

Superviser le SI de votre entreprise

Nagios

Table des matières

I. Introduction	3
II. La solution Nagios	3
A. Présentation	3
B. Quelques fonctionnalités de Nagios :.....	4
C. Concepts et principe de fonctionnement de Nagios :	4
III. Installation de Nagios	5
IV. Installation des « plugins » de Nagios	11
V. Configuration de Nagios.....	12
VI. Installation du « plugin » NRPE	15
A. Installation sur l'hôte à superviser.....	15
B. Installation sur le serveur Nagios.....	16
VII. Description des différentes sondes utilisées ...	17
VIII. Réception des alertes de Nagios par mail	18

I. Introduction

La supervision informatique désigne l'ensemble des outils et ressources déployés pour veiller au bon fonctionnement du système d'information. Le but de la supervision est de mettre en place une maintenance préventive afin d'éviter les interruptions de service et de détecter en amont les failles des infrastructures informatiques. Elle se caractérise par son système d'alerte et permet de récupérer l'état d'un service à l'instant T et vise à répondre à la question : Le service est-il joignable. Une extension de la supervision porte sur le fait de récupérer une valeur chiffrée comme une charge mémoire ou CPU et de lui appliquer un seuil d'alerte. Dans ce cas, aucun historique n'est gardé mais l'on est capable d'obtenir et de surveiller non plus des états, mais des valeurs numériques. On peut alors avertir l'administrateur si un système passe de UP à DOWN et inversement.

Ainsi, un service de supervision permet d'avoir un système d'information opérationnel et disponible. La surveillance des équipements informatiques permet de détecter toute anomalie en temps réel, et de pouvoir ainsi la traiter dans les meilleurs délais. On peut surveiller des éléments du système d'information tel que :

- Les sauvegardes
- Les bases de données
- Les applications
- Les espaces disques et capacités de stockage
- Les contrôles des services Windows et/ou Linux
- Les sites webs et leurs disponibilités
- Les protections antivirus
- Les serveurs, les postes de travail
- L'espace de stockage
- Les performances du système d'information

Dans notre cas nous installerons Nagios et ses plugins sur un serveur appelé serveur « supervision » et un site Wordpress sur un serveur appelé « production ». On pourra par la suite superviser les deux serveurs via l'outil de supervision Nagios dont nous détaillerons l'installation et l'utilisation par la suite.

II. La solution Nagios

A. Présentation

Nagios est un logiciel libre de surveillance des réseaux et systèmes enregistré sous la Licence GNU GPL (General Public License) version 2. Ce qui donne la permission légale de le copier, le distribuer et/ou de le modifier sous certaines conditions. Il permet de surveiller activement et passivement les hôtes et services spécifiés dans son fichier de configuration, et d'alerter les administrateurs systèmes et réseaux en cas d'événement (*Mauvais ou Bon*). Anciennement appelé NetSaint, Nagios à l'origine était destiné uniquement pour les systèmes Linux, mais actuellement, Nagios est compatible tous systèmes. A ce jour il existe deux versions de Nagios, ici nous utiliserons la version appelée Nagios Core.

B. Quelques fonctionnalités de Nagios :

- Surveillance des services réseaux tels que : SMTP, HTTP, FTP, SSH, etc.
- Surveillance des ressources machines telles que : Charge de processeur, Utilisation de l'espace disque
- Utilisation de la mémoire, etc.
- Rotation automatique des fichiers journaux
- Interface Web optionnelle permettant de visualiser l'état actuelle du réseau, les notifications et les fichiers journaux
- Conception des simples « plugins » permettant aux utilisateurs de développer leurs propres vérificateurs de services
- Notification par mail ou sms lorsqu'un problème survient sur un service ou une machine
- Support pour l'implémentation d'un système de surveillance redondant

C. Concepts et principe de fonctionnement de Nagios :

Nagios ne possède aucun mécanisme interne pour surveiller le statut des équipements et des applications. Il repose sur des programmes externes appelés « plugins ». Nagios peut être assimilé à un planificateur de tâches. Il exécute un « plugin » à intervalle régulier lorsqu'un service ou un host doit être surveillé.

Architecture de Nagios :

Nagios peut être décomposé en trois parties :

- Un ordonnanceur, chargé de contrôler quand et dans quel ordre les contrôles des services sont effectués.
- Une interface graphique qui affiche de manière claire et concise l'état des services surveillés.
- Des « plugins »

Nagios Remote Plugin Executor ou NRPE :

Nagios Remote Plugin Executor (NRPE) est un agent Nagios qui permet de surveiller le système à distance à l'aide de scripts hébergés sur les systèmes distants. Il permet de surveiller les ressources telles que l'utilisation du disque, la charge du système ou le nombre d'utilisateurs actuellement connectés. Nagios interroge périodiquement l'agent sur le système distant en utilisant le plugin « check_nrpe ».

Les Plugins Nagios :

Les « plugins » sont des programmes compilés ou des scripts (Perl, Shell, Python, etc.) qui peuvent être exécutés par une ligne de commande pour contrôler l'état d'un hôte ou d'un service. Nagios utilise le résultat des « plugins » pour déterminer le statut actuel des hôtes ou services sur le réseau. Nagios exécute un « plugin » seulement lorsqu'il est nécessaire de vérifier le statut d'un service ou d'un hôte. Les « plugins » sont comme une couche intermédiaire. L'avantage de ce type d'architecture de « plugin » est que l'on peut superviser à peu près tout ce que l'on veut. Si l'on peut automatiser le contrôle d'un service et/ou d'une ressource, on peut le superviser avec Nagios.

III. Installation de Nagios

Nous allons maintenant installer Nagios Core sur un système GNU/Linux (Debian 10) depuis les sources. Tout d'abord, nous mettons à jour la liste des paquets de Debian 10 avec la commande :

`apt-get update`

```
Debian GNU/Linux 10 supervision tty1

supervision login: root
Password:
Linux supervision 4.19.0-17-amd64 #1 SMP Debian 4.19.194-3 (2021-07-18) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@supervision:~# apt-get update
Atteint :1 http://security.debian.org/debian-security buster/updates InRelease
Atteint :2 http://deb.debian.org/debian buster InRelease
Atteint :3 http://deb.debian.org/debian buster-updates InRelease
Lecture des listes de paquets... Fait
root@supervision:~# _
```

Puis pour accéder à l'interface Web de gestion de Nagios, nous aurons besoin d'un serveur Apache et de l'interpréteur PHP que nous installons avec la commande suivante :

`apt-get install apache2 php php-gd php-imap php-curl`

```
root@supervision:~# apt-get install apache2 php php-gd php-imap php-curl
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  apache2-bin apache2-data apache2-utils fontconfig-config fonts-dejavu-core libapache2-mod-php7.3
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libbrotli1 libcurl4 libfontconfig1 libgd3 libjansson4 libjpeg62-turbo liblua5.2-0 libsodium23
  libtiff5 libwebp6 libxpm4 mlock php-common php7.3 php7.3-cli php7.3-common php7.3-curl php7.3-gd
  php7.3-imap php7.3-json php7.3-opcache php7.3-readline psmisc ssl-cert
Paquets suggérés :
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser php-pear uw-mailutils
  libgd-tools openssl-blacklist
Les NOUVEAUX paquets suivants seront installés :
  apache2 apache2-bin apache2-data apache2-utils fontconfig-config fonts-dejavu-core
  libapache2-mod-php7.3 libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libbrotli1
  libcurl4 libfontconfig1 libgd3 libjansson4 libjpeg62-turbo liblua5.2-0 libsodium23
  libtiff5 libwebp6 libxpm4 mlock php-common php-curl php-gd php-imap php7.3
  php7.3-cli php7.3-common php7.3-curl php7.3-gd php7.3-imap php7.3-json php7.3-opcache
  php7.3-readline psmisc ssl-cert
0 mis à jour, 41 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 10,6 Mo dans les archives.
Après cette opération, 36,5 Mo d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n]
```

Nous installons maintenant quelques bibliothèques Perl supplémentaires pour pouvoir exploiter les « plugins » de Nagios avec la commande suivante :

`apt-get install libxml-libxml-perl libnet-snmp-perl libperl-dev libnumber-format-perl libconfig-inifiles-perl libdatetime-perl libnet-dns-perl`

```

libref-util-perl libref-util-xs-perl librole-tiny-perl libspecio-perl libsub-exporter-perl
libsub-exporter-progressive-perl libsub-identify-perl libsub-install-perl libsub-name-perl
libsub-quote-perl libtime-date-perl libtry-tiny-perl liburi-perl libvariable-magic-perl
libwww-perl libwww-robotrules-perl libxml-namespacesupport-perl libxml-parser-perl
libxml-sax-base-perl libxml-sax-expat-perl libxml-sax-perl linux-libc-dev manpages-dev
perl-openssl-defaults
Paquets suggérés :
libgssapi-perl libc-dev libc6-dev libclass-c3-perl libclass-c3-xs-perl libclass-data-inheritable-perl
libclass-inspector-perl libclass-method-modifiers-perl libclass-singleton-perl
libclass-xsaccessor-perl libconfig-inifiles-perl libdata-dump-perl libdata-optlist-perl
libdatetime-locale-perl libdatetime-perl libdatetime-timezone-perl libdevel-callchecker-perl
libdevel-caller-perl libdevel-lexalias-perl libdevel-stacktrace-perl libdigest-bubblebabble-perl
libdigest-hmac-perl libdynamloader-functions-perl libencode-locale-perl libeval-closure-perl
libexception-class-perl libfile-listing-perl libfile-sharedir-perl libfont-afm-perl
libhtml-form-perl libhtml-format-perl libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl
libhttp-cookies-perl libhttp-daemon-perl libhttp-date-perl libhttp-message-perl
libhttp-negotiate-perl libio-html-perl libio-socket-ssl-perl liblwp-mediatypes-perl
liblwp-protocol-https-perl libmailtools-perl libmodule-implementation-perl
libmodule-runtime-perl libmq-compat-perl libnamespace-autoclean-perl libnamespace-clean-perl
libnet-dns-perl libnet-dns-sec-perl libnet-http-perl libnet-ip-perl libnet-libidn-perl
libnet-smtp-ssl-perl libnet-snmp-perl libnet-ssleay-perl libnumber-format-perl
libpackage-stash-perl libpackage-stash-xs-perl libpadwalker-perl libparams-classify-perl
libparams-util-perl libparams-validationcompiler-perl libperl-dev libreadonly-perl
libref-util-perl libref-util-xs-perl librole-tiny-perl libspecio-perl libsub-exporter-perl
libsub-exporter-progressive-perl libsub-identify-perl libsub-install-perl libsub-name-perl
libsub-quote-perl libtime-date-perl libtry-tiny-perl liburi-perl libvariable-magic-perl
libwww-perl libwww-robotrules-perl libxml-libxml-perl libxml-namespacesupport-perl
libxml-parser-perl libxml-sax-base-perl libxml-sax-expat-perl libxml-sax-perl linux-libc-dev
manpages-dev perl-openssl-defaults
0 mis à jour, 94 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 17,0 Mo dans les archives.
Après cette opération, 89,7 Mo d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n] _

```

On installe maintenant les bibliothèques graphiques avec la commande suivante :

```
apt-get install libpng-dev libjpeg-dev libgd-dev
```

Nous installons enfin des outils de compilation standards et le package unzip :

```
apt-get install gcc make autoconf libc6 unzip
```

```

root@supervision:~# apt-get install gcc make autoconf libc6 unzip
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
libc6 est déjà la version la plus récente (2.28-10).
Les paquets supplémentaires suivants seront installés :
  automake autotools-dev binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-8 gcc-8
  libasan5 libatomic1 libbinutils libcc1-0 libgcc-8-dev libgomp1 libisl19 libitm1 liblsan0 libmpc3
  libmpfr6 libmpx2 libquadmath0 libsigsegv2 libtsan0 libubsan1 m4
Paquets suggérés :
  autoconf-archive gnu-standards autoconf-doc libtool gettext binutils-doc cpp-doc gcc-8-locales
  gcc-multilib flex bison gdb gcc-doc gcc-8-multilib gcc-8-doc libgcc1-dbg libgomp1-dbg
  libitm1-dbg libatomic1-dbg libasan5-dbg liblsan0-dbg libtsan0-dbg libubsan1-dbg libmpx2-dbg
  libquadmath0-dbg m4-doc make-doc zip
Les NOUVEAUX paquets suivants seront installés :
  autoconf automake autotools-dev binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-8 gcc
  gcc-8 libasan5 libatomic1 libbinutils libcc1-0 libgcc-8-dev libgomp1 libisl19 libitm1 liblsan0
  libmpc3 libmpfr6 libmpx2 libquadmath0 libsigsegv2 libtsan0 libubsan1 m4 make unzip
0 mis à jour, 29 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 29,7 Mo dans les archives.
Après cette opération, 111 Mo d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n] _

```

Nous allons maintenant créer un utilisateur « nagios » :

```
useradd -m -p $(openssl passwd nagios) nagios
```

Et un groupe appelé nagcmd qui servira à exécuter les commandes Nagios depuis l'interface web :

```
groupadd nagcmd
```

On ajoute ensuite l'utilisateur « nagios » dans le groupe « nagcmd » :

```
usermod -a -G nagcmd nagios
```

Enfin on ajoute l'utilisateur « www-data », sous lequel tournent les processus du serveur Web Apache2 dans le groupe « nagcmd » :

```
usermod -a -G nagcmd www-data
```

```
root@supervision:~# useradd -m -p $(openssl passwd nagios) nagios
root@supervision:~# groupadd nagcmd
root@supervision:~# usermod -a -G nagcmd nagios
root@supervision:~# usermod -a -G nagcmd www-data
root@supervision:~#
```

On crée maintenant un dossier nommé « downloads » qui accueillera les sources de Nagios et de ses « plugins » :

```
mkdir /home/nagios/downloads
```

On se rend maintenant dans le dossier « downloads » pour y télécharger les sources de Nagios depuis son site officiel :

```
cd /home/nagios/downloads
```

```
wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
```

```
root@supervision:~# cd /home/nagios/downloads
root@supervision:/home/nagios/downloads# wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
--2021-07-20 21:19:31-- https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
Résolution de assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connexion à assets.nagios.com (assets.nagios.com)[45.79.49.120]:443... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : 11333414 (11M) [application/x-gzip]
Sauvegarde en : « nagios-4.4.6.tar.gz »

nagios-4.4.6.tar.gz 100%[=====] 10,81M 1,61MB/s ds 7,1s

2021-07-20 21:19:39 (1,51 MB/s) - « nagios-4.4.6.tar.gz » sauvegardé [11333414/11333414]
root@supervision:/home/nagios/downloads#
```

On décompresse l'archive des sources de Nagios avec la commande suivante :

```
tar -zxvf nagios-4.4.6.tar.gz
```

Et on se rend dans le dossier avec la commande suivante :

```
cd nagios-4.4.6
```

On utilise ensuite le script `configure`. Celui-ci permet, entre autres, de s'assurer que les éléments nécessaires sont présents sur le système, et de passer quelques paramètres au processus de compilation. Dans notre cas, nous allons indiquer deux choses : où se situe le répertoire par défaut de configuration des sites Web, et où se situe le groupe nagcmd que nous souhaitons configurer avec la commande suivante :

```
./configure --with-httpd-conf=/etc/apache2/sites-enabled --with-command-group=nagcmd
```

Lors de la compilation d'un paquet, `configure` vérifie que :

- qu'un compilateur est présent pour le langage utilisé dans les sources
- la présence des headers et la lib nécessaire à la compilation / exécution.

Si tout est bon il génère un fichier MakeFile.

Si tout se passe bien, on obtient :

```
*** Configuration summary for nagios 4.4.6 2020-04-28 ***:

General Options:
-----
Nagios executable: nagios
Nagios user/group: nagios,nagios
Command user/group: nagios,nagcmd
Event Broker: yes
Install ${prefix}: /usr/local/nagios
Install ${includedir}: /usr/local/nagios/include/nagios
Lock file: /run/nagios.lock
Check result directory: /usr/local/nagios/var/spool/checkresults
Init directory: /lib/systemd/system
Apache conf.d directory: /etc/apache2/sites-enabled
Mail program: /bin/mail
Host OS: linux-gnu
IOBroker Method: epoll

Web Interface Options:
-----
HTML URL: http://localhost/nagios/
CGI URL: http://localhost/nagios/cgi-bin/
Traceroute (used by WAP): /usr/sbin/traceroute

Review the options above for accuracy. If they look okay,
type 'make all' to compile the main program and CGIs.

root@supervision:/home/nagios/downloads/nagios-4.4.6# _
```

On lance ensuite la compilation avec la commande :

`make all`

```
root@supervision:/home/nagios/downloads/nagios-4.4.6# make all
```

Dès lors qu'un makefile approprié existe, chaque fois que vous modifiez certains fichiers source cette simple commande Shell `make` suffit pour effectuer toutes les compilations nécessaires. Le programme `make` utilise la base de données makefile et l'heure de dernière modification des fichiers pour décider quels fichiers doivent être mis à jour. Pour chacun de ces fichiers, il émet les commandes enregistrées dans la base de données.

Si tout se passe bien, la fin de sortie de cette commande doit ressembler à :

```
*****
Enjoy.

root@supervision:/home/nagios/downloads/nagios-4.4.6#
```


On saisit `make install`. Cette commande invoque à nouveau `make`, qui recherche la cible `install` dans le Makefile et suit les instructions pour installer le programme :
`make install`

Et on obtient une sortie de commande ressemblant à cela :

```
*** Main program, CGIs and HTML files installed ***

You can continue with installing Nagios as follows (type 'make'
without any arguments for a list of all possible options):

make install-init
- This installs the init script in /lib/systemd/system

make install-commandmode
- This installs and configures permissions on the
  directory for holding the external command file

make install-config
- This installs sample config files in /usr/local/nagios/etc

make[1] : on quitte le répertoire « /home/nagios/downloads/nagios-4.4.6 »
root@supervision:/home/nagios/downloads/nagios-4.4.6#
```

Ensuite, on installe le service Nagios, nécessaire au démarrage de Nagios avec la machine en lançant la commande suivante :
`make install-daemoninit`

```
root@supervision:/home/nagios/downloads/nagios-4.4.6# make install-daemoninit
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /lib/systemd/system/nagios.service.

*** Init script installed ***

root@supervision:/home/nagios/downloads/nagios-4.4.6#
```

On installe le « pipe » de Nagios avec la commande suivante :
`make install-commandmode`

```
root@supervision:/home/nagios/downloads/nagios-4.4.6# make install-commandmode
/usr/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

root@supervision:/home/nagios/downloads/nagios-4.4.6#
```

Puis les fichiers de configuration de base de Nagios avec la commande suivante :
`make install-config`

```
*** Config files installed ***

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

root@supervision:/home/nagios/downloads/nagios-4.4.6# _
```

Et on installe l'interface Web d'administration de Nagios avec la commande suivante :
`make install-webconf`

Puis on active les modules `rewrite` et `cgi` d'Apache avec les commandes suivantes :
`a2enmod rewrite`
`a2enmod cgi`

```
root@supervision:/home/nagios/downloads/nagios-4.4.6# make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/apache2/sites-enabled/nagios.conf
if [ 0 -eq 1 ]; then \
    ln -s /etc/apache2/sites-enabled/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

root@supervision:/home/nagios/downloads/nagios-4.4.6# a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
    systemctl restart apache2
root@supervision:/home/nagios/downloads/nagios-4.4.6# a2enmod cgi
Enabling module cgi.
To activate the new configuration, you need to run:
    systemctl restart apache2
root@supervision:/home/nagios/downloads/nagios-4.4.6# _
```

Nous allons maintenant configurer l'accès à Apache pour accéder à l'interface d'administration de Nagios. Il faut configurer un accès Apache `htaccess`. Pour se faire on lance la commande suivante :

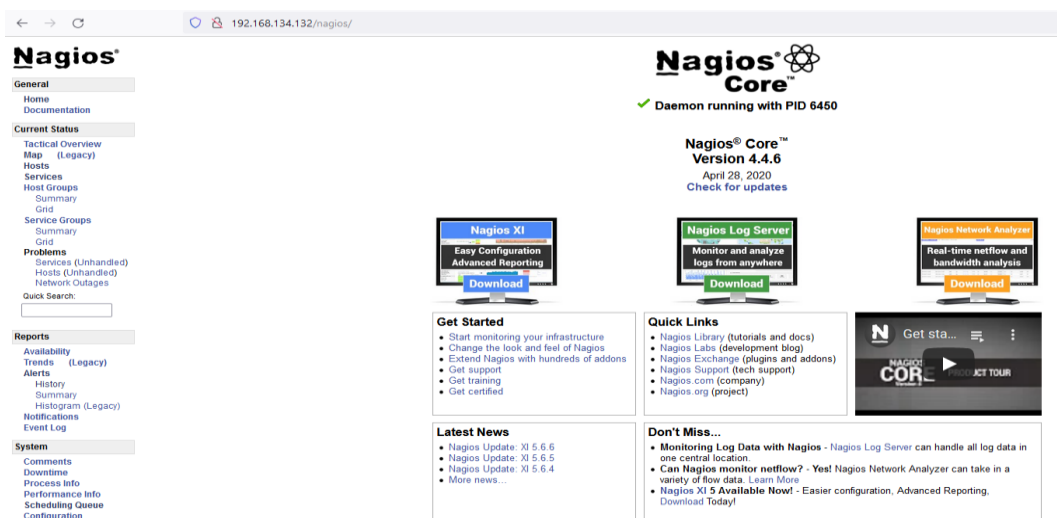
`htpasswd -cb /usr/local/nagios/etc/htpasswd.users nagiosadmin motdepasse`

On configure l'utilisateur appelé « nagiosadmin » avec le mot de passe « motdepasse ».

Enfin, il faut redémarrer le service Apache et démarrer le service Nagios. Pour cela, lancez les commandes suivantes :

`systemctl restart apache2`
`systemctl start nagios`

L'installation de Nagios Core est maintenant terminée et on peut se connecter à l'interface d'administration de Nagios en saisissant dans un navigateur l'adresse suivante :
`http://ipserveurnagios/nagios`



Dans la partie suivante, nous verrons l'installation des « plugins » de Nagios.

IV. Installation des « plugins » de Nagios

L'installation des plugins de Nagios se déroule de la même manière que l'installation de Nagios Core. On se rend d'abord dans le répertoire « downloads » précédemment créé. On télécharge les sources des « plugins » standards de Nagios, puis on décompresse l'archive téléchargée. Et on se rend dans ce dossier.

On va maintenant compiler et installer les « plugins » Nagios avec les commandes suivantes :

```
./configure --with-nagios-user=nagios --with-nagios-group=nagcmd
```

Puis on lance la commande « make » pour compiler tous les *plugins* compatibles avec le système et qui affiche une sortie comme ci-dessous :

```
make[2] : on entre dans le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3/po »
make[2]: rien à faire pour « all ».
make[2] : on quitte le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3/po »
make[2] : on entre dans le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3 »
make[2] : on quitte le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3 »
make[1] : on quitte le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3 »
root@supervision:/home/nagios/downloads/nagios-plugins-2.3.3# make
```

Et on termine l'installation des plugins en exécutant la commande :

`make install`

```
Making install in plugins-root
make[1] : on entre dans le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3/plugins-root »
make[2] : on entre dans le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3/plugins-root »
/usr/bin/install -c -o nagios -g nagcmd check_dhcp /usr/local/nagios/libexec/check_dhcp
chown root /usr/local/nagios/libexec/check_dhcp
chmod ugr-x,u+s /usr/local/nagios/libexec/check_dhcp
/usr/bin/install -c -o nagios -g nagcmd check_icmp /usr/local/nagios/libexec/check_icmp
chown root /usr/local/nagios/libexec/check_icmp
chmod ugr-x,u+s /usr/local/nagios/libexec/check_icmp
make[2]: rien à faire pour « install-data-am ».
make[2] : on quitte le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3/plugins-root »
make[1] : on quitte le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3/plugins-root »
Making install in po
make[1] : on entre dans le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3/po »
/usr/bin/mkdir -p /usr/local/nagios/share
installing fr.gmo as /usr/local/nagios/share/locale/fr/LC_MESSAGES/nagios-plugins.mo
installing de.gmo as /usr/local/nagios/share/locale/de/LC_MESSAGES/nagios-plugins.mo
if test "nagios-plugins" = "gettext-tools"; then \
  /usr/bin/mkdir -p /usr/local/nagios/share/gettext/po; \
  for file in Makefile.in.in remove-potcdate.sin Makevars.template; do \
    /usr/bin/install -c -o nagios -g nagcmd -m 644 ./.$file \
      /usr/local/nagios/share/gettext/po/$file; \
  done; \
  for file in Makevars; do \
    rm -f /usr/local/nagios/share/gettext/po/$file; \
  done; \
else \
: ; \
fi
make[1] : on quitte le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3/po »
make[1] : on entre dans le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3 »
make[2] : on entre dans le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3 »
make[2]: rien à faire pour « install-exec-am ».
make[2]: rien à faire pour « install-data-am ».
make[2] : on quitte le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3 »
make[1] : on quitte le répertoire « /home/nagios/downloads/nagios-plugins-2.3.3 »
root@supervision:/home/nagios/downloads/nagios-plugins-2.3.3# _
```

Les « plugins » de Nagios sont maintenant installés. Nous pouvons le vérifier en allant dans l'onglet « Alerte » du menu « Reports » sur la page d'administration de Nagios.

V. Configuration de Nagios

Les fichiers de configurations de Nagios se trouvent dans le dossier `/usr/local/nagios/etc/`. Dans ce dossier on retrouve le fichier `nagios.cfg`, qui est le fichier de configuration principal de Nagios. Avec ce fichier, on peut spécifier les fichiers ou les dossiers où se trouvent les fichiers de configurations des hôtes. Etant donné que pour l'instant la seule machine supervisée par Nagios est le serveur sur lequel il est installé, son fichier de configuration est activé.

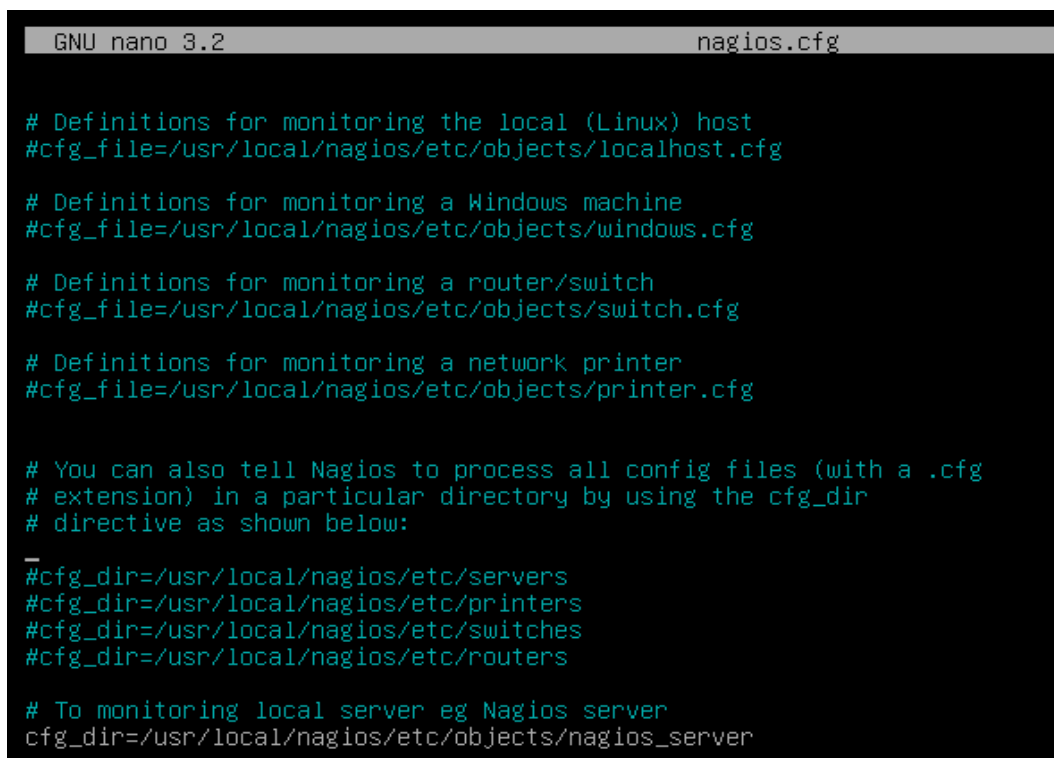
Dans le dossier `/usr/local/nagios/etc/objects/`, nous allons créer un dossier nommé `nagios_server` qui contiendra la définition de nos hôtes à superviser.

Puis nous allons éditer le fichier `nagios.cfg` se trouvant dans `/usr/local/nagios/etc` pour rajouter la ligne :

`cfg_dir=/usr/local/nagios/etc/objects/nagios_server`

Enfin on y commente la ligne :

`#cfg_file=/usr/local/nagios/etc/objects/localhost.cfg`



```
GNU nano 3.2 nagios.cfg

# Definitions for monitoring the local (Linux) host
#cfg_file=/usr/local/nagios/etc/objects/localhost.cfg

# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg

# Definitions for monitoring a router/switch
#cfg_file=/usr/local/nagios/etc/objects/switch.cfg

# Definitions for monitoring a network printer
#cfg_file=/usr/local/nagios/etc/objects/printer.cfg

# You can also tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below:
-
#cfg_dir=/usr/local/nagios/etc/servers
#cfg_dir=/usr/local/nagios/etc/printers
#cfg_dir=/usr/local/nagios/etc/switches
#cfg_dir=/usr/local/nagios/etc/routers

# To monitoring local server eg Nagios server
cfg_dir=/usr/local/nagios/etc/objects/nagios_server
```

Maintenant dans le dossier `nagios_server`, nous allons créer trois fichiers : `linux_hosts.cfg`, `linux_hostgroups.cfg` et `linux_services.cfg`

Dans le fichier `linux_hosts.cfg` on trouvera les définitions des hôtes, dans le fichier `linux_hostgroups.cfg` on trouvera les définitions des groupes et dans le fichier `linux_services.cfg`, on trouvera les définitions des services que nous allons superviser.

Voici donc le contenu du fichier `linux_hosts.cfg` où 2 machines sont supervisées :

```

GNU nano 3.2                                linux_hosts.cfg

##### LINUX HOST DEFINITIONS #####

define host{
    use                linux-server
    host_name          supervision
    alias              Serveur Supervision
    address            192.168.134.132
}

define host{
    use                linux-server
    host_name          production
    alias              Serveur Production
    address            192.168.134.131
}

```

Voici maintenant le contenu du fichier `linux_hostgroups.cfg` :

```

GNU nano 3.2                                linux_hostgroups.cfg

##### LINUX HOSTGROUPS DEFINITIONS #####

define hostgroup{
    hostgroup_name      nagios_server
    alias              Serveurs Linux
    members             supervision, production
}

```

Enfin, voici une partie du fichier `linux_services.cfg` :

```

GNU nano 3.2                                linux_services.cfg

##### LINUX SERVICES DEFINITIONS #####

# Check Uptime : temps d'activité du serveur de supervision
define service{
    use                local-service
    host_name          supervision
    service_description Uptime
    check_command       check_local_uptime
}

# Check Apt : vérification des MAJ disponibles
define service{
    use                local-service
    host_name          supervision
    service_description Apt
    check_command       check_local_apt
}

# Check Disk : vérification espace disque disponible
define service{
    use                local-service
    host_name          supervision
    service_description Espace disque
    check_command       check_local_disk!20%!10%!/!GB
}

```

Chaque service supervisé est défini de la manière suivante :

- `use` : nom du template à utiliser
- `host_name` : hostname de la machine sur laquelle on supervise le service
- `service_description` : nom du service tel qu'il apparaîtra dans Nagios
- `check_command` : commande avec laquelle nagios va checker le service.

Les différentes commandes utilisées via la commande `check_command` sont définies dans le fichier `commands.cfg` se trouvant dans le dossier `/usr/local/nagios/etc/objects/` :

```
GNU nano 3.2      commands.cfg

# These are some example service check commands.  They may or may not work on
# your system, as they must be modified for your plugins.  See the HTML
# documentation on the plugins for examples of how to configure command definitions.
#
# NOTE:  The following 'check_local_...' functions are designed to monitor
#        various metrics on the host that Nagios is running on (i.e. this one).
#####

define command {
    command_name    check_local_uptime
    command_line    $USER1$/check_uptime
}

define command {
    command_name    check_local_apd
    command_line    $USER1$/check_apd
}

define command {
    command_name    check_local_disk
    command_line    $USER1$/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$ -u $ARG4$
}

define command {
    command_name    check_local_load
    command_line    $USER1$/check_load -w $ARG1$ -c $ARG2$
}
```

Par exemple la définition de la commande du service « `uptime` » se définit dans les balises `define command {}` avec comme variables :

- `command_name` : nom de la commande, tel que précisé dans la variable `check_command` de la définition du service
- `command_line` : commandes de check du service
- La variable `$USER1$` pointe sur le dossier `/usr/local/nagios/libexec/` où sont stockés tous les scripts de check de services de Nagios

- Et éventuellement une options `-H` qui permet de stipuler l'hôte sur lequel le check du service va se faire, et les options `-w` et `-c` sont les options que l'on peut associer à la commande afin de définir des seuils d'alerte.

VI. Installation du « plugin » NRPE

NRPE pour Nagios Remote Plugin Executor est un agent de supervision qui permet de récupérer les informations à distance. Son principe de fonctionnement est simple : il suffit d'installer le « plugin » sur la machine distante et de l'interroger à partir du serveur de supervision Nagios. Le « plugin » installé sur le serveur Nagios initie une connexion sur la machine hôte distante, l'hôte distant exécute le « plugin » demandé et retourne au serveur Nagios le code de retour de l'exécution du plugin ainsi que la sortie standard.

Le gros avantage de cet agent est qu'il permet de réduire les charges sur le serveur Nagios. De plus certains « plugins » sont à exécuter obligatoirement en local.

A. Installation sur l'hôte à superviser

Pour superviser notre serveur « production », il nous faut d'abord installer le « plugin » NRPE en exécutant la commande suivante :

```
apt-get install nagios-nrpe-server
```

Nous allons maintenant éditer le fichier nrpe se trouvant dans le dossier `/etc/nagios/` :

```
nano /etc/nagios/nrpe.cfg
```

On modifie la ligne `allowed_hosts=127.0.0.1` avec l'adresse IP de notre serveur Nagios :

```
# ALLOWED HOST ADDRESSES
# This is an optional comma-delimited list of IP address or hostnames
# that are allowed to talk to the NRPE daemon. Network addresses with a bit mask
# (i.e. 192.168.1.0/24) are also supported. Hostname wildcards are not currently
# supported.
#
# Note: The daemon only does rudimentary checking of the client's IP
# address. I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

allowed_hosts=192.168.134.132,127.0.0.1
```

Plus bas dans ce fichier, on retrouve les commandes de check de NRPE. NRPE utilise les scripts de check présents dans le dossier `/usr/lib/nagios/plugins`.

```
# The following examples use hardcoded command arguments...
# This is by far the most secure method of using NRPE

command[check_users]=/usr/lib/nagios/plugins/check_users -w 0 -c 5
command[check_load]=/usr/lib/nagios/plugins/check_load -r -w .15,.10,.05 -c .30,.25,.20
command[check_hda1]=/usr/lib/nagios/plugins/check_disk -u GB -w 20% -c 10% -p /
command[check_zombie_procs]=/usr/lib/nagios/plugins/check_procs -w 5 -c 10 -s Z
command[check_total_procs]=/usr/lib/nagios/plugins/check_procs -w 150 -c 200
command[check_uptime]=/usr/lib/nagios/plugins/check_uptime
command[check_memory]=/usr/lib/nagios/plugins/check_memory -w 80 -c 90
command[check_ddos]=/usr/lib/nagios/plugins/check_ddos -c 100 -w 50
command[check_tcp]=/usr/lib/nagios/plugins/check_tcp 192.168.134.131 -p 3306
command[check_mysql_query.pl]=/usr/lib/nagios/plugins/nagios-plugins/check_mysql_query.pl -H 192.16$
```

On enregistre le fichier et on redémarre le service avec la commande suivante :
service nagios-nrpe-server restart

La configuration de l'agent NRPE sur l'hôte distant étant finit, nous allons maintenant configurer NRPE sur le serveur Nagios.

B. Installation sur le serveur Nagios

Nous allons commencer par créer la commande dans le fichier de configuration **commands.cfg** présent dans le dossier **/usr/local/nagios/etc/objects/** de la manière suivante :

```
#####
## Commande check_nrpe
define command {
    command_name        check_nrpe
    command_line         $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
}
#####
```

Nous allons ensuite rajouter les services à superviser dans le fichier contenant les services Linux. Nous allons donc éditer le fichier **linux_services.cfg** se trouvant dans le dossier **/usr/local/nagios/etc/objects/nagios_server** :

```
# Check disk production
define service{
    use                local-service
    host_name           production
    service_description Espace disque
    check_command       check_nrpe!check_hda1
}

# Check users production
define service{
    use                local-service
    host_name           production
    service_description Nombre utilisateurs
    check_command       check_nrpe!check_users
}

# Check apt production
define service{
    use                local-service
    host_name           production
    service_description MAJ
    check_command       check_nrpe!check_apt
}
```

On redémarre nagios avec la commande :

systemctl restart nagios

VII. Description des différentes sondes utilisées

Voici une liste des « plugins » ou sondes utilisés ainsi que des actions précises à entreprendre pour rétablir le service :

- **check_uptime** : permet de savoir depuis quand un serveur est en activité ou a été redémarré.
commande utilisée : `check_uptime !1 !2 (warning 1j, critique 2j)`
action entreprise : relancer par exemple le serveur Apache avec la commande « `systemctl restart apache` »
- **check_apt** : permet de vérifier les mises à jour disponibles.
commande utilisée : `check_apt`
action entreprise : utiliser la commande « `apt-update / upgrade` » pour mettre à jour le système
- **check_disk** : permet de vérifier l'espace disque disponible.
commande utilisée : `check_disk !20% !10% !/ !GB (warning -20% espace libre, critique -10% d'espace libre)`
action entreprise : utilisation de la commande « `ncdu` » pour analyser le système de fichier et suppression des données et/ou logs qui ne servent plus à rien.
- **check_load** : permet de vérifier la charge moyenne du CPU.
commande utilisée : `check_load !15,10,5 !30,25,20`
action entreprise : analyse des applications, ajout de vcore si possible
- **check_memory** : permet de vérifier la quantité de mémoire disponible et utilisée.
commande utilisée : `check_memory !80 !90 (warning 80% ram utilisée, critique 90% ram utilisée)`
action entreprise : augmenter la mémoire, analyse des applications pour essayer d'utiliser moins de mémoire (fuite de mémoire), ajout de mémoire
- **check_swap** : permet de vérifier la quantité de swap disponible.
commande utilisée : `check_swap !20 !10 (warning à -20%, critique à -10%)`
action entreprise : repartitionnement
- **check_procs** : permet de vérifier tous les processus actifs.
commande utilisée : `check_procs -w 150 -c 200`
action entreprise : vérifier les processus « zombies » et les « kill »
- **check-host-alive** : permet de vérifier si un hôte est toujours joignable.
commande utilisée : `check-host-alive !192.168.134.131 !60.0 !80.0 !5`
action entreprise : vérification de l'hôte
- **check_http** : permet de tester le service HTTP sur un hôte spécifié.
commande utilisée : `check_http !192.168.134.132`
action entreprise : redémarrer le serveur Apache avec la commande « `systemctl restart apache` », vérification des erreurs sur les log Apache
- **check_users** : permet de vérifier le nombre d'utilisateurs actuellement connectés.
commande utilisée : `check_users !1 !5 (warning à plus de 1 utilisateur connecté, critique à plus de 5 utilisateurs connectés)`
action entreprise : réduction du nombre d'utilisateurs connectés simultanément)

- `check_ddos` : permet de détecter une attaque DDOS.
commande utilisée : `check_ddos -w 50 -c 100`
action entreprise : bloquer les adresses IP, plusieurs serveurs pour un même service, serveur tampon
- `check_mysql_query` : permet d'afficher le résultat d'une requête.
commande utilisée : `check_mysql_query.pl -H 192.168.134.131 -u root -p root -d wordpress -q 'SELECT COUNT(c.comment_date) FROM wp_comments as c where hour(timediff(localtime(),c.comment_date)) < 4' -w4 -c10 (warning à 4 commentaires, critique à 10 commentaires en 4h)`
action entreprise : vérifier la longueur de la requête et analyse des logs slow query, blocage des commentaires, bannissement IP
- `check_tcp` : permet de tester la connexion à MySQL.
commande utilisée : `check_tcp 192.168.134.131 -p 3306`
action entreprise : redémarrer le serveur MySQL avec la commande « `systemctl restart mariadb` »

VIII. Réception des alertes de Nagios par mail

Afin de recevoir les notifications d'alerte de Nagios sur notre messagerie électronique nous devons configurer sur notre serveur Nagios un système d'émission de mails d'alerte ou de notification via le service Postfix.

Nous allons commencer par installer un ensemble de packages utiles au bon fonctionnement de Postfix sur notre serveur via la commande :

```
apt-get install postfix mailutils libsasl2-2 ca-certificates libsasl2-modules
```

Pendant l'installation des packages indiqués, une fenêtre de configuration de Postfix apparaît et nous devons renseigner quelques informations nécessaires pour la suite comme le type de serveur de messagerie à mettre en place, ou encore le nom du courrier. On laisse tout par défaut.

Une fois l'installation des différents paquets de configuration de Postfix terminés, nous allons modifier le fichier de configuration principal de Postfix, le fichier `main.cf` qui se trouve dans le répertoire `/etc/postfix/`.

Dans ce fichier, nous allons commencer par ajouter à la variable `relayhost` la valeur `[smtp.gmail.com]:587`, ceci permet d'indiquer à Postfix que nous allons nous servir du SMTP (Simple Mail Transfer Protocol) de Gmail, puisque nous souhaitons utiliser notre adresse de messagerie Gmail pour l'envoi des notifications de Nagios et à la fin de ce fichier, nous allons ajouter les cinq lignes suivantes :

- `smtp_sasl_auth_enable = yes`
- `smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd`
- `smtp_sasl_security_options = noanonymous`
- `smtp_tls_CAfile = /etc/postfix/cacert.pem`
- `smtp_use_tls = yes`

Ensuite nous créons le fichier `/etc/postfix/sasl_passwd` dans lequel nous allons écrire la ligne suivante :

```
[smtp.gmail.com] :587 monadressemail :monmotdepasse
```

Puis, on attribue les droits sur le fichier `sasl_passwd` via la commande :

```
chmod 400 /etc/postfix/sasl_passwd
```

Et nous demandons à Postfix de prendre en compte ce fichier avec la commande

```
postmap /etc/postfix/sasl_passwd
```

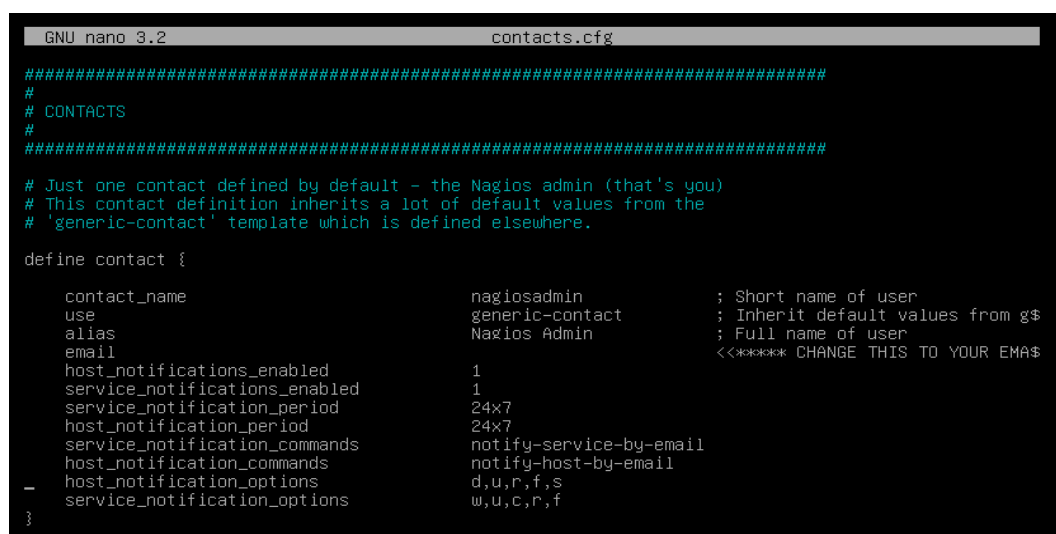
Enfin, nous allons générer un certificat d'authentification que Postfix va utiliser pour chiffrer les différents mails envoyés vers notre messagerie avec la commande :

```
cat /etc/ssl/certs/thawte_Premium_Server_CA.pem | tee -a /etc/postfix/cacert.pem
```

On redémarre le service Postfix pour que Postfix prennent en compte toutes les modifications apportées.

Maintenant que notre service Postfix est installé, on modifie les fichiers `contacts.cfg` et `commands.cfg` situés dans le répertoire `/usr/local/nagios/etc/objects` de notre serveur Nagios pour l'informer qu'à chaque nouvelle notification qui arrive au sein de notre réseau informatique, celle-ci doit être également transmise sur notre messagerie.

Au niveau du fichier `contacts.cfg`, nous allons indiquer dans la section `define contact{}`, l'adresse email de réception des alertes que notre serveur Nagios va transmettre.



```
GNU nano 3.2 contacts.cfg
#####
#
# CONTACTS
#
#####
# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the
# 'generic-contact' template which is defined elsewhere.

define contact {
    contact_name        nagiosadmin        ; Short name of user
    use                 generic-contact     ; Inherit default values from g$
    alias               Nagios Admin       ; Full name of user
    email               <<***** CHANGE THIS TO YOUR EMA$
    host_notifications_enabled    1
    service_notifications_enabled 1
    service_notification_period  24x7
    host_notification_period     24x7
    service_notification_commands notify-service-by-email
    host_notification_commands   notify-host-by-email
    host_notification_options    d,u,r,f,s
    service_notification_options w,u,c,r,f
}
```

Et au niveau du fichier `commands.cfg`, nous allons modifier les lignes `command_line` qui se trouvent dans les sections `define command {command_name notify-host-by-email}` et `define command {command_name notify-service-by-email}` en remplaçant les valeurs `bin/mail -s` par `/usr/bin/mailx -s` :

```

GNU nano 3.2                                commands.cfg

#
# SAMPLE NOTIFICATION COMMANDS
#
# These are some example notification commands.  They may or may not work on
# your system without modification.  As an example, some systems will require
# you to use "/usr/bin/mailx" instead of "/usr/bin/mail" in the commands below.
#
#####

define command {
    command_name    notify-host-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type: $NOTIFICATIONTYP$
}

define command {
    command_name    notify-service-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type: $NOTIFICATIONTYP$
}

```

On redémarre le service Nagios pour valider toutes les modifications effectuées dans les fichiers `contacts.cfg` et `commands.cfg`.