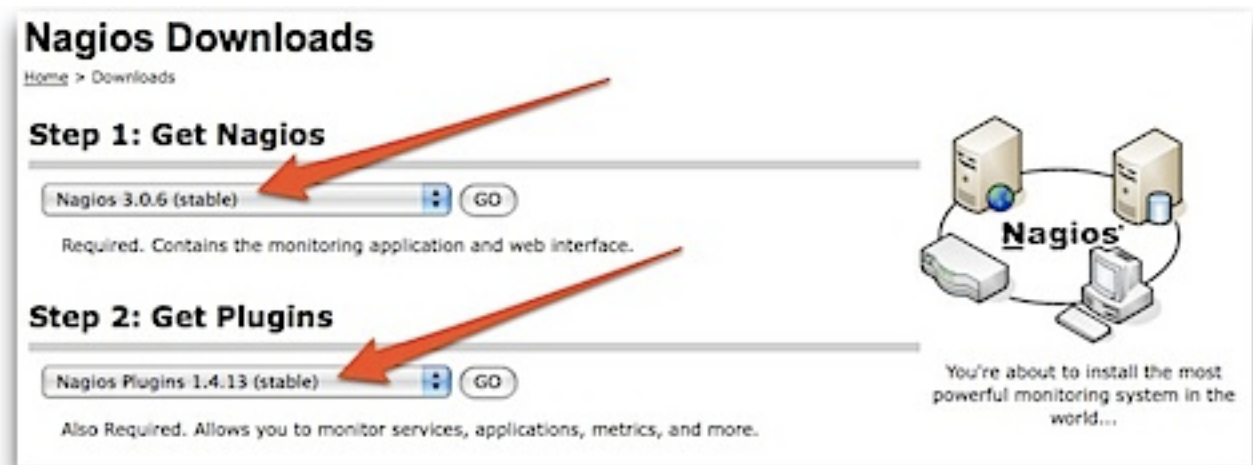
Pour des raisons évidentes de sécurité, le processus Nagios ne sera pas lancé en root (droit administrateur). Nous allons créer un utilisateur système nommé *nagios* et un groupe associé également nommé *nagios*. Le groupe *nagios* comprendra les utilisateurs *nagios* et *www-data* (utilisateur avec lequel le serveur Apache est lancé par défaut).

```
# sudo /usr/sbin/useradd nagios
# sudo passwd nagios
# sudo /usr/sbin/groupadd nagios
# sudo /usr/sbin/usermod -G nagios nagios
# sudo /usr/sbin/usermod -G nagios www-data
```

Il est maintenant temps de passer à la chose sérieuse en installant le cœur de notre système: le logiciel Nagios. A l'heure où je rédige cette procédure d'installation, la dernière version de Nagios est la 3.2.3 (à adapter si une nouvelle version est disponible).

Téléchargement des sources de Nagios

Il faut dans un premier temps se rendre à la page officielle des téléchargements, puis noter les derniers numéros de version de Nagios et des plugins Nagios (respectivement 3.2.3 et 1.4.14 lors de la rédaction de ce billet).



The screenshot shows the 'Nagios Downloads' page. It features two main steps: 'Step 1: Get Nagios' and 'Step 2: Get Plugins'. Each step has a dropdown menu with the version number and a 'GO' button. Red arrows point to the version numbers in both dropdowns. To the right, there is a diagram of a Nagios monitoring system with a central server and several monitored devices. Below the diagram, it says 'You're about to install the most powerful monitoring system in the world...'. The page also includes a breadcrumb 'Home > Downloads' and a note that the selected versions are 'Required'.

Ensuite, on télécharge ces versions sur notre serveur (pour simplifier l'installation, on passe les commandes en mode root):

```
# sudo -s
# cd /usr/src
# wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-3.2.3.tar.gz
# wget http://prdownloads.sourceforge.net/sourceforge/nagiosplug/nagios-plugins-1.4.15.tar.gz
```

Compilation de Nagios depuis les sources

On commence par décompresser les sources:

```
# tar xzf nagios-3.2.3.tar.gz
# cd nagios-3.2.3
```

Nous allons lancer la compilation grâce aux commandes suivantes:

```
# ./configure --with-command-group=nagios
...
General Options:
-----
Nagios executable: nagios
Nagios user/group: nagios,nagios
Command user/group: nagios,nagios
Embedded Perl: no
Event Broker: yes
Install ${prefix}: /usr/local/nagios
Lock file: ${prefix}/var/nagios.lock
Check result directory: ${prefix}/var/spool/checkresults
Init directory: /etc/init.d
Apache conf.d directory: /etc/apache2/conf.d
Mail program: /bin/mail
Host OS: Linux-gnu
# make all
# make install
# make install-config
# make install-commandmode
```

On installe ensuite le script de démarrage pour que Nagios se lance automatiquement avec votre serveur de supervision:

```
# make install-init
# ln -s /etc/init.d/nagios /etc/rcS.d/S99nagios
```

Il faut ensuite installer l'interface Web:

```
# make install-webconf
# sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
REMARQUE: Saisir le mot de passe pour le compte nagiosadmin de l'interface Web
# /etc/init.d/apache2 reload
```

Compilation des plugins Nagios depuis les sources

De base, Nagios est livré sans aucune extension (plugin). Il faut donc installer les plugins standards permettant de surveiller les machines de son réseau.

```
# sudo aptitude install fping libnet-snmp-perl libldap-dev
libmysqlclient-dev libgnutls-dev libradiusclient-ng-dev
# cd /usr/src
# tar xzf nagios-plugins-1.4.15.tar.gz
# cd nagios-plugins-1.4.15
# ./configure --with-nagios-user=nagios --with-nagios-group=nagios
# make
# make install
```

Installation automatique depuis un script

Il y a plusieurs méthodes pour installer Nagios, le système de supervision libre, sur un nouveau serveur. La plus simple est d'utiliser les dépôts officiels de votre distribution GNU/Linux, avec le désavantage de ne pas avoir les dernières versions disponibles. La seconde que nous avons vu dans le chapitre précédant permet d'effectuer une installation depuis les sources.

Je vous propose dans ce chapitre une troisième voie, qui mixe la simplicité de la première méthode et la finesse de la seconde. J'ai en effet développé un petit script (sous licence GPL) permettant d'automatiser l'installation d'un serveur Nagios complet sur une distribution GNU/Linux Ubuntu (j'ai validé le script sur Ubuntu Desktop et Ubuntu Server). Libre à vous de modifier ce script pour l'adapter à vos besoins. Si des âmes charitables veulent modifier le script pour l'adapter à d'autres distribution GNU/Linux ou BSD, je suis preneur pour les mettre en téléchargement sur Internet.

Récupération du script d'installation automatique

On lance la commande suivante pour télécharger le script sur son serveur et le rendre exécutable:

```
# wget http://svn.nicolargo.com/nagiosautoinstall/trunk/nagiosautoinstall-ubuntu.sh
# chmod a+x nagiosautoinstall-ubuntu.sh
```

PS: vous pouvez télécharger le script directement par l'URL suivante:

<http://svn.nicolargo.com/nagiosautoinstall/trunk/nagiosautoinstall-ubuntu.sh>

Exécution du script d'installation automatique

Il suffit ensuite de lancer le script et de répondre aux questions posées par le système:

```
# sudo ./nagiosautoinstall-ubuntu.sh
```

Informations sur l'installation

Dans la version actuelle du script la configuration finale est la suivante:

```
Nagios Core version      3.2.3
Nagios Plugins version   1.4.15
Utilisateur système:     nagios
Groupe système:         nagios
URL de l'interface Web:  http://localhost/nagios/
Utilisateur pour l'interface Web: nagiosadmin
```

Premier test de Nagios

Nagios est distribué avec une configuration initiale opérationnelle (elle permet notamment de surveiller... son serveur de supervision).

Nous allons donc tester les fichiers de configuration grâce à la commande suivante:

```
# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

Si tout ce passe bien vous devriez avoir le message suivant qui s'affiche:

```
Total Warnings: 0
Total Errors: 0
Things look okay - No serious problems were detected during the pre-flight check
```

Ensuite, on peut lancer le serveur Nagios:

```
# /etc/init.d/nagios start
```

Si le message suivant s'affiche, vous pouvez l'ignorer, ce n'est pas important...

```
Starting nagios:No directory, logging in with HOME=/
```

Il ne reste plus qu'à lancer un navigateur Web sur un PC de votre réseau et à saisir l'URL suivante (attention de bien mettre le / à la fin de l'URL):

http://<adresse_IP_serveur>/nagios/

Après une bannière d'authentification (login: **nagiosadmin** / password: **<votremotdepasse>**), vous devriez voir s'afficher:



Nagios
Version 3.0.6
December 01, 2008
[Read what's new in Nagios 3](#)

Copyright (c) 1999-2008 Ethan Galstad

Need help with Nagios?
A variety of worldwide support options are available to help you get Nagios up and running quickly. Visit www.nagios.org/support/ for information on:

- Installation
- Configuration
- Performance Tuning
- Integration
- Customization

Nagios Enterprises **Nagios**


SOURCEFORGE.NET

Nagios and the Nagios logo are trademarks, servicemarks, registered trademarks or registered servicemarks owned by Nagios Enterprises, LLC. Nagios is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE WARRANTY OF DESIGN, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

En cliquant sur le lien "Hostgroup Overview", vous devriez voir la supervision de votre serveur:



Linux Servers (linux-servers)

Host	Status	Services	Actions
localhost	UP	8 OK	

Puis le détail des services supervisés en cliquant sur "localhost":

Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Current Load	OK	01-16-2009 15:48:02	0d 1h 35m 43s	5/4	OK - Charge moyenne: 0.01, 0.01, 0.00
	Current Users	OK	01-16-2009 15:48:40	0d 1h 35m 5s	5/4	UTILISATEURS OK - 1 utilisateurs actuellement connectés sur
	HTTP	OK	01-16-2009 15:49:17	0d 1h 35m 26s	5/4	HTTP OK: HTTP/1.1 200 OK - 358 bytes en 0.001 secondes
	PING	OK	01-16-2009 15:49:55	0d 1h 37m 50s	5/4	PING OK - Paquets perdus = 0%, RTT = 0.02 ms
	Root Partition	OK	01-16-2009 15:50:32	0d 1h 37m 13s	5/4	DISK OK - free space: / 66606 MB (97% inode=99%)
	SSH	OK	01-16-2009 15:51:10	0d 1h 36m 35s	5/4	SSH OK - OpenSSH_5.1p1 Debian-3ubuntu1 (protocole 2.0)
	Send Check	OK	01-16-2009 15:51:47	0d 1h 35m 56s	5/4	SNMP OK - 100% libre (3153 MB sur un total de 3153 MB)
	Total Processes	OK	01-16-2009 15:52:25	0d 1h 35m 20s	5/4	PROCS OK: 17 processus avec ETAT = RSLDZT

Vous pouvez dès à présent configurer Nagios à votre besoin.

Maintenir à jour son serveur Nagios, la

méthode manuelle

Quand une nouvelle version de Nagios est mise à disposition, il est conseillé de mettre à jour vos serveurs de supervision. Ce chapitre a pour but de détailler la procédure à suivre pour mettre à jour son serveur Nagios entre une version 3.x et une version 3.y (pour une migration entre une version 2.x et une version 3.y, vous pouvez consulter [ce tutorial](#)).

Backup de l'ancienne version

Il est préférable de sauvegarder son ancienne configuration... au cas où..

```
# cd /tmp
# tar zcvfh ./nagios-backup.tgz /usr/local/nagios --exclude var/archives
# cp /usr/local/nagios/share/side.php side.php.MODIF
```

Si quelque chose se passe mal au niveau de la mise à jour, il sera toujours possible de revenir en arrière en saisissant les commandes suivantes:

```
# cd /tmp
# tar zxvf ./nagios-backup.tgz
```

Téléchargement de la dernière version de Nagios

Il faut au préalable des opérations suivantes, se loguer en tant qu'utilisateur *nagios* afin que les fichiers soient générés avec les bons droits.

```
# sudo - nagios
```

Puis télécharger la dernière version stable (3.2.0 au moment de l'écriture de ce billet).

```
# mkdir src
# cd src
# wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-3.2.0.tar.gz
```

Compilation et installation

On décompresse puis on lance la compilation:

```
# tar zxvf nagios-3.2.0.tar.gz
# cd nagios-3.2.0
# ./configure --with-command-group=nagios
# make all
```

Si la compilation se termine sans erreur, vous pouvez l'installer sur votre système, sinon je vous conseille de poster votre erreur dans le [forum officiel de Nagios](#):

```
# make install
# cp /usr/local/nagios/share/side.php /tmp/side.php.DEFAULT
# cp /tmp/side.php.MODIF /usr/local/nagios/share/side.php
```

Vérification de la configuration et redémarrage de Nagios

On va dans un premier temps vérifier que nos fichiers de configurations sont compatibles avec cette nouvelle version:

```
# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

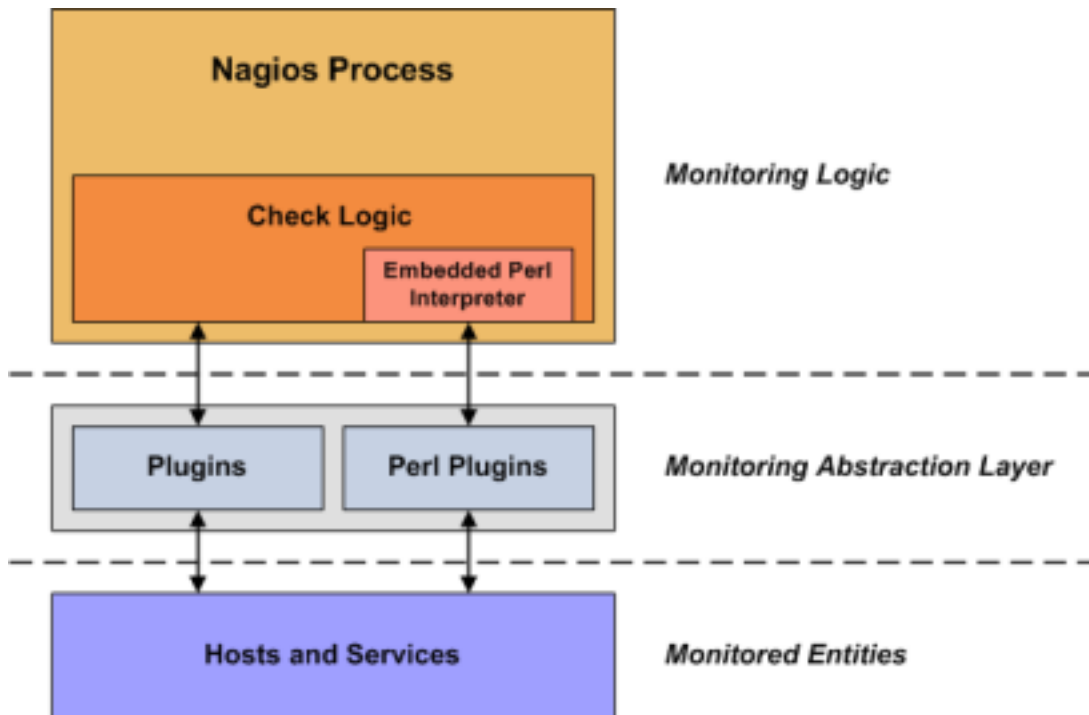
```
Si vous avez ce rapport à la fin:
Total Warnings: 0
Total Errors: 0
C'est que tout est OK !
```

Finalement on relance Nagios (en repassant en mode root) dans la nouvelle version:

```
# /etc/init.d/nagios restart
```

Mise à jour des plugins de Nagios

Comme vous le savez tous (ou pas), Nagios est composé d'un coeur (Nagios le bien nommé actuellement en version 3.x) et de packages d'extensions (NDO, Nagios-plugins...).



Voici une simple procédure à suivre pour mettre à jour les plugins dans Nagios.

Téléchargement des plugins

Il faut au préalable des opérations suivantes, se loguer en tant qu'utilisateur *nagios* afin que les fichiers soient générés avec les bons droits.

Sur Fedora:

```
su -l nagios
```

Sur Ubuntu ou Debian:

```
sudo -s nagios
```

Puis télécharger la dernière version stable des plugins (1.4.12 au moment de l'écriture de ce billet).

```
mkdir src
cd src
wget http://heanet.dl.sourceforge.net/sourceforge/nagiosplug/nagios-plugins-1.4.12.tar.gz
tar zxvf nagios-plugins-1.4.12.tar.gz
cd nagios-plugins-1.4.12
```

Mise à jour des plugins

On commence par compiler les plugins:

```
./configure  
make
```

puis on installe en lieu et place des anciens plugins:

```
make install  
exit
```

Certains plugins (check_dhcp) nécessitent les droits root pour être installés:

```
su - root  
make install-root  
exit
```

Redémarrage de Nagios

Afin que les plugins soit pris en compte, il faut redémarrer Nagios en utilisant les commandes suivantes:

Sur Fedora:

```
service nagios restart
```

Sur Ubuntu ou Debian:

```
/etc/init.d/nagios restart
```

Et voili, un beau Nagios et ses plugins à jour !

Maintenir à jour son serveur Nagios, la méthode automatique

Après [l'installation automatique](#), voici un nouveau script shell "maison" permettant de mettre simplement à jour votre serveur Nagios (core et plugins inclus).

Ce script est une synthèse des articles "[Comment mettre à jour son serveur Nagios ?](#)" et "[Mise à jour des plugins Nagios](#)". Il ne doit être utilisé que si vous avez installé Nagios à partir de cette [procédure](#) ou de [ce script d'installation automatique](#).

Le script est distribué sous licence GPL. Libre à vous de le modifier pour l'adapter à vos besoins. Si des âmes charitables veulent modifier le script pour l'adapter à d'autres distributions GNU/Linux ou BSD, je suis preneur pour les mettre en téléchargement sur mon serveur.

Récupération du script

On lance la commande suivante pour télé-charger le script sur son serveur et le rendre exécutable:

```
wget http://svn.nicolargo.com/nagiosautoinstall/trunk/nagiosautoupdate-ubuntu.sh
chmod a+x nagiosautoupdate-ubuntu.sh
```

PS: vous pouvez télécharger le script directement par l'URL suivante:

<http://svn.nicolargo.com/nagiosautoinstall/trunk/nagiosautoupdate-ubuntu.sh>

Lancement du script

Il suffit ensuite de lancer le script et de répondre aux questions posées par le système:

```
sudo ./nagiosautoupdate-ubuntu.sh
```

Et si la mise à jour se passe mal ?

Le script archive la configuration n-1, il suffit donc d'ouvrir un terminal et de saisir les commandes suivantes pour revenir dans l'ancienne version:

```
cd /
sudo tar zxvf /tmp/nagios-backup.tgz
```

Informations sur la mise à jour

Dans la version 0.1 du script la mise à jour se fera vers:

Nagios Core version 3.2.3
Nagios Plugins version 1.4.15

Configuration de Nagios

Le passage en version 3.0 de **Nagios** (l'outil de monitoring système et réseau) a apporté son lot de nouveautés. L'une d'elle est la réorganisation des fichiers de configuration. Nous allons dans ce billet détailler cette structure et préparer notre configuration de Nagios pour qu'elle soit facile à administrer...

Arborescence des fichiers de configuration

Si vous avez suivi ce [tutorial](#) pour l'installation de Nagios 3 depuis les sources, vos fichiers de configuration (fichiers se terminant avec l'extension `.cfg`) se trouve sous le répertoire `/usr/local/nagios/etc/`.

Personnellement, j'utilise la structure suivante:

cgi.cfg

> Définition des paramètres des scripts CGI. Vous pouvez utiliser le fichier fourni par défaut par Nagios.

nagios.cfg

> Fichier de configuration de Nagios. A modifier par vos soins selon votre configuration et l'arborescence choisie. Vous pouvez partir du fichier fourni en standard par Nagios et le modifier selon votre configuration.

resource.cfg

> Définition des ressources externes. Vous pouvez utiliser le fichier fourni par défaut par Nagios.

objects/

> C'est dans ce sous-répertoire que sont centralisé les définitions des machines et services à surveiller part votre serveur Nagios.

objects/commands.cfg

> C'est là que nous allons définir les commandes utilisées par Nagios pour interroger vos machines. Vous pouvez partir du fichier fourni en standard par Nagios et le modifier selon votre configuration.

objects/contacts.cfg

> Dans ce fichier, il faut configurer les contacts pouvant être prévenu en cas d'alerte. Vous pouvez partir du fichier fourni en standard par Nagios.

objects/hostclients.cfg

> Ce fichier est à créer, il comportera la définition de toutes vos machines clients à surveiller (c'est à dire les postes utilisateurs). Ce fichier n'existe pas dans la structure de base de Nagios.

objects/hostservers.cfg

> Ce fichier est à créer, il comportera la définition de toutes vos machines serveurs à surveiller (c'est à dire les serveurs Web, DNS, DB...). Ce fichier n'existe pas dans la structure de base de Nagios.

objects/localhost.cfg

> Ce fichier est là pour que Nagios puisse surveiller le serveur sur lequel il est installé (localhost). Vous pouvez partir du fichier fourni en standard par Nagios.

objects/templates.cfg

> C'est le fichier où se trouve la définition des "templates". Vous pouvez partir du fichier fourni en standard par Nagios.

objects/timeperiods.cfg

> Ce fichier définit les périodes de temps. Vous pouvez partir du fichier fourni en standard par Nagios.

objects/network.cfg

> Ce fichier est à créer, il comporte la définition de toutes les machines composant l'infrastructure de votre réseau (routeur, switch ou hub, borne Wifi ...)

C'est une bonne base de départ mais vous pouvez l'adapter en fonction de vos besoins (par exemple avec un découpage plus fin au niveau des hosts).

Attention aux droits sur ces fichiers. Ils doivent être lisibles par l'utilisateur système avec lequel le daemon Nagios est lancé (user nagios par défaut). Pour être sûr de ne pas avoir de problème, je vous conseille de taper les commandes suivantes quand vous avez fini la configuration de vos fichiers:

```
# cd /usr/local/nagios/etc
# chown -R nagios *
```

Configuration des fichiers `cgi.cfg`, `nagios.cfg` et `resource.cfg`

Ces 3 fichiers définissent la configuration "maître" de Nagios. Ils sont donc à configurer de manière préalable à tous les autres fichiers `.cfg`.

Configuration de Nagios: `nagios.cfg`

Ce fichier définit comment Nagios doit fonctionner. Il dispose de nombreuses options. Je vous conseille donc de partir du fichier fourni en standard et de modifier les sections suivantes. Modifier le fichier ou les logs du processus Nagios seront écrits (vous pouvez laisser la valeur par défaut).

```
log_file=/var/log/nagios.log
```

Attention, ce fichier doit être en lecture/écriture l'utilisateur avec lequel le daemon Nagios est lancé (user nagios par défaut).

La deuxième chose à faire est de définir l'arborescence des fichiers de configuration. Dans notre cas, cela donne:

```
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/hostclients.cfg
cfg_file=/usr/local/nagios/etc/objects/hostservers.cfg
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/network.cfg
```

Les options suivantes permettent de configurer l'utilisateur et le groupe système utilisés pour lancer Nagios. Je vous conseille de laisser les valeurs par défaut (sinon il faut créer les utilisateurs).

```
nagios_user=nagios
nagios_group=nagios
```

Je vous laisse consulter les autres options de ce fichier.

Configuration des CGI: cgi.cfg

Comme vous le savez, Nagios se base sur une interface Web pour générer ses rapports. Cette interface Web est générée dynamiquement par des scripts CGI. Le fichier `cgi.cfg` a pour but de configurer ces CGI selon votre configuration. Là encore, je vous conseille de partir du fichier fourni par défaut.

On commence par définir l'emplacement du fichier `nagios.cfg`. En effet ce dernier doit être lisible par les CGI.

```
main_config_file=/usr/local/nagios/etc/nagios.cfg
```

On peut ensuite configurer l'URL avec laquelle on va accéder au serveur Web de Nagios. Si par exemple vous voulez y accéder par l'adresse `http://monboserveurnagios.com/nagios`, il faut éditer l'option:

```
url_html_path=/nagios
```

Enfin, il est fortement conseillé d'utiliser une authentification (même si c'est une basique authentification HTTP) pour accéder à Nagios:

```
use_authentication=1
authorized_for_system_information=admin
```

Configuration des ressources externes: resource.cfg

Nagios utilise un système de plugins. Le fichier `resources` est là pour définir où sont ces plugins. Le fichier fourni en standard définit un moyen pour accéder aux plugins standards:

```
$USER1$=/usr/local/nagios/libexec
```

Libre à vous d'ajouter d'autres répertoires de plugins.

Configuration des objets

Entrons dans le vif du sujet avec la définition des fichiers de configuration des objets composants notre réseau à surveiller. On commence par les fichiers génériques: `commands.cfg`, `contacts.cfg`, `templates.cfg` et `timeperiods.cfg`

Vous pouvez utiliser les fichiers fournis en standard avec Nagios. Pour l'adapter à notre besoin, j'ai juste modifié le fichier `templates.cfg` pour y ajouter les templates suivants:

Un premier template pour les machines de type Linux:

```
# Linux host definition template - This is NOT a real host, just a template!
define host{
    name linux-host ; The name of this host template
    use generic-host ; This template inherits other values from the generic-
    host template
    check_period 24x7 ; By default, Linux hosts are checked round the clock
    check_interval 5 ; Actively check the host every 5 minutes
    retry_interval 1 ; Schedule host check retries at 1 minute intervals
    max_check_attempts 10 ; Check each Linux host 10 times (max)
    check_command check-host-alive ; Default command to check Linux hosts
```

```

notification_period workhours ; Linux admins hate to be woken up, so we
only notify during the day
; Note that the notification_period variable is being overridden from
; the value that is inherited from the generic-host template!
notification_interval 120 ; Resend notifications every 2 hours
notification_options d,u,r ; Only send notifications for specific host
states
contact_groups admins ; Notifications get sent to the admins by default
register 0 ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST, JUST A
TEMPLATE!
action_url /nagios/pnp/index.php?host=$HOSTNAME ; PNP
}

```

Un autre pour les machines Apple et BSD:

```

# BSD host definition template - This is NOT a real host, just a template!
define host{
    name bsd-host ; The name of this host template
    use linux-host ; This template inherits other values from the generic-
host template
}
# Apple host definition template - This is NOT a real host, just a template!
define host{
    name apple-host ; The name of this host template
    use bsd-host ; This template inherits other values from the generic-host
template
}

```

Pour les machines sous Windows:

```

define host{
    name windows-host ; The name of this host template
    use generic-host ; Inherit default values from the generic-host template
    check_period 24x7 ; By default, Windows servers are monitored round the
clock
    check_interval 5 ; Actively check the server every 5 minutes
    retry_interval 1 ; Schedule host check retries at 1 minute intervals
    max_check_attempts 10 ; Check each server 10 times (max)
    check_command check-host-alive ; Default command to check if servers
are "alive"
    notification_period 24x7 ; Send notification out at any time - day or
night
    notification_interval 30 ; Resend notifications every 30 minutes
    notification_options d,r ; Only send notifications for specific host
states
    contact_groups admins ; Notifications get sent to the admins by default
    register 0 ; DONT REGISTER THIS - ITS JUST A TEMPLATE
}

```



```
}
```

Et enfin pour les machines composant votre infrastructures réseaux:

```
define host{
    name network-host ; The name of this host template
    use generic-host ; Inherit default values from the generic-host template
    check_period 24x7 ; By default, switches are monitored round the clock
    check_interval 5 ; Switches are checked every 5 minutes
    retry_interval 1 ; Schedule host check retries at 1 minute intervals
    max_check_attempts 10 ; Check each switch 10 times (max)
    check_command check-host-alive ; Default command to check if routers
    are "alive"
    notification_period 24x7 ; Send notifications at any time
    notification_interval 30 ; Resend notifications every 30 minutes
    notification_options d,r ; Only send notifications for specific host
    states
    contact_groups admins ; Notifications get sent to the admins by default
    register 0 ; DONT REGISTER THIS - ITS JUST A TEMPLATE
}
```

Définition des machines réseaux: network.cfg

On va utiliser le template `network-host` et définir chaque noeud de votre réseau (routeur, switch...). Il faut bien sur que ces machines est une adresse IP. Les services à surveiller dépendent de votre configuration (par exemple la MIB des routeurs Cisco est très verbeuse). Par défaut Nagios "pingue" les machines et affiche leur status (UP/DOWN). Mais on peut aller bien plus loin, par exemple en mettant des alertes si un interface réseau est saturé, si la charge CPU de votre Firewall devient trop grande...

On utilise l'option `hostgroups` pour insérer le serveur dans le groupe `monreseau`.

Un exemple de définition pour un routeur Cisco et un switch réseau:

```
define host{
    use network-host
    host_name monboswitch
    alias Switch réseau 100 Mbps
    address 192.168.0.254
    hostgroups monreseau
}
```

```
define host {
    use network-host
    host_name monborouteur
    alias Routeur acces Internet
```

```
        address 192.168.0.1
        parents monboswitch
        hostgroups monreseau
    }
```

Pour surveiller sa liaison Internet, le plus simple est de pinguer sur Internet un serveur digne de confiance (style www.google.fr):

```
define host{
    use network-host
    host_name internet
    alias Liaison Internet (ping Google)
    address www.google.fr
    parents monborouteur
    hostgroups monreseau
}
```

Définition des machines serveurs: hostservers.cfg

On va utiliser un des templates (linux|bsd|apple|windows)-host et définir tous les serveurs de votre réseau. Pour chaque serveur, on va définir dans le même fichier (c'est une nouveauté par rapport à Nagios 2.x) les services à surveiller (par exemple on va vérifier que le serveur Web est bien lancé grâce au plugin `check_http`).

On utilise également l'option *hostgroups* pour insérer le serveur dans le groupe messerveurs.

```
define host{
    use linux-host
    host_name monboserveur
    alias Serveur Web
    address 192.168.0.100
    parents monboswitch
    hostgroups messerveurs
}

define service{
    use generic-service
    host_name monboserveur
    service_description HTTP
    check_command check_http!8080
}
```

Définition des machines clientes (utilisateurs): hostclients.cfg

On va utiliser un des templates (linux|bsd|apple|windows)-host et définir toutes les machines

clients de votre réseau.

On utilise également l'option *hostgroups* pour insérer le serveur dans le groupe mesclients (il est possible de faire appartenir un même host à plusieurs groupes en listant dans les même options les différents groupes séparés par des virgules).

```
define host{
    use apple-host
    host_name monboportable
    alias MacBookPro
    address 192.168.0.200
    parents monboswitch
    hostgroups mesclients
}

define host{
    use linux-host
    host_name monbopc
    alias Ubuntu
    address 192.168.0.201
    parents monboswitch
    hostgroups mesclients
}
```

Durées et fréquences des checks

Pour surveiller votre réseau, **Nagios** utilise un certain nombre d'objets (machines, services, contacts...). Par défaut, un objet hérite une grande partie de ses paramètres du template nommé "generic-*". Nous allons dans ce billet nous focaliser sur les notions de temps et de fréquences de ces objets.

Pour rappel, à un instant *t*, un objet peut avoir un des les états suivants:

- **OK**: tout va bien, votre objet fonctionne correctement
- **WARNING**: votre objet ne fonctionne pas nominalement
- **CRITICAL**: votre objet ne fonctionne plus
- **UNKNOWN**: impossible de déterminer l'état de votre objet

Un exemple: imaginons un service qui surveille la débit de votre liaison Internet...

- **OK**: le débit est inférieure à 70% de la bande passante totale
- **WARNING**: la débit est supérieure à 70% de la bande passante totale
- **CRITICAL**: la liaison Internet est DOWN (plus de connectivité avec Internet)
- **UNKNOWN**: impossible de récupérer les valeurs du débit

Définition de la période d'activité d'un objet

La variable `check_period` permet de définir l'intervalle de temps durant lequel l'objet est actif. On définit une période de temps en utilisant la structure `timeperiod`.

Par exemple pour définir une période de temps correspondant aux heures ouvrées de votre entreprise vous pouvez utiliser la définition suivante:

```
define timeperiod {
    timeperiod_name workhours
    alias Heures ouvrées
    monday 09:00-18:00 ; Lundi
    tuesday 09:00-18:00; Mardi
    wednesday 09:00-18:00; Mercredi
    thursday 09:00-18:00; Jeudi
    Friday 09:00-18:00; Vendredi
}
```

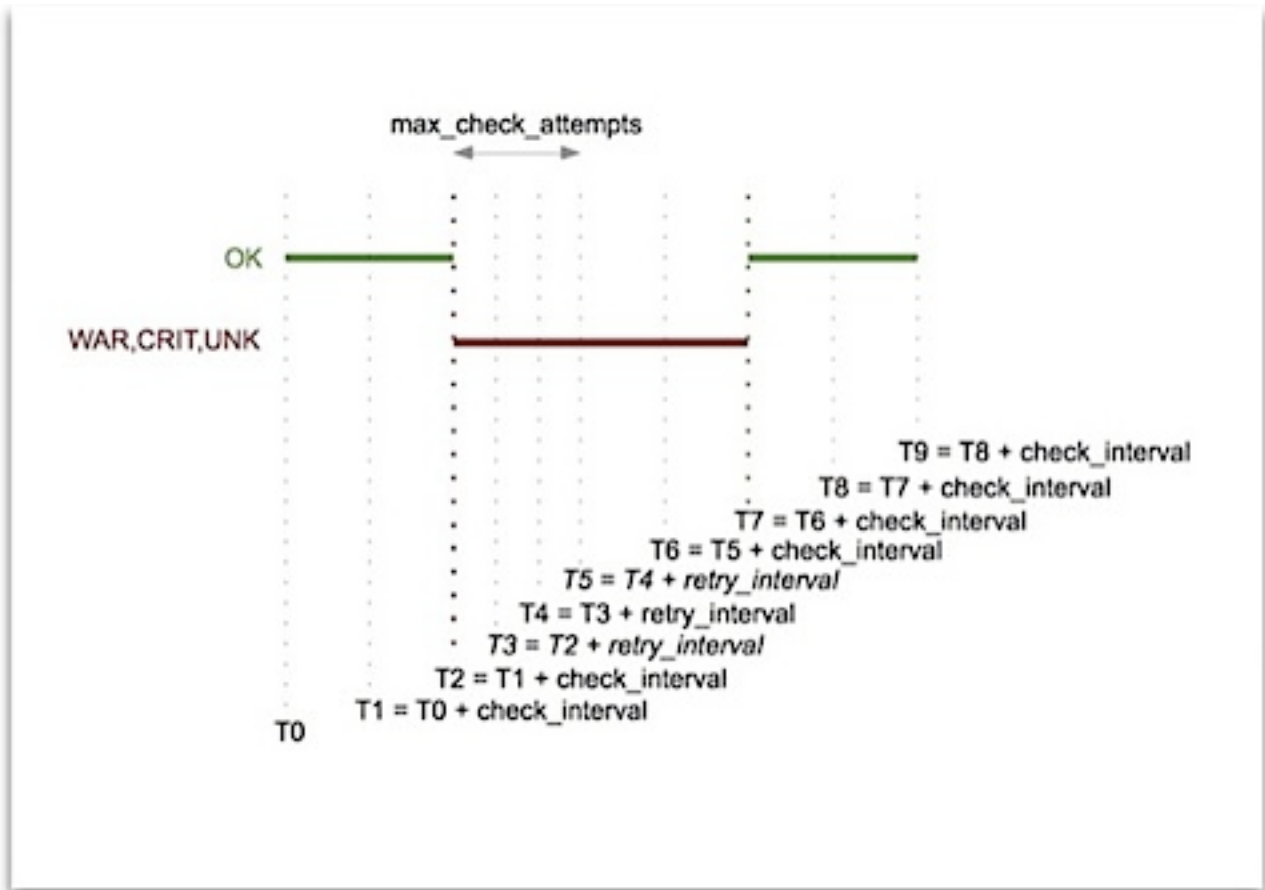
La déclaration de cette période de temps dans notre objet (par exemple un service) se fera ainsi:

```
define service {
    ...
    check_period workhours
}
```

Définition des intervalles de vérification d'un objet

Quand on se trouve dans une période d'activité d'un service, plusieurs paramètres rentrent en jeu pour fixer la durée et la fréquence des vérifications (checks) du objet en question. Quand un objet est OK, il est vérifié toutes les `check_interval` minutes. Si il passe WARNING, CRITICAL ou UNKNOWN, il est alors vérifié `max_check_attempts` fois à un intervalle `deretry_interval` minutes. Si l'état de l'objet n'est pas revenu à OK au bout des `max_check_attempts` essais, l'intervalle de vérification redevient de `check_interval` minutes...

Je sais ce n'est pas très simple mais ce schéma devrait vous aider à comprendre.



Un exemple de paramétrage avec un valeur de `check_interval` (quand tout va bien) à 10 minutes puis un `retry_interval` (quand cela commence à aller mal) à 2 minutes et un nombre de `max_check_attempts` à 3:

```
define service {
    ...
    check_interval 10
    retry_interval 2
    max_check_attempts 3
}
```

Et les notifications ?

Il existe des variables permettant de fixer comment les notifications sont remontés aux administrateurs.

La première variable est `first_notification_delay`. Elle permet de définir le temps (en secondes) que Nagios doit attendre avant d'envoyer une notification quand un objet passe d'un état OK à un état WARNING, CRITICAL ou UNKNOWN. Une valeur de 0 permet d'envoyer la notification dès ce changement d'état. La variable `notification_interval` permet, en cas de problème sur

un objet, de fixer l'intervalle de temps (en minutes) entre deux notifications. Pour que Nagios n'envoie qu'une seule fois une alerte, il faut fixer cette variable à 0.

Par exemple, pour que la première notification se fasse immédiatement, sans répétition, un objet doit être défini ainsi:

```
define service {
    ...
    first_notification_delay 0
    notification_interval 0
}
```

Nagios et les greffons (aka plugins)

A mes yeux, la principale force de Nagios est sa grande modularité qui lui permet de s'adapter aux besoins des utilisateurs. Il est ainsi possible de surveiller un grands nombres de paramètres sur les machines de votre réseau. Nous allons dans ce billet évoquer les différentes méthodes que l'on peut utiliser pour récupérer ces informations.

Le protocole SNMP

C'est LE protocole pour la gestion de réseaux. Basée sur UDP (port 161), ce protocole de communication permet la remontée d'informations stockées dans la table MIB ("Management Information Base") des machines. SNMP se base sur une architecture client (Nagios) / serveur (la machine à surveiller). Il est donc nécessaire que vos machines soient compatibles SNMP. C'est le cas de la plupart des équipements réseaux de type routeurs, commutateurs Ethernet...

Sur les serveurs de type Linux, il suffit d'installer le daemon `snmpd` de la suite [Open-SNMP](#). Sur Windows, il est également possible d'installer le daemon [Net-SNMP](#). Enfin, Mac OS X inclue un daemon SNMP (UCD-SNMP), il suffit de suivre [cette procédure](#) pour l'activer.

Les plugins locaux

En standard, SNMP ne remonte que des informations systèmes basiques. Pour aller plus loin et surveiller des processus plus complexe, Nagios à mis en place un système de type plugins locaux. Un plugin local est un script localisé sur le serveur Nagios (`/usr/lib/nagios/plugins` sous Linux, c'est pour cela que l'on dit qu'il est local).

Ce script, lancé à la demande de Nagios, doit retourner un code dont la signification est la suivante:

- Code 0: **OK** - Tout va bien
- Code 1: **WARNING** - Alerte

- Code 2: **CRITICAL** - Alerte critique
- Code 3: UNKNOWN - Problème lors de l'exécution du plugin

En plus de ces codes, un plugins peut fournir d'autres informations (sous la forme d'une chaîne de caractères) qui seront affichées à côté du statut de la machine.

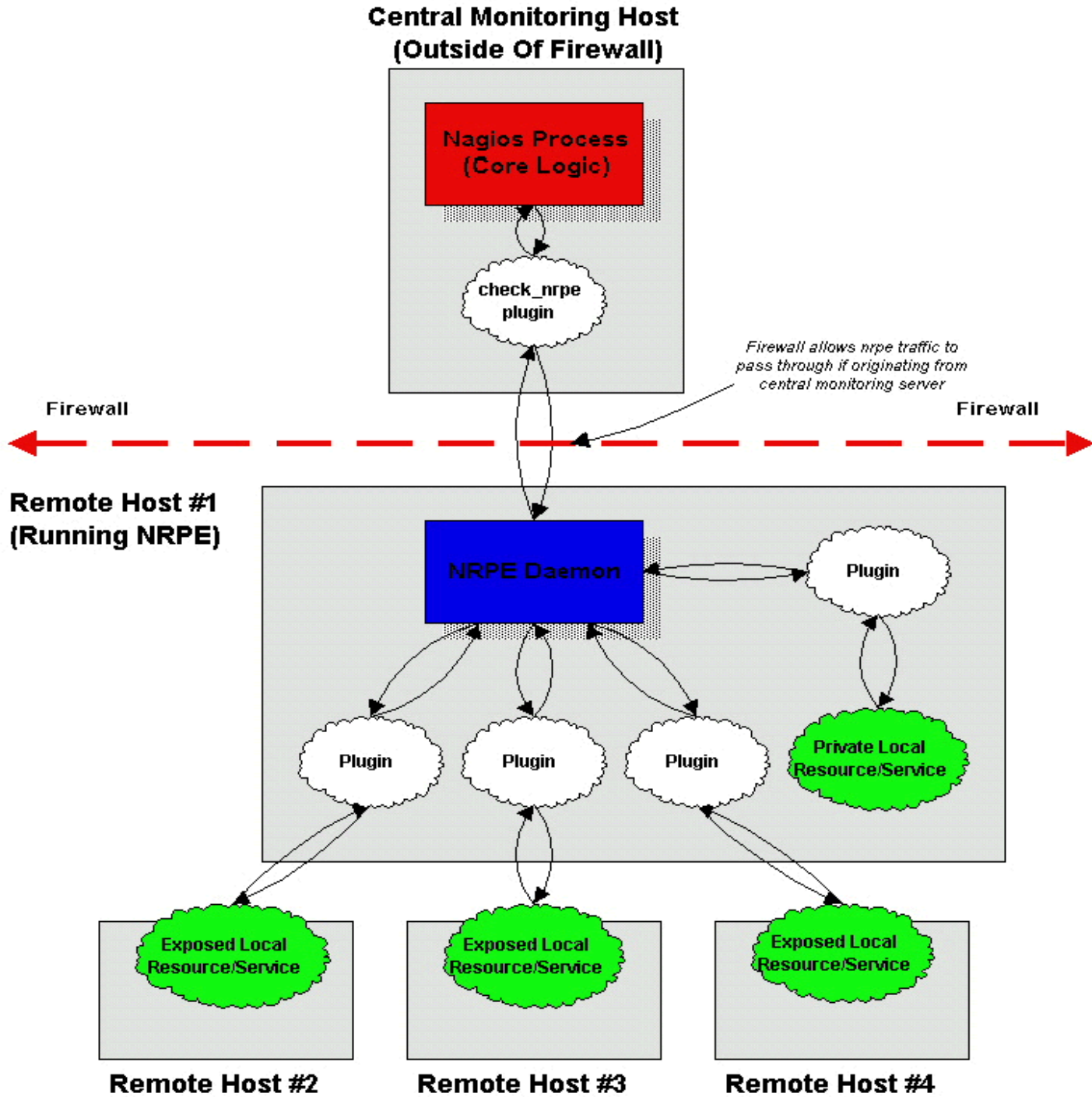
Les plugins actifs avec NRPE

A la différence des plugins locaux, le plugin NRPE permet l'exécution de plugins dit actifs directement sur les machines à surveiller.

L'architecture est la suivante (schéma trouvé sur le site officiel de Nagios):

Indirect Service Checks

Last Updated: 07-12-2001



Avec NRPE, la demande d'exécution d'un plugin actif est faite à l'initiative du serveur Nagios. La procédure interne est la suivante:

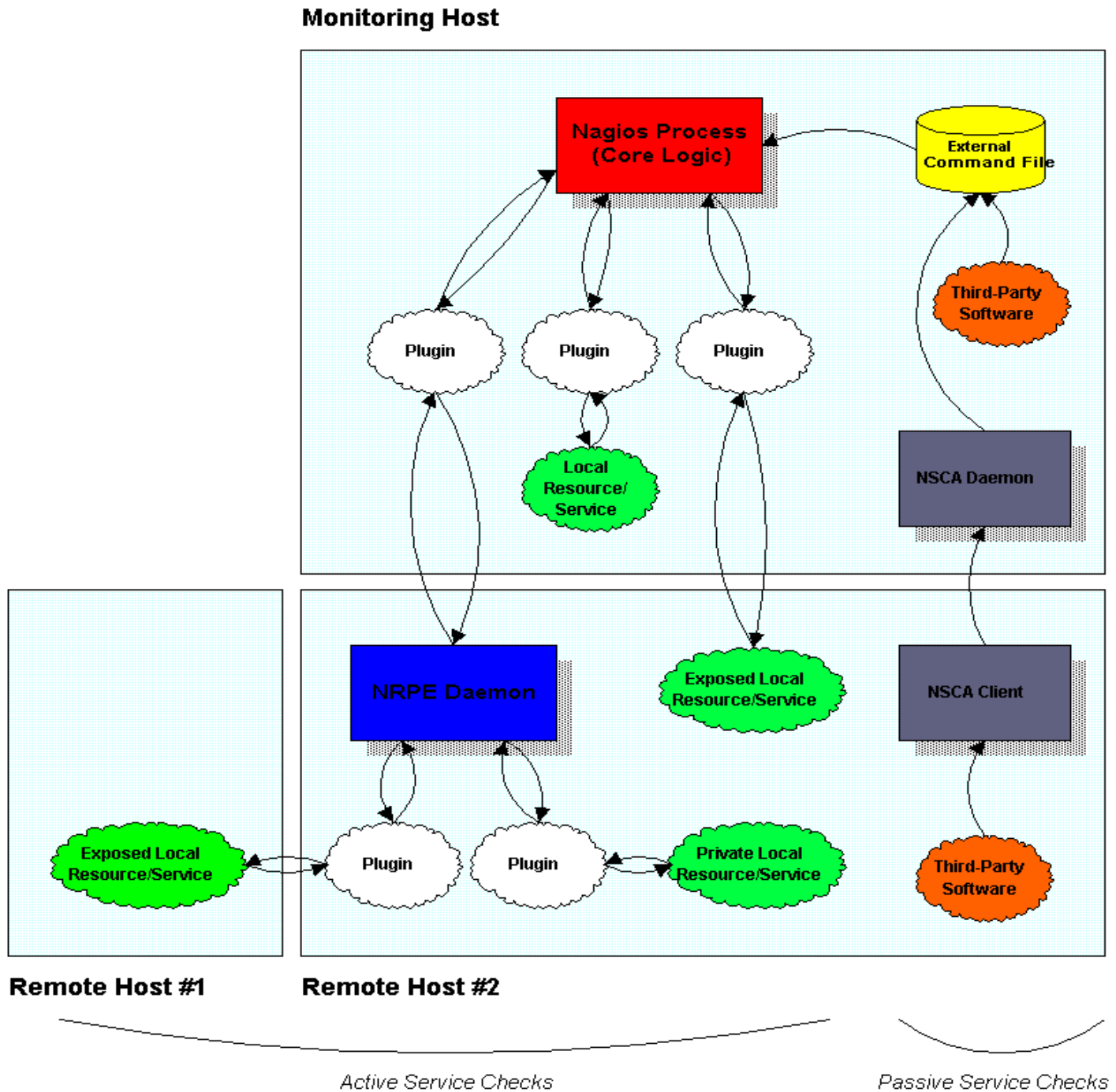
- le serveur Nagios demande l'exécution du plugin P sur la machine H
- le daemon NRPE hébergé sur la machine H, reçoit la requête d'exécution du plugin P
- le plugin P est exécuté sur la machine H
- le daemon NRPE de la machine H envoie le résultat du plugin P au serveur Nagios
- le serveur Nagios interprète les résultats retournés par le pugin P

Les plugins passifs avec NSCA

Comme l'on vient de le voir NRPE est déclenché à l'initiative du serveur Nagios. Ce mode de fonctionnement peut poser problème, par exemple dans le cas ou les machines à surveiller son derrière un réseau sécurisé par un Firewall ou si le processus à surveiller demande une fréquence d'exécution très courte. Le plugin NSCA répond à ce problème en proposant l'exécution de plugins passifs sur les machines à surveiller.

Using Active And Passive Checks Together

Last Updated: 07-21-2001



Ici, c'est donc le daemon NSCA qui va envoyer l'information au serveur Nagios. On peut comparer cette fonction à un TRAP SNMP.

Et sous Windows ?

Les plugins NRPE et NSCA ne sont disponibles que pour Linux et [Mac OS X](#). Si vous souhaitez surveiller des machines sous Windows (il vaut mieux les surveiller de prêt ces bêtes là...) , il va falloir utiliser le plugin [NSClient](#).

Quelques exemples de plugins Nagios

Quelques exemples de services pour Nagios. N'hésitez pas à donner d'autres exemples via les commentaires ou ma fiche de contact. J'intégrerai les exemples les plus pertinents au billet.

- superviser un serveur Web (HTTP)
- superviser un serveur Web sécurisé (HTTP over SSL)
- superviser un serveur de messagerie (mail)
- superviser un serveur VoIP (sip)
- superviser un serveur de fichiers (samba)
- superviser un serveur annuaire (LDAP)
- superviser un serveur de base de données (mySQL ou pgSQL)
- superviser un serveur réseau (DHCP et DNS)
- supervision de disques RAID (carte hardware HP Proliant)
- supervision de disques [RAID 1 logiciel sous FreeBSD](#)
- [supervision d'un serveur Asterisk \(suivre ce lien\)](#)
- supervision d'un trunk IAX sur un serveur Asterisk

Superviser un serveur Web

Nous utilisons pour cela le plugin [check_http](#) qui permet de tester si un serveur HTTP est bien lancé sur la machine à superviser.

Exemple:

```
define command{
    command_name check_http
    command_line $USER1$/check_http -I $HOSTADDRESS$ $ARG1$
}

define host{
    use generic-host
    host_name monserveur
}
```

```

        alias Serveur Web
        address 192.168.0.100
    }

    define service{
        use generic-service
        host_name monbeauserveur
        service_description HTTP
        check_command check_http
    }

```

Le plugin utilise par défaut une requête sur le port TCP/80. Pour changer ce port, vous pouvez utiliser l'option `-p` et créer une nouvelle commande (à utiliser dans le `check_command` de votre nouveau service):

```

    define command{
        command_name check_http_8080
        command_line $USER1$/check_http -I $HOSTADDRESS$ $ARG1$-p 8080
    }

```

Superviser un serveur Web sécurisé

Si votre serveur Web est sécurisé avec le protocole SSL. Il faut créer une nouvelle commande basée sur [check_http](#).

Exemple:

```

    define command{
        command_name check_https
        command_line $USER1$/check_http -S -I $HOSTADDRESS$ $ARG1$
    }

    define host{
        use generic-host
        host_name monserveur
        alias Serveur Web
        address 192.168.0.100
    }

    define service{
        use generic-service
        host_name monbeauserveur
        service_description HTTPS
        check_command check_https
    }

```

Superviser un serveur de messagerie

Pour surveiller un serveur Mail proposant les protocoles SMTP, POP3 et IMAP (merci [Dovecot](#)), nous allons utiliser les plugins [check_smtp](#), [check_pop](#) et [check_imap](#).

Exemple:

```
define host{
    use generic-host
    host_name monserveur
    alias Serveur de messagerie
    address 192.168.0.100
}

define service{
    use generic-service
    host_name monserveur
    service_description SMTP
    check_command check_smtp
}

define service{
    use generic-service
    host_name monserveurmail
    service_description POP
    check_command check_pop
}

define service{
    use generic-service
    host_name monserveurmail
    service_description IMAP
    check_command check_imap
}
```

Superviser un serveur Asterisk

Pour surveiller un serveur SIP (par exemple votre serveur Asterisk), nous allons utiliser le plugin [check_sip](#). n'étant pas fourni en standard, il faut d'abord l'installer puis le configurer avant de pouvoir l'utiliser comme service.

Installation:

```
cd /usr/src
wget http://www.bashton.com/downloads/nagios-check_sip-1.2.tar.gz
```

```
tar zxvf nagios-check_sip-1.2.tar.gz
cd nagios-check_sip-1.2
cp check_sip /usr/local/nagios/libexec/
chown apache:nagios /usr/local/nagios/libexec/check_sip
```

Configuration (à ajouter dans votre fichier commands.cfg de Nagios):

```
##### SIP #####
define command{
    command_name check_sip
    command_line $USER1$/check_sip -H $HOSTADDRESS$ -u
    sip:user@mondomaine.com
}
```

Il faut penser à mettre un nom d'utilisateur SIP valide après l'option -u.

Et enfin un exemple de service:

```
define host{
    use generic-host
    host_name monserveur
    alias Serveur SIP
    address 192.168.0.100
}

define service{
    use generic-service
    host_name monserveur
    service_description SIP
    check_command check_sip
}
```

Superviser un serveur de fichiers

Un serveur de fichier peut se baser sur de nombreux protocoles. Nous allons nous focaliser sur un serveur Windows ou Linux (avec Samba), souvent utilisé en entreprise grâce au protocole SMB.

Configuration (à ajouter dans votre fichier commands.cfg de Nagios):

```
##### SMB #####
define command{
    command_name check_smb
    command_line $USER1$/check_tcp -H $HOSTADDRESS$ -p 445
}
```

Et enfin un exemple de service:

```
define host{
    use generic-host
    host_name monserveur
    alias Serveur de fichiers
    address 192.168.0.100
}

define service{
    use generic-service
    host_name monserveur
    service_description SMB
    check_command check_smb
}
```

Superviser un serveur LDAP

Un serveur est souvent le coeur d'un système d'information. C'est donc un point critique à surveiller dans Nagios. Nous allons pour cela utiliser le service `check_ldap` qui prend en paramètres l'adresse du serveur LDAP ainsi que le DN à tester.

Configuration (à ajouter dans votre fichier `commands.cfg` de Nagios):

```
##### LDAP #####
define command{
    command_name check_ldap
    command_line $USER1$/check_ldap -H $HOSTADDRESS$ -b $ARG1$
}
```

Exemple de service:

```
define host{
    use generic-host
    host_name monserveur
    alias Serveur LDAP
    address 192.168.0.100
}
define service{
    use generic-service
    host_name monserveur
    service_description LDAP
    check_command check_ldap!"dc=mondomaine,dc=com"
}
```

Superviser un serveur de base de données

MySQL et pgSQL sont des serveurs de base de données open-source très répandus. Il existe donc deux plugins spécifiques `check_mysql` et `check_pgsql` disponible en standard avec Nagios. Personnellement, je n'utilise pas ces deux plugins car ils peuvent présenter une faille dans votre réseau. En effet, il utilise une requête SQL nécessitant un login/password. Hors ce couple apparaîtra dans la liste des processus lors de l'exécution du plugin par Nagios. Je préfère donc utiliser des plugins basés sur `check_tcp`.

Configuration (à ajouter dans votre fichier `commands.cfg` de Nagios):

```
##### SQL #####

define command{
    command_name check_pgsql
    command_line $USER1$/check_tcp -H $HOSTADDRESS$ -p 5432
}

define command{
    command_name check_mysql
    command_line $USER1$/check_tcp -H $HOSTADDRESS$ -p 3306
}
```

Exemple de service:

```
define host{
    use generic-host
    host_name monserveur
    alias Serveur MYSQL
    address 192.168.0.100
}

define service{
    use generic-service
    host_name monserveur
    service_description MYSQL
    check_command check_mysql
}
```

Superviser un serveur réseau

DHCP et DNS sont la base de votre infrastructure réseau. Nous allons donc utiliser les plugins `check_dns` et `check_dhcp` pour les surveiller de prêt.

Configuration (à modifier dans votre fichier commands.cfg de Nagios):

```
##### DNS #####
define command{
    command_name check_dns
    command_line $USER1$/check_dns -H www.google.fr -s $HOSTADDRESS$
}

##### DHCP #####
define command{
    command_name check_dhcp
    command_line $USER1$/check_dhcp -s $HOSTADDRESS$
}
```

Pour le check_dns, Je vous conseille de mettre un serveur qui ne risque pas de disparaître du jour au lendemain dans les DNS mondiaux en paramètre -H.

Exemple de service DNS:

```
define host{
    use generic-host
    host_name monserveur
    alias Serveur DNS
    address 192.168.0.100
}

define service{
    use generic-service
    host_name monserveur
    service_description DNS
    check_command check_dns
}
```

Exemple de service DHCP:

```
define host{
    use generic-host
    host_name monserveur
    alias Serveur DHCP
    address 192.168.0.100
}

define service{
    use generic-service
    host_name monserveur
    service_description DHCP
    check_command check_dhcp
}
```

```
}
```

Supervision de disques RAID hardware HP Proliant

Le but est de surveiller le bon état des disques RAID d'un serveur HP Proliant sous FreeBSD. Nous allons passer par un script lancé par NRPE.

Configuration à faire sur le serveur où se trouvent les disques RAID:

```
cd /usr/local/libexec/nagios/  
wget http://www.klintrup.dk/soren/proliant/check_smartarray.sh  
chmod 555 check_smartarray.sh
```

Il faut également modifier ce script pour qu'il soit exécutable par l'utilisateur Nagios (le user utilisé par NRPE).

```
Ligne 6> PATH="/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin"  
Ligne 7> DEVICES="$(sudo camcontrol devlist|grep "COMPAQ RAID"|sed -  
Ee 's/.*(pass[0-9]{1,3}).*/\1/')"  
Ligne 14> DEVICENAME="$(sudo camcontrol devlist|grep ${DEVICE}|sed -  
Ee 's/.*(da[0-9]{1,3}).*/\1/')"  
Ligne 15> DEVICESTRING="$(sudo camcontrol inquiry ${DEVICE} -D|sed -n -e 's/  
^[^<]*<([>]*\>).*$/\1/p')"
```

Puis enfin ajouter la ligne suivante au fichier /usr/local/etc/sudoers:

```
nagios ALL = NOPASSWD: /sbin/camcontrol
```

Edition du fichier /usr/local/etc/nrpe.cfg en ajoutant la ligne suivante:

```
command[check_raid]=/usr/local/libexec/nagios/check_smartarray.sh
```

Redémarrage du process NRPE:

```
/usr/local/etc/rc.d/nrpe2 restart
```

Configuration de Nagios, avec un nouveau service pour surveiller les disques RAID:

```
define service{  
    use                generic-service  
    host_name          monserveuravecduraid  
    service_description RAID  
    check_command      check_nrpe!check_raid  
}
```

Supervision de disques RAID 1 logiciel sous FreeBSD

Le but est de surveiller le bon état des disques RAID 1 logiciel d'un serveur sous FreeBSD.

On commence par configurer le serveur où se trouvent les disques RAID. Pour permettre à l'utilisateur nagios de lancer la commande de surveillance des disques RAID, on doit ajouter la ligne suivante au fichier `/usr/local/etc/sudoers`:

```
nagios ALL= NOPASSWD: /sbin/gmirror status
```

Édition du fichier `/usr/local/etc/nrpe.cfg` en ajoutant la ligne suivante:

```
command[check_raid]=/usr/local/bin/sudo /sbin/gmirror status | tail +2
```

Redémarrage du processus NRPE:

```
/usr/local/etc/rc.d/nrpe2 restart
```

Configuration de Nagios, avec un nouveau service pour surveiller les disques RAID:

```
define service{
    use                generic-service
    host_name          monserveuravecduraid
    service_description RAID
    check_command      check_nrpe!check_raid
}
```

Supervision d'un trunk IAX sur un serveur Asterisk

Quand vous voulez relier deux serveurs Asterisk entre eux, il faut utiliser un trunk IAX. Un trunk IAX est en fait un tunnel UDP (port 4569). Si vous voulez vérifier qu'un serveur Asterisk est apte à recevoir un trunk IAX, il faut lui envoyer une commande "POKE" sur le port UDP/4569, il répondra normalement par un "PONG".

Le script `check_iax` permet d'automatiser ce test. Une fois le script compilé (il est écrit en C) et installé dans le répertoire `/usr/local/nagios/libexec`, il faut créer la commande dans Nagios:

```
# 'check_iax' command definition, 800ms WARNING, 1000ms CRITICAL
define command{
    command_name check_iax
    command_line $USER1$/check_iax -H $HOSTADDRESS$ -w 800 -c 1000
}
```

Ensuite, il ne reste plus qu'à définir un service associé à une machine hébergeant le serveur

Asterisk.

```
define service{
    use                generic-service
    host_name          asterisk
    service_description IAX trunk
    check_command      check_iax
}
```

Superviser l'espace disque avec Nagios via SNMP



Voici une méthode simple rapide et efficace (enfin plus rapide à mettre en place que NRPE) pour surveiller l'espace disque disponible de ses machines Linux/BSD/Windows à partir de Nagios en utilisant le protocole SNMP.

Les pré-requis sont les suivants:

- avoir un **Nagios** correctement installé
- la machine à surveiller doit héberger **un serveur SNMP** dont la configuration permette au serveur Nagios de lire les informations (l'accès read-only v1/v2 de SNMP est suffisant)
- suivre la suite de ce billet 😊

Configuration de la machine à surveiller

Après avoir installé et configuré son serveur SNMP, il faut ajouter la ligne suivante au fichier de configuration snmpd.conf (la localisation de ce dernier est os dépendant):

```
disk / 100000
```

PS: le deuxième paramètre permet de fixer le seuil en dessous duquel une alerte SNMP est remontée. Il n'est pas très important pour nous car c'est Nagios qui va générer cette alerte avec nos propres valeurs.

On doit bien sûr relancer le service snmpd pour lire la configuration, par exemple:

```
/etc/init.d/snmpd restart
```

Configuration du serveur Nagios

La première chose à faire est de vérifier que l'on arrive bien à récupérer les informations SNMP sur la machine à surveiller (d'adresse IP 192.168.0.200 dans notre exemple). Pour cela on peut utiliser la commande suivante:

```
snmpget -v 1 -c public 192.168.0.200 .1.3.6.1.4.1.2021.9.1.9.1
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 32
```

La commande a réussi. On a bien récupéré la valeur 32 par SNMP. Donc On a 32% d'espace disque occupé sur le disque de la machine 192.168.0.200.

On configure Nagios de la manière suivante, on édite le fichier commands.cfg:

```
#####
# check_snmp_disk
#####

# Check free disk space using SNMP (add the "disk 100000" line to the
snmpd.conf)
define command{
    command_name check_snmp_disk
    command_line $USER1$/check_snmp -H $HOSTADDRESS$ -
o .1.3.6.1.4.1.2021.9.1.9.1 -C $ARG1$ -w $ARG2$ -c $ARG3$ -u "% used"
}
}
```

Puis on configure le service pour la machine à surveiller (dans un autre fichier comme par exemple services.cfg):

```
define service{
    use generic-service
    host_name Ma_Machine_192.168.0.200
    service_description DISK SPACE
    check_command check_snmp_disk!public!90!95
}
}
```

La fonction check_snmp_disk prend 3 paramètres:

- le nom de la communauté SNMP (public)
- le seuil au dessus duquel un warning est généré par Nagios (90%)
- le seuil au dessus duquel un error est généré par Nagios (95%)

Il ne reste plus qu'à relancer Nagios pour prendre en compte la configuration !

Surveiller vos espaces disques SMB avec Nagios

Nous allons voir dans ce chapitre une petite astuce pour surveiller facilement avec Nagios, l'espace disque disponible sous ses serveurs Windows ou Samba (protocole SMB).

La première chose à faire est de tester le plugin en ligne de commande. Par exemple pour surveiller l'espace disque monpartage du serveur monserveur (\\monserveur\monpartage) en utilisant le couple login/password monuser/monpassword:

```
/usr/local/nagios/libexec/check_disk_smb -H monserveur -s monpartage -u monlogin -p monpassword
```

Il est possible que vous rencontriez l'erreur suivante sous Ubuntu:

```
Can't exec "//monserveur/monpartage": No such file or directory at /usr/local/nagios/libexec/check_disk_smb line 166.
Use of uninitialized value $res in split at /usr/local/nagios/libexec/check_disk_smb line 172.
Use of uninitialized value $_ in pattern match (m//) at /usr/local/nagios/libexec/check_disk_smb line 180.
Result from smbclient not suitable
```

Dans ce cas, je vous conseille la lecture de [cette discussion dans le forum](#) pour corriger le problème.

On commence par éditer le fichier commands.cfg pour ajouter le plugin:

```
define command{
    command_name check_disk_smb
    command_line $USER1$/check_disk_smb -H $HOSTADDRESS$ -s $ARG1$ -u $ARG2$ -p $ARG3$
}
```

Puis on utilisera la définition de service suivante (à mettre par exemple dans le fichier objects/servers.cfg):

```
define service{
    use generic-service
    host_name monserveur
    service_description Disk space
    check_command check_disk_smb!servernas!monpartage!monuser!monpassword
}
```

Un alerte d'avertissement (**warning**) sera générée si l'espace disque est inférieur à 15%, une alerte critique (**critical**) si cette valeur passe à 5%.

Surveiller les interfaces de son Cisco avec Nagios

Qui veut surveiller l'état des interfaces de ses routeurs Cisco doit obligatoirement se plonger dans l'arborescence des MIB SNMP. Heureusement, Patrick Proy a eu la bonne idée d'intégrer dans un seul et même script Perl toutes les fonctions pour répondre à ce besoin. Nous allons dans ce chapitre voir comment installer et tester le plugin `check_snmp_int.pl` puis configurer Nagios pour prendre en compte ce nouveau plugin.

Installation du plugin

Il faut dans un premier temps se rendre dans le répertoire où les plugin Nagios sont stockés (`/usr/local/libexec/nagios` par exemple) puis y télécharger la dernière version du plugin (1.4.8 au moment de l'écriture de ce billet):

```
cd /usr/local/libexec/nagios
wget http://nagios.manubulon.com/check_snmp_int.pl
```

Ensuite on le rend exécutable par Nagios:

```
chown nagios:nagios check_snmp_int.pl
chmod 555 check_snmp_int.pl
```

Enfin on vérifie que l'utilisateur Nagios (ou celui qui doit lancer Nagios) peut exécuter ce script:

```
su - nagios
nagios> /usr/local/libexec/nagios/check_snmp_int.pl -h
... la syntaxe va s'afficher ...
nagios> exit
```

Si vous avez une erreur lors de l'exécution du script, cela peut être dû à l'absence des pré-requis suivants sur votre système:

- Perl doit être installé dans le répertoire `/usr/bin/perl`
- La librairie Perl nommée `Net::SNMP` doit être installée (via CPAN)
- Le fichier `utils.pm` doit être présent dans le répertoire des plugins Nagios

Test du plugin

Nous allons dans un premier temps chercher la liste des interfaces réseau du routeur Cisco (ou autre compatible avec MIB-2 standard) à superviser:

```
/usr/local/libexec/nagios/check_snmp_int.pl -H <adresseIPdurouteur> -C public -n zzzz -v
```

```
Alarm at 15 + 5
SNMP v1 login
Filter : zzzz
OID : 1.3.6.1.2.1.2.2.1.2.26, Desc : Async54
OID : 1.3.6.1.2.1.2.2.1.2.28, Desc : Async56
OID : 1.3.6.1.2.1.2.2.1.2.18, Desc : Async46
OID : 1.3.6.1.2.1.2.2.1.2.12, Desc : Async40
OID : 1.3.6.1.2.1.2.2.1.2.31, Desc : Async59
OID : 1.3.6.1.2.1.2.2.1.2.19, Desc : Async47
OID : 1.3.6.1.2.1.2.2.1.2.22, Desc : Async50
OID : 1.3.6.1.2.1.2.2.1.2.34, Desc : Async62
OID : 1.3.6.1.2.1.2.2.1.2.13, Desc : Async41
OID : 1.3.6.1.2.1.2.2.1.2.23, Desc : Async51
OID : 1.3.6.1.2.1.2.2.1.2.4, Desc : Null0
OID : 1.3.6.1.2.1.2.2.1.2.35, Desc : Loopback0
OID : 1.3.6.1.2.1.2.2.1.2.30, Desc : Async58
OID : 1.3.6.1.2.1.2.2.1.2.5, Desc : Async33
OID : 1.3.6.1.2.1.2.2.1.2.11, Desc : Async39
OID : 1.3.6.1.2.1.2.2.1.2.1, Desc : Ethernet3/0
OID : 1.3.6.1.2.1.2.2.1.2.39, Desc : Virtual-Access2
OID : 1.3.6.1.2.1.2.2.1.2.14, Desc : Async42
OID : 1.3.6.1.2.1.2.2.1.2.37, Desc : Virtual-Template1
OID : 1.3.6.1.2.1.2.2.1.2.16, Desc : Async44
OID : 1.3.6.1.2.1.2.2.1.2.20, Desc : Async48
OID : 1.3.6.1.2.1.2.2.1.2.17, Desc : Async45
OID : 1.3.6.1.2.1.2.2.1.2.29, Desc : Async57
OID : 1.3.6.1.2.1.2.2.1.2.36, Desc : Tunnel0
OID : 1.3.6.1.2.1.2.2.1.2.3, Desc : Serial3/1
OID : 1.3.6.1.2.1.2.2.1.2.25, Desc : Async53
OID : 1.3.6.1.2.1.2.2.1.2.2, Desc : Serial3/0
OID : 1.3.6.1.2.1.2.2.1.2.7, Desc : Async35
OID : 1.3.6.1.2.1.2.2.1.2.33, Desc : Async61
OID : 1.3.6.1.2.1.2.2.1.2.9, Desc : Async37
OID : 1.3.6.1.2.1.2.2.1.2.6, Desc : Async34
OID : 1.3.6.1.2.1.2.2.1.2.15, Desc : Async43
OID : 1.3.6.1.2.1.2.2.1.2.10, Desc : Async38
OID : 1.3.6.1.2.1.2.2.1.2.27, Desc : Async55
OID : 1.3.6.1.2.1.2.2.1.2.32, Desc : Async60
OID : 1.3.6.1.2.1.2.2.1.2.8, Desc : Async36
OID : 1.3.6.1.2.1.2.2.1.2.21, Desc : Async49
OID : 1.3.6.1.2.1.2.2.1.2.24, Desc : Async52
```

Il faut ensuite noter le nom système des interfaces à surveiller (par exemple Serial 3/0 et Serial 3/1 dans mon cas).

Puis on lance ensuite la commande pour vérifier l'état de l'interface Serial 3/0:


```
/usr/local/libexec/nagios/check_snmp_int.pl -H <adresseIPdurateur> -C public -n "Serial3/0" -r  
Serial3/0:UP:1 UP: OK
```

On a donc maintenant la commande à exécuter, il ne reste plus qu'à l'intégrer dans notre configuration de Nagios.

Configuration de Nagios

On commence par se rendre dans le répertoire où se trouvent les fichiers de configuration de Nagios (/usr/local/etc/nagios par exemple):

```
cd /usr/local/etc/nagios
```

Puis si vous êtes sous Nagios 3.x ou supérieur dans le sous-répertoire objects:

```
cd objects
```

Enfin on édite le fichier commands.cfg puis on y ajoute la section suivante:

```
#####  
# check_snmp_int.pl  
#####  
# 'check_snmp_int.pl', vérifier l'état de l'interface réseau  
define command{  
    command_name check_snmp_int  
    command_line $USER1$/check_snmp_int.pl -H $HOSTADDRESS$ -C $ARG1$ -n $ARG2$ -r  
}
```

On vient donc de créer une nouvelle commande Nagios qui appellera le plugin check_snmp_int avec deux paramètres:

- \$ARG1\$: nom de la communauté SNMP à utiliser (public la plupart du temps)
- \$ARG2\$: nom de l'interface à superviser

Enfin, on ajoute un nouveau service pour le routeur Cisco à superviser (par exemple dans le fichier network.cfg):

```
define host{  
    use routeur  
    host_name cisco  
    alias Routeur liaison WAN  
    address <adresseIProuteurcisco>  
}  
  
define service{  
    use generic-service
```

```

host_name          cisco
service_description LS 1
check_command      check_snmp_int!public!"Serial3/0"
}

```

Et si en plus je veux...

...surveiller la bande passante utilisée et envoyer une alerte si celle-ci dépasse un certain seuil ?

Il suffit de créer une nouvelle commande Nagios et d'utiliser l'option -k du plugin. Par exemple pour surveiller un interface Serial3/0 et emettre une alerte de warning si débit dépasse 1.6 Mbps puis une alerte critique si ce dernier passe la barre des 1.9 Mbps ?

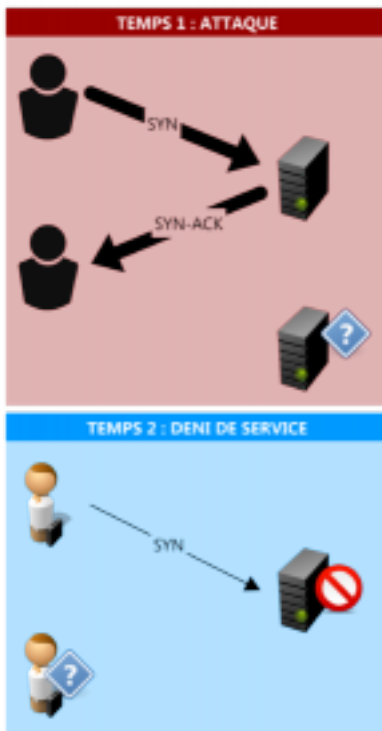
```

/usr/local/libexec/nagios/check_snmp_int.pl -H <adresseIPdurouteur> -C public
-n "Serial3/0" -k -w 1600,1600 -c 1900,1900

```

Je vous laisse convertir cette commande système vers une commande Nagios, cela vous fera un bon exercice...

Détection des attaques DDOS avec Nagios



Depuis l'affaire Wikileaks, les attaques réseau de type DDOS sont sous le feu de la rampe (et des médias). C'est une attaque assez difficile à détecter car contrairement à des attaques plus "classiques", elle se base sur le fait d'inonder la machine cible de requêtes venant d'un nombre important de machines zombies (c'est à dire infecté par un programme qui va lancer une attaque).

Nous allons dans ce billet voir comment utiliser Nagios pour envoyer des alertes en cas de détection d'une attaque de type **DDOS SYN Flood**.

Pour cela j'ai développé (sous licence GPL v3) un plugin Nagios disponible à l'adresse suivante:

http://svn.nicolargo.com/nagiosautoinstall/trunk/check_ddos.pl

Installation du script

Il faut disposer d'un **serveur Nagios correctement configuré**. Puis exécuter les commandes suivantes:

```
cd /usr/local/nagios/libexec
sudo rm -f check_ddos.pl
wget http://svn.nicolargo.com/nagiosautoinstall/trunk/check_ddos.pl
chmod a+rx check_ddos.pl
sudo chown nagios:nagios check_ddos.pl
```

Test du script:

```
./check_ddos.pl -w 50 -c 60
No DDOS attack detected (5/50)
```

Configuration de Nagios

Pour ajouter un service de détection DDOS SYN Flood sur la machine locale (en clair pour vérifier les attaques DDOS vers le serveur hébergeant Nagios), il faut dans un premier temps éditer le fichier `commands.cfg` (par défaut dans le répertoire `/usr/local/nagios/etc/objects`) pour ajouter la nouvelle commande de détection DDOS SYN Flood:

```
# check_ddos
define command{
    command_name check_ddos
    command_line $USER1$/check_ddos.pl -w $ARG1$ -c $ARG2$
}
```

Puis il faut éditer le fichier `localhost.cfg` (qui se trouve également dans le répertoire `/usr/local/nagios/etc/objects`):

```

# Define a DDOS SYN Flood detection service
# http://blog.nicolargo.com/?p=4100
# Warning: >50 SYN_RECV
# Critical: >70 SYN_RECV
define service{
    use local-service
    host_name bilbo
    service_description DDOS SYN Flood detect
    check_command check_ddos!50!70
}

```

Nous venons ainsi de définir un service qui va émettre une **alerte Warning** quand le serveur aura plus de 50 connexions de type SYN_RECV ouvertes (plus de 70 pour une **alerte Critical**). Ces chiffres sont bien sûr à adapter selon les serveurs...

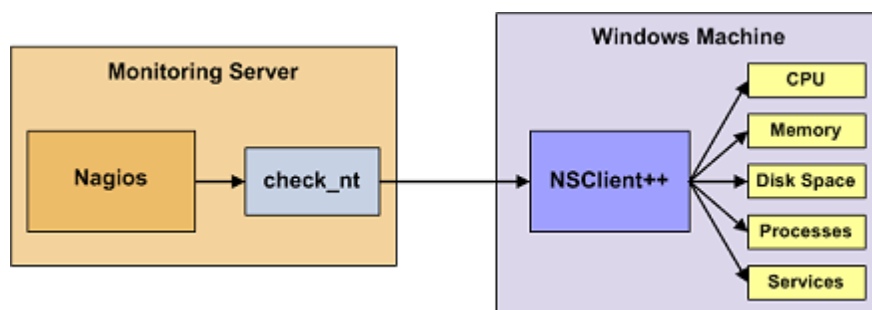
En bonus, si une alerte est générée, le plugin affiche le TOP 10 des adresses IP des machines zombies (pratique pour les **bloquer avec des règles de Firewall Iptables**).

Si vous souhaitez surveiller les attaques DDOS SYN Flood sur une autre machine, il faut utiliser **le plugin NRPE** qui va faire l'interface entre le serveur Nagios et le serveur à superviser.

Surveiller vos serveurs Windows avec Nagios

Voici un chapitre dédié aux lecteurs qui ont à administrer des machines sous Windows. Nous allons décrire l'installation de NSClient, un plugin permettant de récupérer un nombre important de d'informations à surveiller sur une machine Windows.

Comme **les plugins NRPE et NSCA** (disponible seulement sous Linux et Mac OS X), NSClient se base sur une architecture client/serveur. La partie cliente (nommée check_nt), doit être disponible sur le serveur Nagios. La partie serveur (NSClient++) est à installer sur chacune des machines Windows à surveiller.



Installation de check_nt

Il y a de forte chance que le plugin `check_nt` soit installé par défaut sur votre serveur Nagios. Pour le vérifier, il faut se rendre dans le répertoire Nagios où se trouvent vos plugins (`/usr/lib/nagios/plugins` sur Fedora, `/usr/local/nagios/plugins` sur Ubuntu).

```
# cd /usr/lib/nagios/plugins
# ls check_nt
check_nt
```

Si ce n'est pas le cas, il suffit de l'installer grâce aux commandes suivantes:

Fedora:

```
# sudo yum install nagios-plugins-nt
```

Ubuntu:

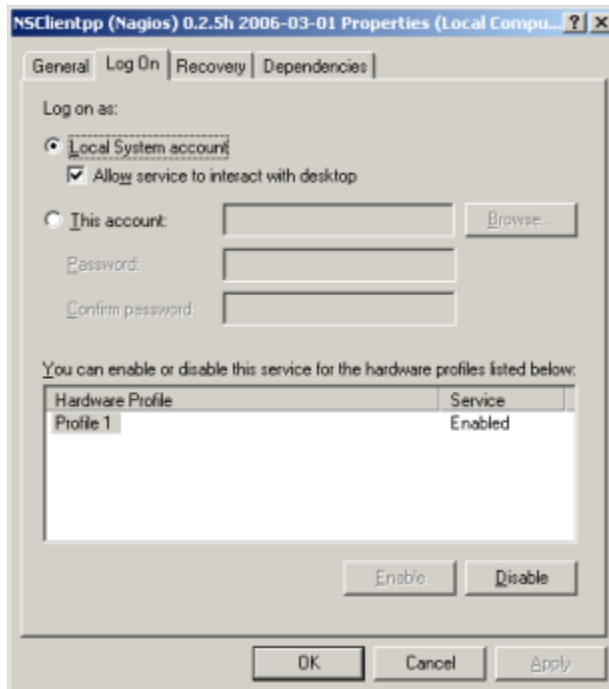
```
# sudo apt-get install nagios-plugins-nt
```

Installation de NSClient++

Remarque: cette opération est à faire sur l'ensemble des PC Windows à surveiller.

La première chose à faire est de télécharger la dernière version (0.2.5e ou supérieure) à l'adresse suivante: [Sourceforge de NSClient++](#). Ensuite il faut:

- "dézipper" le client dans le répertoire `c:\nsclient`
- ouvrir une commande DOS (`cmd.exe`)
- puis entrer les commandes suivantes:
 - `cd \nsclient`
 - `nsclient++ /install`
- Ouvrir le gestionnaire des services et vérifier que le service est autorisé à "Interagir avec le bureau"



- Editer le fichier c:\nsclient\NSC.INI en:
 - décommentant tous les modules listé dans la section [modules] sauf CheckWMI.dll et RemoteConfiguration.dll
 - décommentant la ligne allowed_hosts dans la section [Settings] et en y ajoutant l'adresse du serveur Nagios.
- puis entrer les commandes suivantes dans votre fenêtrés DOS:
 - cd \nsclient
 - nsclient++ /start

Pour tester que l'installation à bien marché, le plus simple est de faire un test depuis le serveur Nagios. Pour cela, il faut:

```
# cd /usr/lib/nagios/plugins
# ./check_nt -H IPMACHINEWINDOWS -v CLIENTVERSION -p 12489
Si tout ce passe bien, le client doit envoyer la version de NSClient (0.2.5e)
```

Si cela ne fonctionne pas, il faut peut être vérifier que la requête (TCP sur port 12489) n'est pas bloqué par un Firewall.

Configuration de Nagios pour surveiller vos machines Windows

Une fois le client et le serveur installé, il faut configurer Nagios de la manière suivantes. Il faut dans un premier temps éditer votre fichier de configuration des hosts (hosts.cfg par défaut) et y ajouter votre machine Windows:

```

define host {
    use generic-host
    host_name billgates
    alias Ma machine Win
    address      192.168.6.66
}

```

Puis ajouter les services offerts par NSClient (dans le fichier services.cfg):

```

# Affiche la version du NSClient
define service {
    use generic-service
    host_name billgates
    service_description VERSION
    check_command check_nt!CLIENTVERSION
}

# Temps écoulé depuis le dernier reboot (uptime)
define service {
    use generic-service
    host_name billgates
    service_description UPTIME
    check_command check_nt!UPTIME
}

# Charge CPU
# WARNING si charge > 80% pendant plus de 5 minutes
# CRITICAL si charge > 90% pendant plus de 5 minutes
define service {
    use generic-service
    host_name billgates
    service_description CPU
    check_command check_nt!CPULOAD!-1 5,80,90
}

# Etat de la mémoire vive libre
# WARNING si mémoire > 80%
# CRITICAL si mémoire > 90%
define service {
    use generic-service
    host_name billgates
    service_description MEM
    check_command check_nt!MEMUSE!-w 80 -c 90
}

# Etat de la mémoire disque libre (sur disque c:)

```

```

# WARNING si mémoire > 80%
# CRITICAL si mémoire > 90%
define service {
    use generic-service
    host_name billgates
    service_description DISK
    check_command check_nt!USEDISKSPACE!-1 c -w 80 -c 90
}

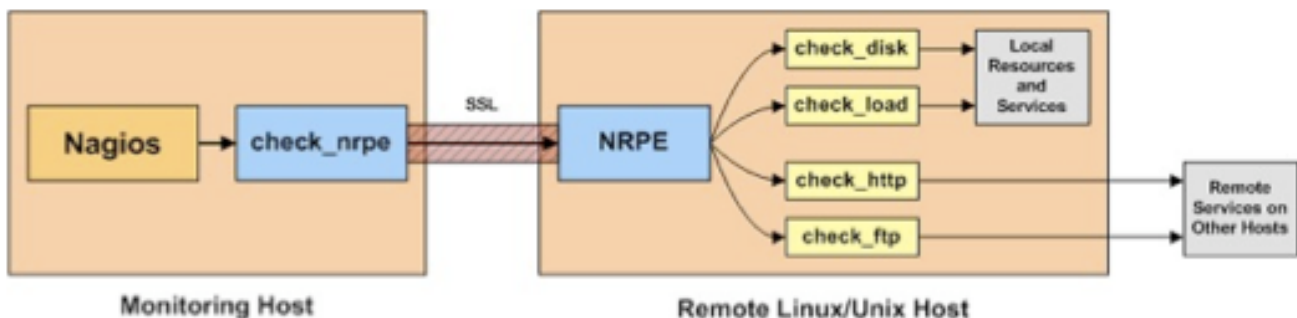
```

Il est également possible de surveiller l'état d'un service (SERVICESTATE) ou d'un processus (PROCSTATE).

Surveiller vos serveurs GNU/Linux avec Nagios

Installation de NRPE depuis les sources

Afin de disposer de la dernière version de NRPE (le plugin pour superviser vos serveurs GNU/Linux, BSD ou Mac OS X sous Nagios), il est parfois nécessaire de la compiler depuis les sources. Voici donc une simple procédure pour installer NRPE 2 et les plugins Nagios "standards" sous une distribution GNU/Linux.



Récupération des sources

Nous partons, bien sûr, sur l'hypothèse où votre machine cible (c'est à dire celle où vous aller compiler NRPE) dispose des logiciels de développement de base (configure, make, gcc...).

Si votre machine dispose d'un accès internet, vous pouvez saisir les commandes suivantes (en remplaçant les numéros de versions par les dernières disponibles):

```

wget http://surfnet.dl.sourceforge.net/sourceforge/nagios/nrpe-2.12.tar.gz
wget http://heanet.dl.sourceforge.net/sourceforge/nagiosplug/nagios-plugins-1.4.13.tar.gz

```

Préalablement à l'installation de NRPE, il faut créer un utilisateur 'nagios' sur votre machine:

adduser nagios

Pour des raisons de sécurité, il est préférable que cet utilisateur n'est pas de shell:

```
vipw
Remplacer la ligne:
nagios:x:500:500::/home/nagios:/bin/bash
Par:
nagios:x:500:500::/home/nagios:/bin/noshell
```

Installation de NRPE

On lance la fameuse séquence:

```
tar zxvf nrpe-2.12.tar.gz
cd nrpe-2.12
./configure
make all
make install
```

Lors de la compilation il est possible qu'il manque des dépendances. Par exemple, si vous avez le message suivant:

```
checking for SSL headers... configure: error: Cannot find ssl headers
```

Il faut installer les bibliothèques SSL (libssl-dev sous Ubuntu):

```
apt-get install libssl-dev
```

Installation des plugins Nagios standards

Pareil que miguel...:

```
tar zxvf nagios-plugins-1.4.13.tar.gz
cd nagios-plugins-1.4.13
./configure
make install
```

Puis une initialisation du script de configuration (/usr/local/nagios/etc/nrpe.conf):

```
mkdir /usr/local/nagios/etc
cp sample-config/nrpe.cfg /usr/local/nagios/etc
```

Correction des droits sur les fichiers

De base, les plugins sont installés avec les droits de l'utilisateur qui à lancé la compilation. Pour être sûr que NRPE puisse lancer les plugins, on doit saisir la commande suivante:

```
chown -R nagios:nagios /usr/local/nagios/
```

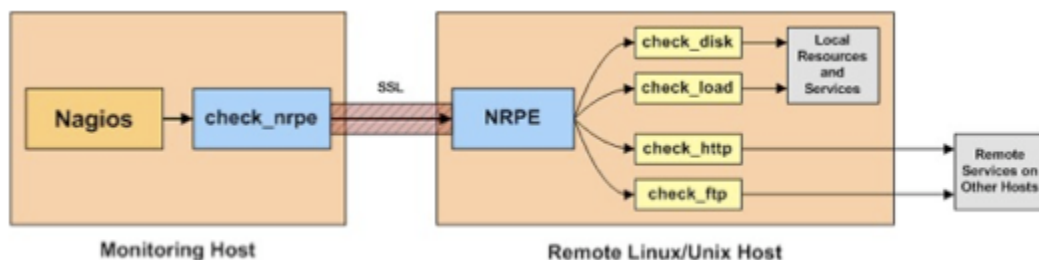
Lancement automatique au démarrage

Un script standard est fourni dans les sources:

```
cp init-script /etc/init.d/nrpe  
chmod 755 /etc/init.d/nrpe
```

Surveiller vos serveurs Linux avec Nagios et NRPE

Suite à l'introduction sur [les greffons Nagios](#), voici une simple procédure pour mettre en place le monitoring de serveurs sous Linux (voir ce billet pour [des serveurs BSD ou Mac OS X](#)) à partir de Nagios en utilisant le plugin NRPE.



Sur votre serveur Nagios...

... il faut installer le plugin NRPE. Pour cela, le plus simple est de faire confiance à votre gestionnaire de paquets.

Sous Fedora, la commande suivante devrait suffire:

```
sudo yum install nagios-plugins-nrpe
```

Sous Ubuntu/Debian:

```
sudo apt-get install nagios-nrpe-plugin
```

Il faut également vérifier que la définition du plugin est bien présente dans le fichier de configuration des commandes (commands.cfg):

```
...
#####
# NRPE
#####
# 'check_nrpe' command definition
define command{
    command_name check_nrpe
    command_line $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
}

```

Sur votre serveur Linux à surveiller...

La procédure est un peu plus longue. Il faut d'abord installer le daemon NRPE et les plugins Nagios (qui vont être lancés localement par le daemon NRPE):

Sous Fedora:

```
# sudo yum install nrpe
# sudo yum install nagios-plugins-all
```

Sous Ubuntu/Debian:

```
# sudo apt-get install nagios-nrpe-server
# sudo apt-get install nagios-plugins
```

Puis éditer le fichier /etc/nagios/nrpe.cfg pour modifier la ligne suivante:

```
...
allowed_hosts = Mettre ici l'adresse IP de votre serveur Nagios
...

```

On automatise le lancement du daemon au démarrage du serveur avec la commande:

```
# chkconfig --add nrpe
```

On ajoute une règle pour autoriser le Firewall IPtable à laisser passer les requêtes NRPE (à adapter selon vos règles):

```
# iptables -I RH-Firewall-1-INPUT 10 -p tcp --dport 5666 -j ACCEPT
```

Il ne reste plus qu'à lancer le daemon:

Sous Fedora:

```
# service nrpe start
```

Sous Ubuntu/Debian:

```
# /etc/init.d/nagios-nrpe-server start
```

On teste la communication...

Pour tester que la communication entre le serveur Nagios et le serveur à surveiller se passe bien, il suffit de se rendre dans le répertoire des plugins (`/usr/lib/nagios/plugins`) de Nagios et de tester le plugin NRPE:

```
# ./check_nrpe -H Adresse_IP_du_serveur_Linux  
NRPE v2.7
```

Si tout est OK, cette commande devrait renvoyer la version du daemon NRPE. Vous pouvez tester directement les plugins avec la commande suivante (exemple donnée pour un check de la charge):

```
# ./check_nrpe -H Adresse_IP_du_serveur_Linux -c check_load
```

On configure Nagios...

La dernière étape consiste à modifier les fichiers de configuration de Nagios pour intégrer le monitoring du/des serveur Linux. Il faut dans un premier temps éditer votre fichier de configuration des hosts (`hosts.cfg` par défaut) et y ajouter votre machine Linux:

```
define host {  
    use generic-host  
    host_name linus  
    alias Ma machine Linux  
    address 192.168.0.7  
}
```

Puis ajouter les services offerts par NRPE (dans le fichier `services.cfg`), quelques exemples:

```

# Charge CPU
define service{
    use generic-service
    host_name remotehost
    service_description CPU Load
    check_command check_nrpe!check_load
}

# Memoire
define service{
    use generic-service
    host_name remotehost
    service_description Memory
    check_command check_nrpe!check_mem
}

```

Pour ajouter des nouveaux plugins executable par NRPE, il faut éditer le fichier `/etc/nagios/nrpe.cfg` et ajouter une ligne par service:

```

...
command[check_disk]=/usr/lib/nagios/plugins/check_disk -w 20 -c 10 -p /dev/hda
...

```

Ne pas oublier de relancer le daemon quand on change le fichier de configuration (`nrpe.cfg`):

```
# service nrpe restart
```

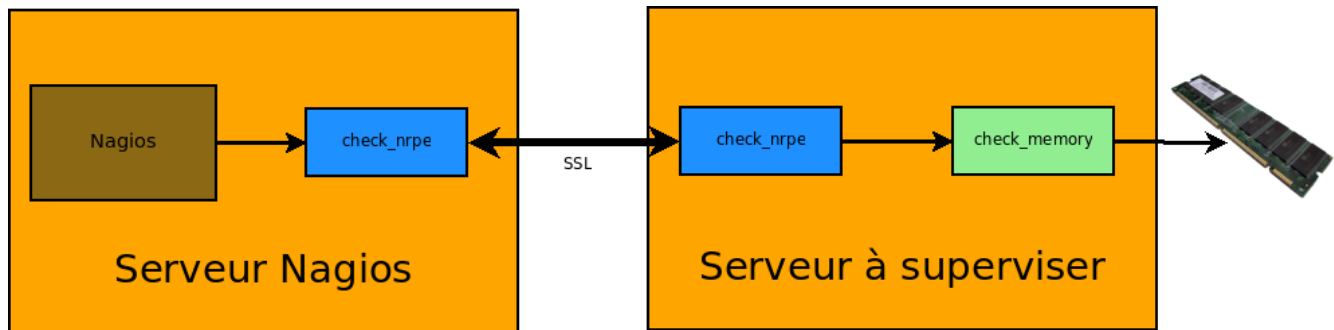
Il est bien entendu possible **d'écrire son propre plugin Nagios** et de le faire exécuter par NRPE.

Surveiller la mémoire de vos serveurs avec Nagios

Voici un petit billet de plus sur **Nagios**. Nous aborderons le sujet de la supervision à distance de la mémoire vive (RAM) de nos serveurs.

Nous allons pour cela utiliser deux plugins, le premier est NRPE (il permet de lancer des commandes à distance sur d'autres machines), le second est un script permettant d'obtenir un état de la mémoire vive à un instant "t".

Nous utiliserons donc le schéma suivant:



Installation de NRPE

Ce premier plugin doit être installé sur le serveur Nagios et sur toutes les machines à superviser.

Pour une procédure d'installation sous GNU/Linux, lire [ce billet](#). Si vous voulez surveiller des machines FreeBSD, lire [celui là](#).

A ce stade vous devez donc avoir un plugin NRPE opérationnel entre votre serveur Nagios et les machines à surveiller.

Installation du plugin de supervision de la mémoire

Il existe un grand nombre de méthodes pour obtenir la mémoire libre sur une machine. Personnellement j'utilise les deux scripts suivants selon que je sois:

- sous Linux: [check_memory](#)
- ou sous FreeBSD: [check_mem](#)

Mais, si le coeur vous en dit, vous pouvez [écrire vous-même](#) le plugin le plus adapté à votre système (Windows, Unix divers et varié...).

Une fois connecté sur votre machine à superviser, il faut mettre le plugin dans le répertoire `/usr/local/libexec/nagios/` et lui donner les droits en lecture et exécution:

```
cp ./check_memory.pl /usr/local/libexec/nagios/
chmod 555 /usr/local/libexec/nagios/check_memory.pl
```

Vous pouvez tester localement le plugin grâce la commande suivante:

```
/usr/local/libexec/nagios/check_memory.pl -f -w 90 -c 50
Memory WARNING - 87.5% (1879588864 kB) free |pct=87.5
```

Configuration de NRPE pour prendre en compte le script `check_memory`

Toujours sur la machine cible, il faut éditer le fichier de configuration de NRPE pour y ajouter la définition du plugin `check_memory`:

```
vi /usr/local/etc/nrpe.cfg
...
command[check_mem]=/usr/local/libexec/nagios/check_memory.pl -f -w 30 -c 15
...
```

Dans la configuration ci-dessus, on demande à `check_memory` de déclencher une alerte (*warning*) si la mémoire passe en dessous des 30% et une erreur (*critical*) si elle descend en dessous des 15%. Avous d'adpater ces valeurs selon vos besoins. Pour que NRPE prenne en compte la nouvelle configuration il faut relancer le daemon:

```
Sous Linux:
    service nrpe2 restart
ou
    /etc/init.d/nrpe2 restart
```

```
Sous FreeBSD:
    /usr/local/etc/rc.d/nrpe2 restart
```

Configuration du serveur Nagios

Il ne reste plus qu'à modifier la configuration du serveur Nagios pour ajouter le service à superviser, voici un exemple:

```
# Mon serveur
define host{
    use          generic-host
    host_name    monserveur
    alias        Mon beau serveur
    address      192.168.0.200
}
define service{
    use          generic-service
    host_name    monserveur
    service_description    Memoire vive
    check_command    check_nrpe!check_mem
}
```

Et voila le résultat:

Service State Information

Current Status:	OK (for 0d 0h 45m 54s)
Status Information:	Memory OK - 87.5% (1879011328 kB) free
Performance Data:	pct=87.5
Current Attempt:	1/3 (HARD state)
Last Check Time:	07-08-2008 16:05:52
Check Type:	ACTIVE
Check Latency / Duration:	0.169 / 0.162 seconds
Next Scheduled Check:	07-08-2008 16:15:52
Last State Change:	07-08-2008 15:25:52
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	07-08-2008 16:11:43 (0d 0h 0m 3s ago)

THE END !