

Documentation du système de supervision NAGIOS

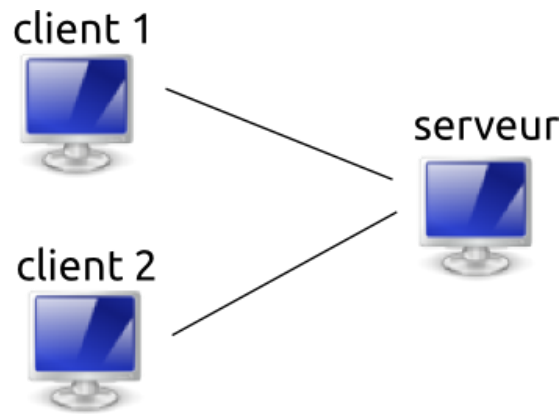
Procédure rédigée par : Vassenet-Guihot Romain

I -Présentation générale du système de supervision Nagios :	1
I.1 -Installation Nagios (Serveur Supervision) Distribution Debian 10 :	2
I.2 -Installation NRPE (Serveur Production) Debian 10 :	6
II -Rsyslog, système de gestion centralisé des logs :	8
II.1 -Installation de Rsyslog Serveur :	9
II.2 -Installation de Rsyslog Client :	10
III -Installation du système de notification E-Mail Nagios avec MailX :.....	11
IV -Nagios, Host & Services :	14
IV.1 -Configurations Nagios Système Supervision.....	14
IV.1 -Configurations Nagios Système Production	16

I - Présentation générale du système de supervision Nagios :



Nagios est une application permettant la surveillance système et réseau. Elle surveille les hôtes et services spécifiés, alertant lorsque les systèmes ont des dysfonctionnements et quand ils repassent en fonctionnement normal. Nagios est modulable de manière à installer les indicateurs nécessaires à la supervision mais il est également possible de développer ses propres indicateurs et les partager à la communauté. C'est un logiciel libre sous licence GPL.



I.1 - Installation Nagios (Serveur Supervision) Distribution Debian 10 :

Installer le serveur Apache :

Pour accéder à l'interface Web de gestion de Nagios, nous avons besoin d'un serveur Apache et de l'interpréteur PHP.

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

Installer les bibliothèques graphiques :

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

Installer les outils de compilation standards :

Enfin pour installer Nagios et ses plugins nous aurons besoin des outils de compilation standards et du package unzip :

En effet, Nagios est un programme compilé en C, donc le compilateur GCC et ses outils associés MAKE et AUTOCONF sont indispensables, et les sources sont téléchargées souvent dans un format compressé, donc UNZIP est nécessaire également.

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

Créer l'environnement Nagios :

Désormais, la **création de l'environnement Nagios**, avec son utilisateur, son groupe et son répertoire de travail. En effet, Nagios est un programme qui n'a pas besoin de tourner sous root. Pour ajouter l'utilisateur nagios sur le système, saisir la commande suivante :

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

Créer un répertoire de stockage pour tous les téléchargements :

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

Télécharger et décompresser Nagios :

```
cd /home/nagios/downloads  
wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.2.tar.gz  
tar -zxvf nagios-4.4.2.tar.gz
```

Configurer Nagios en indiquant le répertoire de destination des sites web :

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

Lancer la compilation de Nagios :

```
make all
```

Créer L'arborescence Nagios :

Afin d'y installer les fichiers binaires

```
make install
```

Installer le service et le pipe Nagios :

Installer le service Nagios, c'est-à-dire les composants nécessaires au démarrage de Nagios avec la machine.

```
make install-daemoninit  
make install-commandmode
```

Installer les fichiers de configuration de Nagios :

```
make install-config
```

Installer l'interface Web administration :

```
make install-webconf
```

Cette commande dépose le fichier nagios.conf dans l'arborescence Apache (/etc/apache2/sites-enabled). Pour fonctionner correctement, l'interface d'administration de Nagios nécessite les modules rewrite et cgi d'Apache.

```
a2enmod rewrite  
a2enmod cgi
```

Configurer l'accès Apache :

Pour accéder à l'interface d'administration de Nagios, il est nécessaire de configurer un accès Apache **htaccess**

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

Cette commande crée le fichier htaccess dans l'arborescence du site d'administration (/usr/local/nagios/etc/htpasswd.users) et configure un premier utilisateur comme ci-dessous :

- **login : nagiosadmin ;**
- **password : pass**

Configurer les droits pour la configuration :

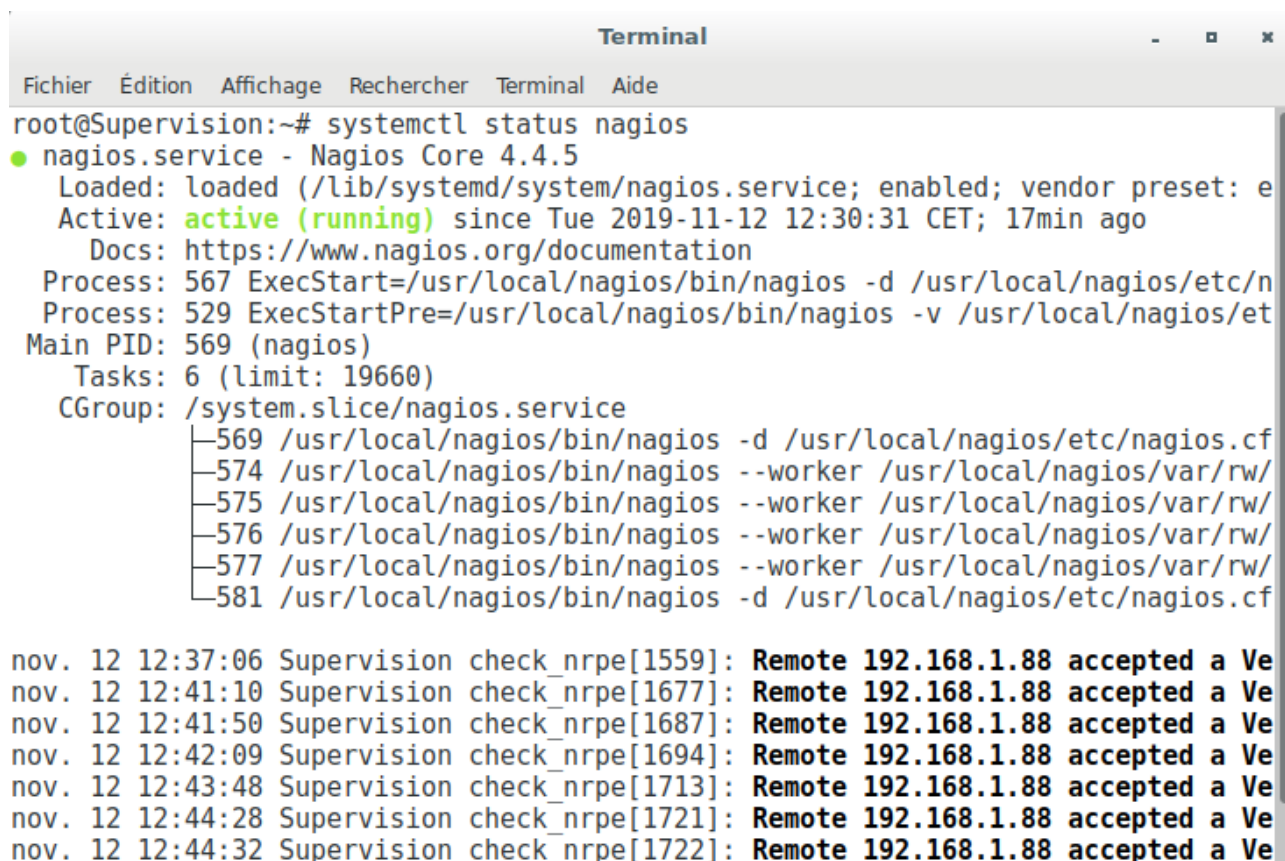
```
root@NagiosDebian:~# chown -R nagios /usr/local/nagios
```

Redémarrer Apache :

```
systemctl restart apache2  
systemctl start nagios
```

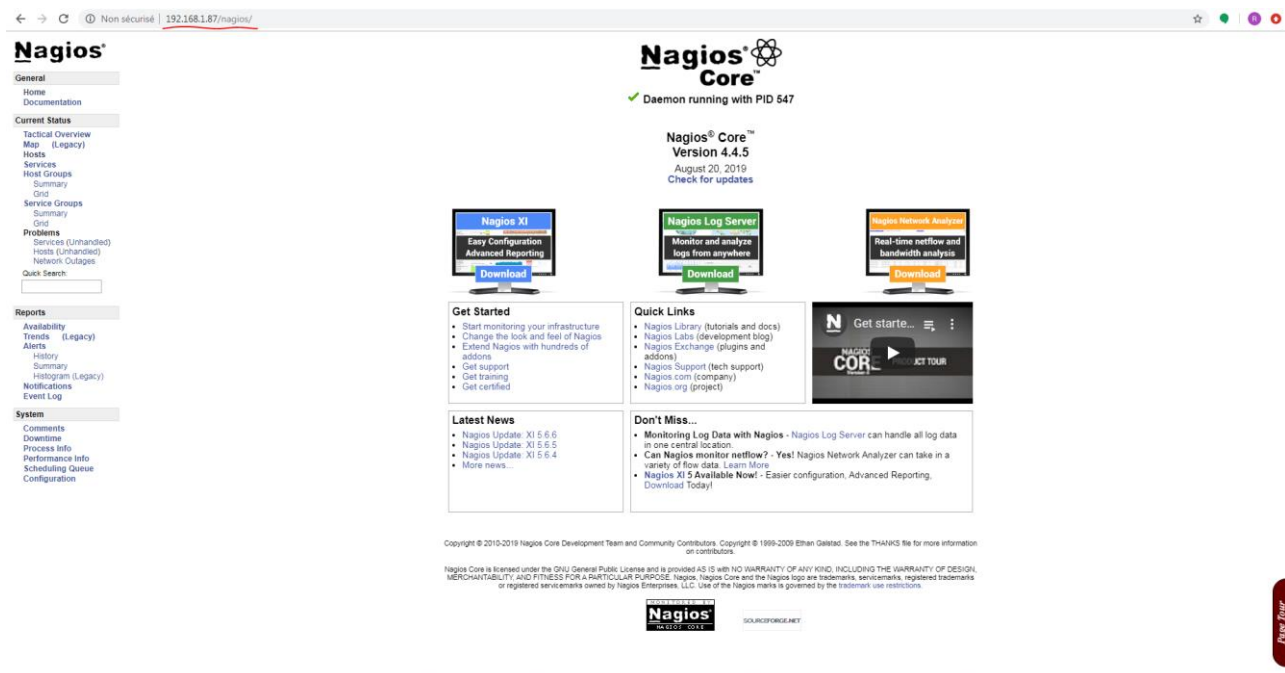
Vérifier le bon fonctionnement de Nagios :

```
systemctl status nagios
```



```
Terminal
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
root@Supervision:~# systemctl status nagios
● nagios.service - Nagios Core 4.4.5
   Loaded: loaded (/lib/systemd/system/nagios.service; enabled; vendor preset: e
   Active: active (running) since Tue 2019-11-12 12:30:31 CET; 17min ago
     Docs: https://www.nagios.org/documentation
   Process: 567 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/n
   Process: 529 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/et
   Main PID: 569 (nagios)
     Tasks: 6 (limit: 19660)
    CGroup: /system.slice/nagios.service
           └─569 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cf
           └─574 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/
           └─575 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/
           └─576 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/
           └─577 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/
           └─581 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cf

nov. 12 12:37:06 Supervision check_nrpe[1559]: Remote 192.168.1.88 accepted a Ve
nov. 12 12:41:10 Supervision check_nrpe[1677]: Remote 192.168.1.88 accepted a Ve
nov. 12 12:41:50 Supervision check_nrpe[1687]: Remote 192.168.1.88 accepted a Ve
nov. 12 12:42:09 Supervision check_nrpe[1694]: Remote 192.168.1.88 accepted a Ve
nov. 12 12:43:48 Supervision check_nrpe[1713]: Remote 192.168.1.88 accepted a Ve
nov. 12 12:44:28 Supervision check_nrpe[1721]: Remote 192.168.1.88 accepted a Ve
nov. 12 12:44:32 Supervision check_nrpe[1722]: Remote 192.168.1.88 accepted a Ve
```



I.2 - Installation NRPE (Serveur Production) Debian 10 :

NRPE (Nagios Remote PluginExecutor) est un agent de supervision qui permet de récupérer les informations à distance. Son principe de fonctionnement est simple : il suffit d'installer le serveur NRPE sur la machine distante et de l'interroger à partir du serveur Nagios.

Installer le plugin NRPE :

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

Installer le serveur NRPE :

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

Autoriser l'accès au serveur NRPE :

Pour que le serveur Nagios puisse récupérer des informations au sujet de son hôte, il faut lui donner l'accès au serveur NRPE et le définir dans le fichier `/etc/nagios/nrpe.cfg` en ajustant selon ses configurations réseaux : `server_address` et `allowed_host`

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

```
# SERVER ADDRESS
# Address that nrpe should bind to in case there are more than one interface
# and you do not want nrpe to bind on all interfaces.
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

server_address=192.168.1.88

# 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=192.168.1.0/24
```

Ajouter NRPE dans le fichier `commands.cfg` :

Dans le répertoire `/usr/local/nagios/etc/objects/`

```
GNU nano 2.7.4          Fichier : commands.cfg

define command {
    command_name    process-service-perfdata
    command_line    /usr/bin/printf "%b" "$LASTSERVICECHECK$\t$HOSTNAME$\t$SERV$
}

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

[]

^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^J Justifier ^C Pos. cur.
^X Quitter   ^R Lire fich.^_ Remplacer  ^U Coller    ^T Orthograp.^_ Aller lig.
```

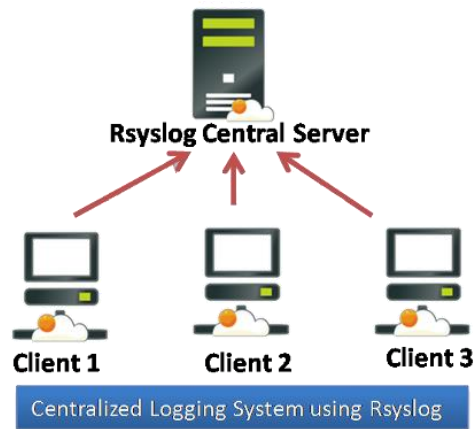
Redémarrer le service :

```
service nagios-nrpe-server restart
```

Vérifier le bon fonctionnement de NRPE (Côté Serveur Supervision) :

```
/usr/lib/nagios/plugins/check_nrpe -H 192.168.1.88 NRPE
NRPE v3.0.1
```

II - Rsyslog, système de gestion centralisé des logs :



Sous Linux, le serveur rsyslog peut être configuré pour exécuter un gestionnaire de logs centralisé, à l'aide d'un modèle service-client, et envoyer des messages de journal sur le réseau via des protocoles de transport TCP ou UDP, ou recevoir des journaux des périphériques réseau, serveurs, routeurs, commutateurs ou autres.

Le démon Rsyslog peut être configuré pour s'exécuter simultanément en tant que client et serveur. Configuré pour fonctionner en tant que serveur, Rsyslog écoute le port par défaut 514 TCP et UDP et commencera à collecter les messages de journal envoyés sur le réseau par des systèmes distants. En tant que client, Rsyslog envoie sur le réseau les messages du journal interne à un serveur Rsyslog distant via les mêmes ports TCP ou UDP.

II.1 - Installation de Rsyslog Serveur :

Installer le plugin NRPE :

```
apt-get install rsyslog
```

Définir Rsyslog en tant que serveur :

Pour configurer un programme rsyslog à exécuter en mode serveur, modifier le fichier de configuration principal dans **/etc/rsyslog.conf**. Dans ce fichier supprimer le # devant :

```
ModLoad imup
UDPServerRun 514
```

```
ModLoad imtcp
InputTCPServerRun 514
```

Définir les restrictions réseaux :

```
AllowedSender TCP, 127.0.0.1, 192.168.1.0/24
```

Définir le répertoire de stockage des logs :

```
template Incoming-logs, "/var/log/%HOSTNAME%/%PROGRAMNAME%.log"
*. * ?Incoming-logs
```

Redémarrer le service Rsyslog :

```
Systemctl restart rsyslog
```

II.2 - Installation de Rsyslog Client :

Pour permettre au démon rsyslog de s'exécuter en mode client et de générer des logs locaux vers un serveur Rsyslog distant, modifier le fichier **/etc/rsyslog.conf** et ajouter l'une des lignes suivantes :

```
*. * @@IP_reomte_syslog_server:514
```

(Dans mon cas : *. * @@192.168.1.87 :514)

*Note : @@ = TCP Protocol ; @ = UDP Protocol

Logs du serveur de production depuis le serveur de supervision :

```
192.168.1.87 - PuTTY
root@Supervision:/var/log# ls
alternatives.log      debug                lightdm              syslog.2.gz
alternatives.log.1   debug.1             mail.info            syslog.3.gz
apache2               debug.2.gz          mail.info.1          syslog.4.gz
apt                   debug.3.gz          mail.info.2.gz       syslog.5.gz
auth.log              debug.4.gz          mail.log              syslog.6.gz
auth.log.1            dpkg.log            mail.log.1           syslog.7.gz
auth.log.2.gz         dpkg.log.1          mail.log.2.gz        user.log
auth.log.3.gz         exim4               messages             user.log.1
auth.log.4.gz         faillog             messages.1           user.log.2.gz
btm                    fontconfig.log      messages.2.gz        user.log.3.gz
btm.1                 installer           messages.3.gz        user.log.4.gz
cups                  kern.log            messages.4.gz        wtmp
daemon.log            kern.log.1          Production           wtmp.1
daemon.log.1          kern.log.2.gz       speech-dispatcher    Xorg.0.log
daemon.log.2.gz       kern.log.3.gz       Supervision          Xorg.0.log.old
daemon.log.3.gz       kern.log.4.gz       syslog
daemon.log.4.gz       lastlog             syslog.1
root@Supervision:/var/log# cd Production/
root@Supervision:/var/log/Production# ls
anacron.log  cron.log    kern.log      sshd.log
auth.log     CRON.log   liblogging-stdlog.log  systemd.log
authpriv.log daemon.log local7.log     user.log
root@Supervision:/var/log/Production#
```

Pour aller plus loin nous pouvons également utiliser syslog-ng de manière à faire des sauvegardes sans saturer le HDD ou même intégrer directement les logs dans une base de donnée avec mysql.

III - Installation du système de notification E-Mail Nagios avec MailX :

```
apt-get install postfix heirloom-mailx
```

(si libsasl2-2 ca-certificates et libsasl2-modules ne sont pas installé, penser à également télécharger les paquets)

Une fois l'installation terminée, choisir via l'interface graphique postfix « site internet » et lui donner un nom (nagios.notification)

Ajouter le smtp de son service de messagerie, dans mon cas celui de gmail à l'instruction relayhost en modifiant le fichier `/etc/postfix/main.cf`

```
relayhost = [smtp.gmail.com] :587
```

Ajouter également les instructions suivantes à la fin du fichier main.cf afin d'activer les notifications de façon sécurisée :

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

```
smtp_use_tls = yes
```

Créer le fichier contenant l'adresse mail et le mot de passe dans le répertoire définit

```
nano /etc/postfix/sasl_passwd
```

```
[smtp.gmail.com]:587 example@gmail.com:motdepasse
```

Ajouter les droits/privilèges au fichier :

```
chmod 600 /etc/postfix/sasl_passwd
```

Définir les requêtes postfix avec postmap afin d'éviter les conflits d'écrasement de fichier :

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

```
chmod 600 /etc/postfix/sasl_passwd.db
```

Dupliquer le flux de donnée du certificat dans le fichier .pem avec la commande suivante :

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

Relancer le service postfix :

```
service postfix reload
```

Nous pouvons faire un test de l'envoi d'un mail avec la commande suivante :

```
Echo « Nagios Notification Test » | mail -s "Ceci est un test" example@gmail.com
```

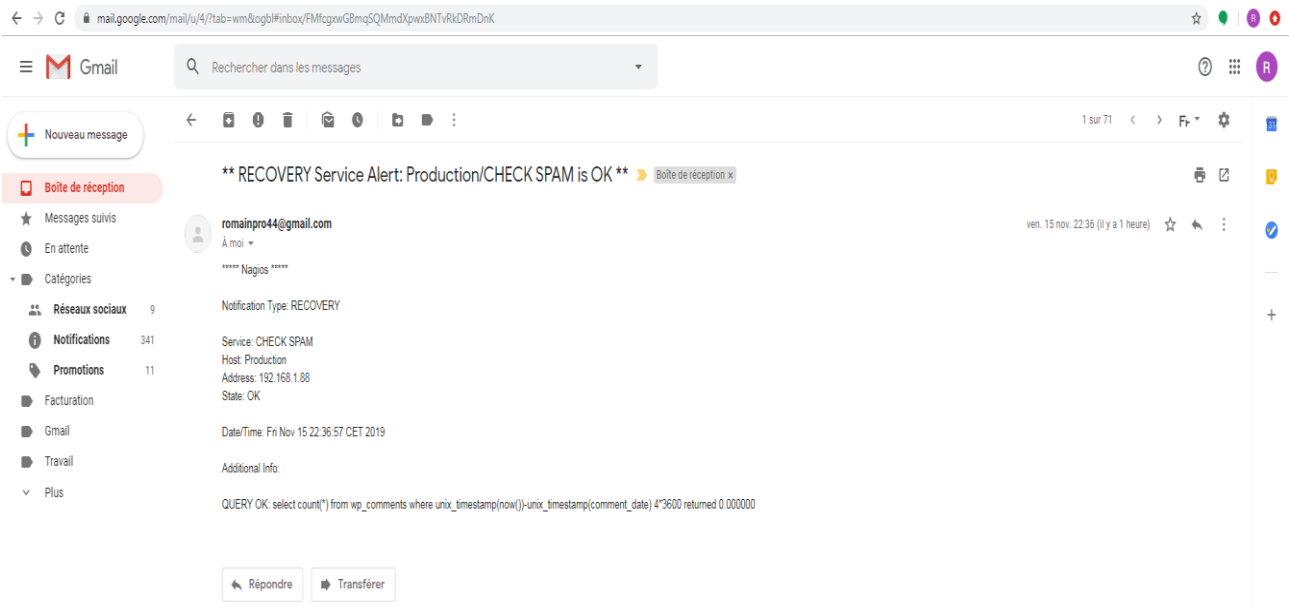
Définir le répertoire du service mailx dans nagios pour activer les notifications par mail en modifiant le fichier : /usr/local/nagios/etc/objects/commands

```
192.168.1.87 - PuTTY
#
#####
#
#####
# 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 Typ
e: $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress: $HOSTADDRE
SS$\nInfo: $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" | /usr/bin/mailx -s "*"
$NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATE$ *" $CONTACTEMAIL$
}

define command {
    command_name    notify-service-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Typ
e: $NOTIFICATIONTYPE$\nService: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HO
STADDRESS$\nState: $SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional Inf
o:\n\n$SERVICEOUTPUT$\n" | /usr/bin/mailx -s "*" $NOTIFICATIONTYPE$ Service Aler
t: $HOSTALIAS/$SERVICEDESC$ is $SERVICESTATE$ *" $CONTACTEMAIL$
}
}
```

Enfin ajouter l'adresse mail dans le fichier contacts de Nagios dans **/usr/local/nagios/etc/objects/contacts.cfg**

```
192.168.1.87 - PuTTY
#
# 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 generic-contact template (defined above)
    alias           Nagios Admin                ; Full name of user
    email           romainpro44@gmail.com    ; <<***** CHANGE THIS TO
YOUR EMAIL ADDRESS *****
}
}
```



IV - Nagios, Host & Services :

IV.1 - Configurations Nagios Système Supervision

Nous pouvons modifier les instructions de supervision du serveur dans ce répertoire :
`/usr/local/nagios/etc`

commands.cfg définit les commandes générales des différents instruments de supervision possible avec la possibilité d'ajout d'argument du type `-w $ARG1$ -c $ARG2$` (Argument warning & Argument Critique)

Ce fichier contiendra la liste de l'ensemble des outils de supervision possible, mais il restera à définir lesquels nous souhaitons utiliser par la suite pour notre host (localhost)

Exemple d'une commande pour l'instrument de supervision du Disque définit dans **commands.cfg** :

```
define command {  
  
    command_name    check_local_disk  
  
    command_line    $USER1$/check_disk -w $ARG1$ -c $ARG2$
```

```
}
```

La force de Nagios étant de donner la possibilité à ces différents instruments de supervision une ou plusieurs valeurs de seuil à ne pas dépasser pour estimer la qualité du ou des services supervisés.

localhost.cfg

Ce fichier nous servira ensuite à définir des informations général sur notre serveur de supervision (Nom host, adresse IP, etc ..) ainsi que l'ensemble des services voulant être supervisés :

Dans l'exemple ci-dessous, nous définissons pour la commande `check_local_procs` enregistré dans le fichier `commands.cfg` les valeurs des arguments ARG1 & ARG2 à 250 & 400.

```
define service {  
  
    use          local-service      ; Name of service template $  
    host_name    localhost  
    service_description  TOTAL PROCS  
    check_command    check_local_procs!250!400!RSZDT  
}
```

En conclusion, si la valeur du nombre total de processus est > 250 alors une alerte Attention s'affichera sur le dashboard Web de Nagios ainsi qu'une notification mail si le système est correctement configuré. De même si le nombre de processus est > 400 pour une alerte critique

Définir un nouvel Host (Nouveau client à superviser depuis notre Serveur Nagios, par exemple un serveur de production)

Créer le document `hosts` dans `/usr/local/nagios/etc/`

Créer le fichier `Production.cfg` dans `/usr/local/nagios/etc/hosts` (ou le nom de l'host distant à superviser)

Puis définir les instructions de notre fichier :

```
nano /usr/local/nagios/etc/host/Production.cfg
```

```
define host{  
use linux-server  
host_name Production  
alias Production  
address 192.168.1.88  
}
```

(Chaque nouveaux clients s'ajouteront en commençant par : define host {informations host})

The screenshot shows the Nagios web interface. At the top, there are summary statistics for Host Status Totals and Service Status Totals. Below that, there is a table titled 'Host Status Details For All Host Groups' with columns for Host, Status, Last Check, Duration, and Status Information. The table shows two hosts: 'Production' and 'localhost', both with a status of 'UP'.

Host	Status	Last Check	Duration	Status Information
Production	UP	11-16-2019 17:59:06	0d 1h 36m 57s	PING OK - Paquets perdus = 0%, RTA = 0.38 ms
localhost	UP	11-16-2019 17:59:58	28d 21h 15m 19s	PING OK - Paquets perdus = 0%, RTA = 0.03 ms

IV.1 - Configurations Nagios Système Production

Définir les services à superviser pour notre client distant Production :

Créer le document services dans `/usr/local/nagios/etc/`

Créer le fichier Production.cfg dans `/usr/local/nagios/etc/services` (ou le nom de l'host distant à superviser)

Définir les instructions : (Veiller à ce que les instruments de supervision soit bien installés sur le serveur de production afin de retourner la valeur au serveur de supervision grâce à NRPE.

```
define service{  
use generic-service  
host_name Production  
service_description CHECK BANDWIDTH
```



```
check_command check_nrpe!check_eth  
}
```

The screenshot shows the Nagios web interface. At the top, there are summary statistics for Host Status Totals and Service Status Totals. Below this, there are sections for 'Current Network Status', 'Host Status Totals', and 'Service Status Totals'. The main part of the interface is a table titled 'Service Status Details For All Hosts'. This table lists various services for two hosts: 'Production' and 'localhost'. Each row includes the service name, its status (OK, Warning, Unknown, Critical, Pending), the last check time, duration, and attempts. The 'Status Information' column provides details for each service, such as disk space usage, memory usage, and network connectivity.

Important ! : Tous les instruments de supervision téléchargés à partir de nagios ou de github sont à installer dans le répertoire /usr/local/nagios/libexec (Dans le cas du serveur de supervision)

```
192.168.1.87 - PuTTY  
root@Supervision:/usr/local/nagios/libexec# ls  
attachment.php?link_id=6873  check_icmp          check_nt             check_swap  
check_apt                    check_ide_smart     check_ntp           check_tcp  
check_breeze                 check_ifoperstatus check_ntp_peer     check_time  
check_by_ssh                 check_ifstatus      check_ntp_time     check_udp  
check_clamd                  check_imap          check_nwstat       check_ups  
check_cluster                check_ircd          check_oracle        check_uptime  
check_dhcp                   check_load          check_overcr        check_users  
check_disk                   check_log           check_ping          check_wave  
check_disk_smb               check_mailq         check_pop            negate  
check_dummy                  check_mem.pl        check_procs         urlize  
check_eth                    check_mrtg          check_real           utils.pm  
check_file_age               check_mrtgtraf     check_rpc            utils.sh  
check_flexlm                 check_nagios        check_sensors  
check_ftp                    check_nntp          check_sntp  
check_http                   check_nrpe          check_ssh
```

Important ! : Tous les instruments de supervision téléchargés à partir de nagios ou de github sont à installer dans le répertoire /usr/local/nagios/libexec (Dans le cas du serveur de supervision)

cas du serveur de supervision, /usr/lib/nagios/plugins dans le cas du serveur de production)

```
root@Production:/usr/lib/nagios/plugins# ls
attachment.php?link_id=2516  check_ifstatus  check_ping
check_apt                    check_imap      check_pop
check_breeze                 check_ircd      check_procs
check_by_ssh                  check_jabber    check_radius
check_clamd                   check_ldap      check_real
check_cluster                 check_ldaps     check_rpc
check_dbi                     check_load      check_rta_multi
check_dhcp                    check_log        check_sensors
check_dig                     check_mailq     check_simap
check_disk                    check_mem.pl    check_smtp
check_disk_smb                check_mrtg      check_snmp
check_dns                      check_mrtgtraf  check_spop
check_dummy                   check_mysql     check_ssh
check_eth                      check_mysql_query  check_ssmt
check_file_age                check_nagios    check_swap
check_flexlm                  check_nntp      check_tcp
check_fping                   check_nntp      check_time
check_ftp                      check_nt        check_udp
check_game                     check_ntp       check_ups
check_host                     check_ntp_peer  check_users
check_hpjd                     check_ntp_time  check_wave
check_http                     check_nwstat    negate
check_icmp                     check_oracle    urlize
check_ide_smart                check_overcr    utils.pm
check_ifoperstatus            check_pgsql     utils.sh
```

Avant d'inscrire en « dur » nos commandes à superviser, il est conseillé de tester dans un premier temps la validité de l'instruction.

Par exemple pour tester la commande `check_eth` sur l'interface `ens32` avec deux arguments

```
/usr/local/nagios/libexec/check_eth -i ens32 -w 1024K Bps -c 2048K Bps
```

Il est également possible de customiser et créer nos propres instruments de supervision :

Dans l'exemple ci-dessous nous allons comptabiliser le nombre d'enregistrements de la table `wp_comments` à partir de la connexion à la base de donnée (nous donnons les arguments `-d` pour la connexion à la base de donnée `mysql` `-u` pour le login `-p` pour le mot passe)

La fonction `unix_timestamp(now())` nous permet de définir le temps actuel – `unix_timestamp(comment_date)` qui nous permet de définir de temps de la table.

En passant la condition $<4*3600$ et les arguments attention à 4 et critique à 10.

Ainsi, si le site web reçoit 4 commentaires ou plus dans les 4 dernières heures, une alerte se déclenchera dans le système de supervision Nagios.

```
command[check_spam]=/usr/lib/nagios/plugins/check_mysql_query -q "select count(*) from wp_comments where unix_timestamp(now()-unix_timestamp(comment_date)) <4*3600" -w 4 -c 10 -H 127.0.0.1 -P 3306 -d wordpress_db -u wpuser -p wppassword
```

Voici une liste de sondes afin d'étudier le fonctionnement des différents services du système :

CHECK BANDWIDTH : Sonde pour étudier la bande passante en fonction de l'interface réseau choisit (ifdown ens32 /ifup ens32 pour rétablir l'interface réseau)

CHECK DISK : Sonde pour étudier l'utilisation du disque. Bloquer le programme responsable d'une utilisation excessive si alerte

CHECK FTP : Sonde pour étudier l'utilisation du service FTP : Vérifier le port 21 ouvert si problème et relancer le service : netstat -laputen | grep LISTEN (systemctl restart vsftpd)

CHECK HTTP : Sonde pour étudier le service http : Vérifier le port 80 ouvert si problème et relancer le service : netstat -laputen | grep LISTEN (service httpd stop puis service httpd start)

CHECK LOAD : Sonde pour étudier l'état général (cpu / ram) : Relancer la machine si une alerte est défini. Si le problème persiste diagnostiquer si c'est un problème de mémoire ou de cpu et analyser les logs machines.

CHECK SPAM : Sonde pour étudier l'état général du nombre de commentaires du site wordpress : si problème persiste, ajouter un plugin ban_IP et ajouter une vérification afin ajout d'un commentaire.

CHECK SSH : Sonde pour vérifier le protocole SSH. Si problème vérifier le port 22 et relancer le service : systemctl stop ssh puis systemctl start ssh

CHECK USERS : Sonde pour vérifier le nombre d'utilisateurs sur la machine : Limiter le nombre d'utilisateur total de la machine si problème et supprimer les faux

PING HOST : Sonde pour vérifier si la machine est bien sur le réseau. Vérifier si la machine est bien sur le réseaux, DHCP ? ifconfig ? nano /etc/network/interfaces (pour configurer le manuel)

TOTAL PROCS : Sonde pour vérifier le nombre de processus actifs (Supprimer le programme responsable d'un trop grand nombre de processus si problème actif)

ZOMBIE PROCS : Sonde pour vérifier le nombre de processus « zombie » d'un programme n'étant plus actif.

CHECK SWAP : Sonde pour vérifier l'allocation de mémoire « cache » si besoin : Réinstaller le programme en question si leak de mémoire ou upgrade la ram du système.

Dans la majorité des sondes il est possible de vérifier l'état du service avec un `systemctl status « nom du service »` parfois `systemctl « nom du service » status`