Université d'Angers
UFR Sciences
Promotion 2008/2009

Projet Tuteuré

Etude de faisabilité de réalisation d'un serveur de supervision de type FAN

LAMBERT Jonathan
BELLANGER Laurent
Vuilmet Nicolas
MOREAU Cyril

Sommaire

Présentation du projet :

Le but de notre projet est, dans un premier temps, d'évaluer et de tester la distribution FAN (Fully Automated Nagios). La seconde partie consistait à vérifier si l'intégration de SALSA dans FAN était possible. SALSA est un ensemble de modules utilisé au département informatique qui sert à alerter les administrateurs en cas de problème sur un service surveillé (exemple: température).

Ce projet a été axé principalement sur la surveillance de température. Une sonde de température placée dans une salle renvoie la température ambiante d'un local technique pour alerter les administrateurs de la température. En cas de température trop élevée ou trop basse, une alerte est envoyée aux administrateurs par téléphone via le module SALSA.

De plus, ce projet inclue la génération de graphiques pour évaluer les modifications d'état des services à surveiller (exemple: la température).

La supervision :

La fonction de supervision consiste à indiquer et à commander l'état d'un appel, d'un système ou d'un réseau. Elle inclut la surveillance, la visualisation, l'analyse et le pilotage.

La supervision informatique permet de surveiller l'ensemble du système d'information d'une entreprise :

* Le réseau et ses équipements(routeur etc...)
* Les serveurs
* Les périphériques
* Les applications

Il est important de garantir la disponibilité du système en cas de panne, mais aussi de prévenir en cas de problème et, le cas échéant, garantir une remontée d'information rapide pour une durée d'intervention minimale.

Il existe plusieurs méthodes pour superviser un système d'information :

* Analyser les fichiers de log
* Récupérer des résultats de commandes et de scripts locaux ou distants
* Le protocole SNMP : Simple Network Management Protocole.

Lambert Jonathan Bellanger Laurent Vuilmet Nicolas Moreau Cyril

Présentation de FAN :

FAN est une solution simple et rapide pour avoir un système de supervision. Le but du projet FAN est de proposer une image iso prête à l'emploi du logiciel de supervision Nagios entouré de divers outils et addons.

FAN s'appuie sur la distribution CentOS à laquelle, lors de l'installation, Nagios et ses outils de configuration, de cartographie et de reporting y ont été ajoutés.

Installation :

FAN utilise le programme d'installation de CentOS.

Une fois l'installation effectuer tous est prêt a l'emploi. Il ne reste plu qu'a ouvrir un navigateur et d'y entrer l'adresse du serveur pour avoir accès a la page d'accueil de FAN qui regroupe l'accès a tous les logiciels.

Configuration :

Pour tester un service sur une machine il faut dans un premier temps ajouter la machine (l'host) dans Nagios. Puis dans un second temps il faut ajouter le service a tester en le liant avec l'host en question.

Par défaut les identifiants pour tous les logiciels sont les suivants :

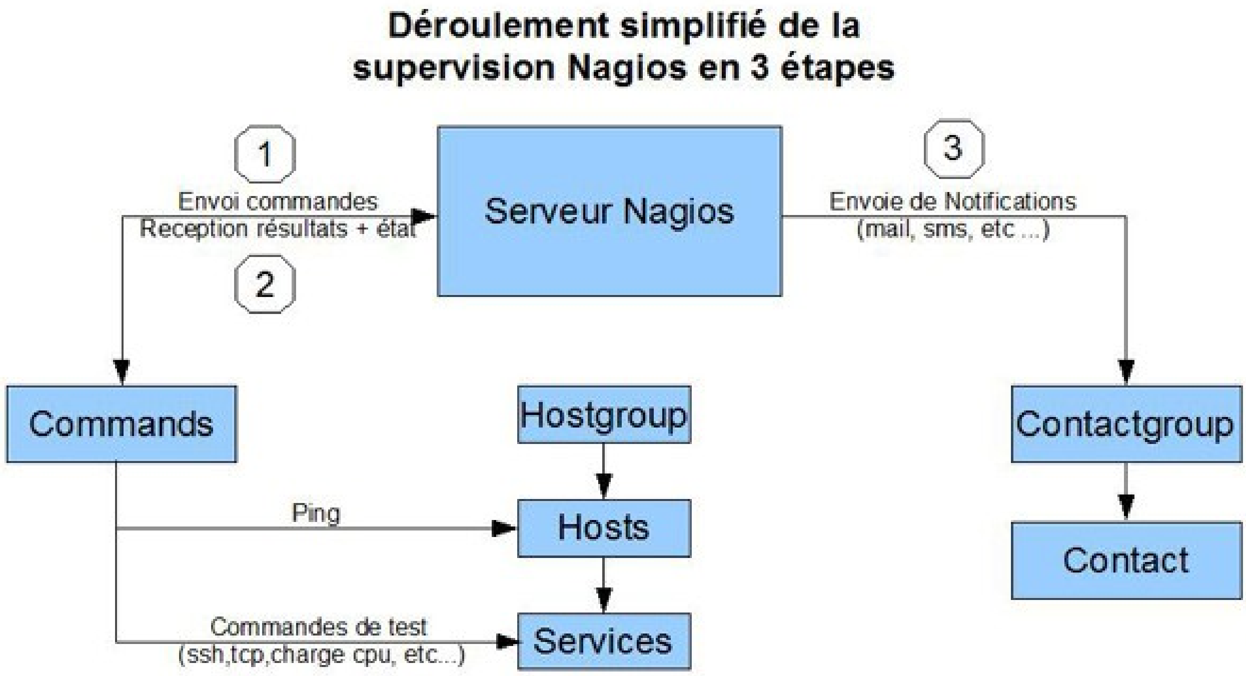
*Login : nagiosadmin

*Password : nagiosadmin

Vous pouvez changer la langue pour metre le logiciel en francais dans Options > My Account puis changer la langue principale en Fr. (Ne pas oublier de confirmer le mot de passe avant de valider)

Nagios :

Les avantages de Nagios sont très nombreux. Ce projet bénéficie d'une communauté très active qui fait évoluer le logiciel en permanence et ajoute de nombreux services. Beaucoup de Plugin ont étaient développer pour tester divers services. Plusieurs logiciels, utilisant la récupération d'informations de Nagios, sont nés.



Fichiers de configuration Nagios (/etc/nagios/):

La description de chaque fichier contient une liste non exhaustive des paramètres de configuration. Ceux présentés ici sont ceux qui sont le plus utilisés.

nagios.cfg :
Fichier principal de configuration du serveur nagios.

hosts.cfg :
Configuration des machines (adresse ip, commande de test d'état, contactgroup à prévenir en cas de modification d'état, état appelant une notification, période de notification, interval de notification).

hostextinfo.cfg :
Configuration des informations externes des machines (affichage sur l'interface web pour l'utilisateur : lien vers un graph d'état par exemple).

hostgroup.cfg :
Configuration des groupes de machine (nom du groupe et machines membres du groupe).

contacts.cfg :
Configuration des contacts à prévenir en cas de changement d'état d'une machine ou d'un service (nom, email, pager, période de notification du contact pour les machines, période de notification du contact pour les services, états à notifier pour les machines, états à notifier pour les services, commandes de notification à utiliser par le serveur).

contactgroup.cfg :
Configuration des groupes de contacts (nom du groupe et contacts membres du groupe).

services.cfg :
Configuration des services (tests) à effectuer par le serveur (nom du service, machine auquel le service est rataché, test passif ou actif : supervision partagée ou non, nom de la commande de check à utiliser).

serviceextinfo.cfg :
Configuration des informations externes des services (affichage sur l'interface web pour l'utilisateur : lien vers un graph de valeurs par exemple).

commands.cfg :
Configuration des commandes utilisées par Nagios, commandes de check, commandes d'envoi de mail etc ... Cette configuration contient le chemin vers le fichier exécutable de la commande afin de faire le lien entre le chemin et le nom utilisé dans la configuration.

ressources.cfg :
Configuration des ressources de Nagios, c'est à dire des chemin « important » utilisés par nagios, comme le chemin vers le dossier racine du dossier contenant les commandes. Ce fichier sert essentiellement à fixer des variables d'environnement pour Nagios et simplifier ainsi les différentes configurations de fichiers (notement nagios.cfg).

timeperiods.cfg :
Configuration des périodes de notification de nagios (horaires de notification de chaque journée de la semaine).

Note : Pour tous ces fichiers (excepté nagios.cfg), il est possible de créer des templates (modèles) afin de simplifier la configuration. Des valeurs par défauts seront donc fixées et ne seront prises en compte que si la configuration de l'host utilisant le template ne précise pas de valeur pour ces informations (exemple : notification_period). Ainsi on prévient bon nombre de problèmes en affectant une configuration correcte et fonctionnelle pour chaque configuration.

Centreon :

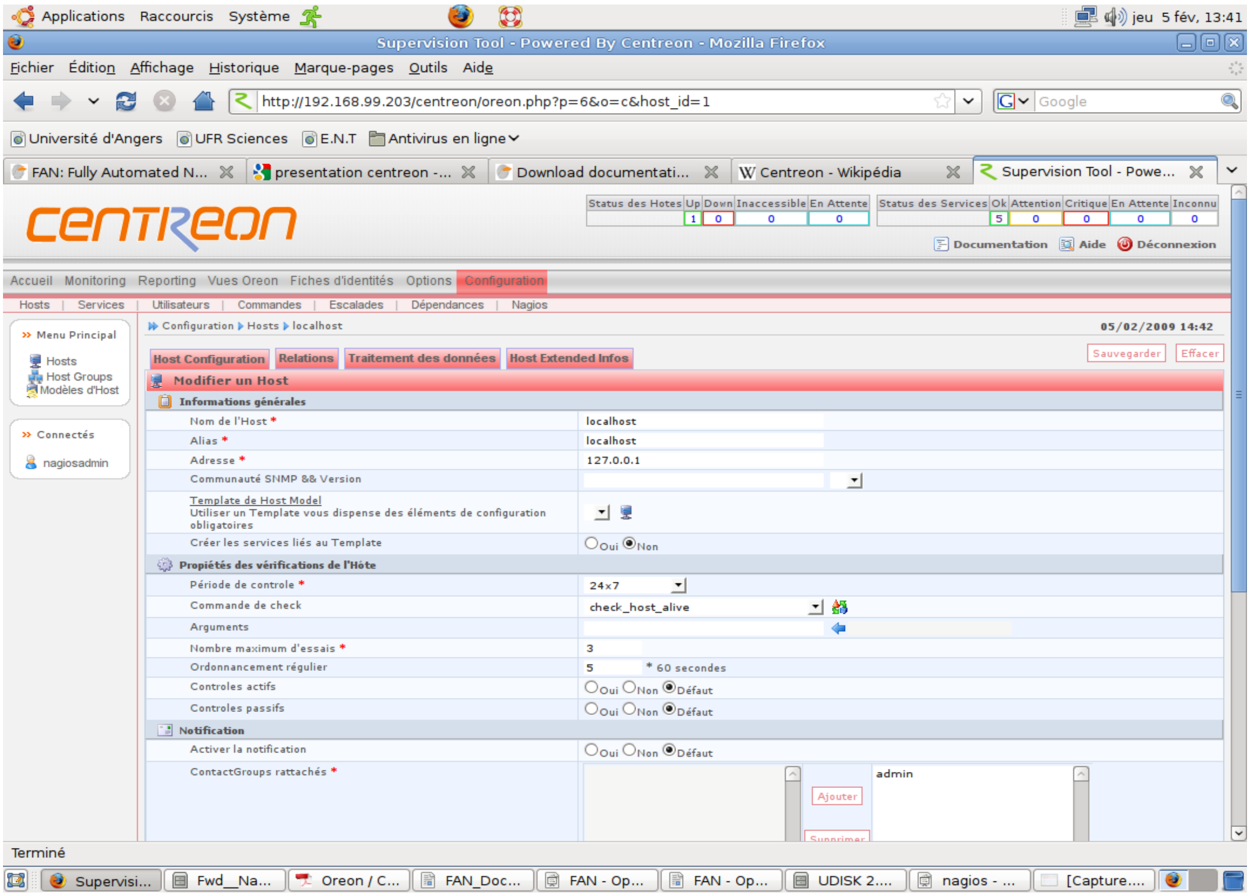
Nagios comporte déjà une interface web informant l'utilisateur sur l'état des indicateurs, mais elle ne permet pas d'agir sur les fichiers de configuration de Nagios. Centreon est une interface web permettant, tout comme le fait déjà Nagios, la consultation des indicateurs, mais aussi le paramétrage de Nagios par interface graphique. Il génère à la demande de nouveaux fichiers de configuration.

Exemples d'ajout de machines et services :

Ajouter une machine :

Nous allons voir comment ajouter un host dans Nagios.

Dans un premier temps ouvrez un navigateur et aller sur l'interface Centreon de votre serveur, puis aller dans le menu Configuration. Vous êtes automatiquement arriver dans la configuration des hosts. Pour ajouter un host, cliquer sur ajouter en haut de la liste.



Il faut ensuite renseigner les différents champs, ceux avec une étoile rouge sont obligatoire :

Informations générale:

Nom de l'host : Entrer le nom de la machine.

Alias : Nom qu'affichera Nagios (souvent le même que Nom de l'host).

Adresse : Adresse de l'host (en IP ou son nom sur le domaine).

Propriété de vérification de l'host :

Période de contrôle : 24x7 (En permanence).

Commande de check : choisir la commande qui servira a savoir si l'host est présent.

Nombre maximum d'essai : 3.

Propriété de notification:

ContactGroup rattaché : Ajouter admin.

intervalle de notification : 5 (pour toute les 5 minutes).

Periode : 24x7 (En permanence).

Type : cocher Down et Unrechable pour être prévenu a chaque fois que l'host est éteint ou injoignable.

Vous pouvez maintenant cliquer sur enregistrer. Votre host est ajouter a la liste des host a tester. Mais ce n'est pas encore pris en compte par Nagios, avant tout ajoutons lui un service a tester.

Ajouter un service :

Maintenant que l'host est enregistrer dans Nagios nous pouvons ajouter les tests à effectuer.

Comme pour ajouter une machine, avec un navigateur aller sur l'interface Centreon de votre serveur, puis aller dans le menu Configuration puis le sous menu Services. En haut de la liste cliquer sur ajouter.

Remplir les différent champs :

Informations générales :

Description : Nom du service que vous tester.

Statut du Service :

Période : 24x7 (en permanence).

Commande de check : choisir la commande approprier au service par exemple check_http pour tester le serveur http de la machine.

Nombre maximum d'essai : 3 (essai avant notification d'un problème).

Ordonnancement régulier et non régulier : 1 (pour toute les minutes).

Notification :

ContactGroup rattaché : ajouter admin (C'est le groupe admin qui sera prévenu).

Interval de notification : 5 pour toutes les 5 minutes.

Periode de notification : 24x7 (en permanence).

Type de notification : Warning, Unknow et Critical.

Puis aller dans l'onglé Relations. Vous pouvez alors ajouter les hosts qui subirons le test.

Une fois fait clique sur Sauvegarder.

Vous avez maintenant lier un service a tester a un host, mais pour que vos changement soit activé il faut regénérer les fichiers de configuration Nagios.

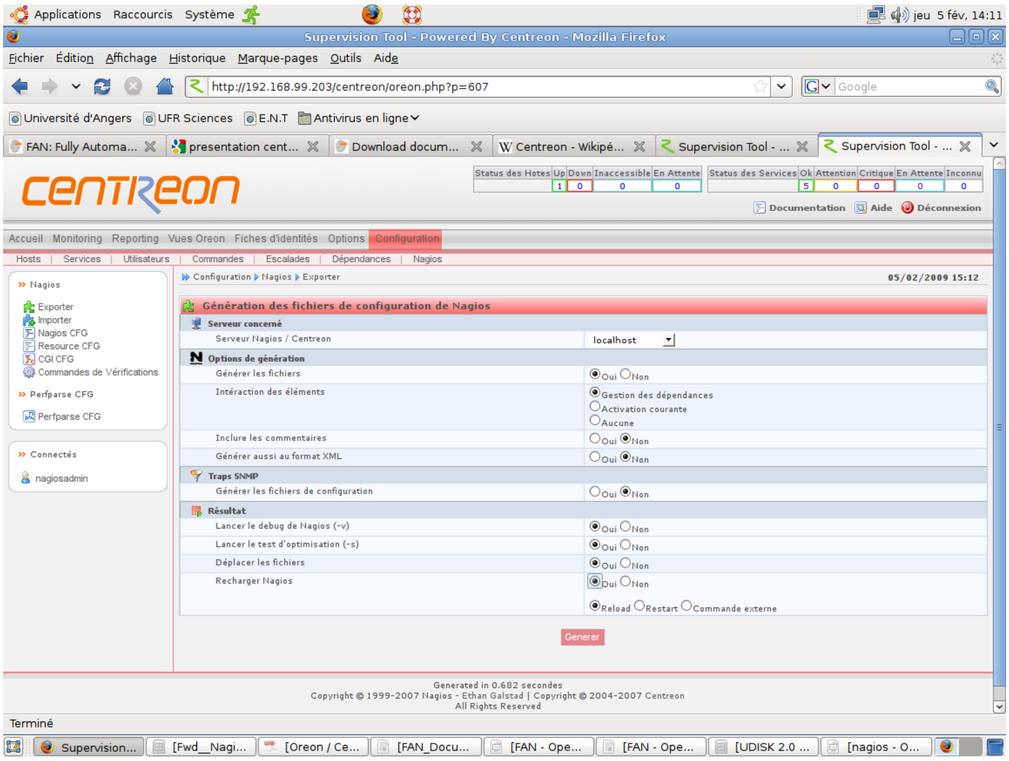
Générer les nouveaux fichiers de configuration:

Toujours dans l'interface de Centreon, cliquer sur Configuration puis aller dans le sous menu Nagios.

Ici vous pouvez sauvegarder et appliquer les changement effectuer dans le menu Configuration de Centreon.

Pour cela, dans le sous menu Resultat du formulaire cocher toutes les options a oui et cliquer sur Générer.

Un log vous indique que les fichiers ont tous été générés et que Nagios les a pris en compte. Pour vérifier aller dans le menu Monitoring et cliquer sur tous les services.



Lambert Jonathan Bellanger Laurent Vuilmet Nicolas Moreau Cyril

HowTo plug-in / 1-wire :

Module d'interfaçage utilisé :

- Digitemp v3.0.6. for Linux (http://www.digitemp.com).

Installation :

➔ Télécharger les sources archivées et compressés de Digitemp (digitemp-3.6.0.tar.gz) sur le site du développeur (http://www.digitemp.com/software.shtml).

➔ Placer le fichier téléchargé dans le répertoire */usr/src/.*

➔ Décompresser et extraire le contenu de l'archive :

```
cd /usr/src
tar -zxvf digitemp-3.6.0.tar.gz
```

➔ Compiler les sources du projet en spécifiant l'utilisation de la liaison série (module ds9097):

```
make ds9097
```

Note :
Il est impératif de récupérer, puis d'installer les paquetages make et gcc (non disponibles par défaut dans la distribution FAN) au préalable, sans quoi la compilation et l'installation de Digitemp sera impossible.

```
yum install make
yum install gcc
```

➔ Déplacer le fichier binaire résultant de la compilation dans le répertoire /usr/local/bin : `mv digitemp_DS9097 /usr/local/bin`

➔ Créer un lien symbolique vers le fichier binaire résultant de la compilation (l'appel à l'exécutable digitemp_DS9097 se fera via la commande digitemp) :

```
ln -s /usr/local/bin/digitemp_DS9097 /usr/local/bin/digitemp
```

Configuration :

➔ Spécifier le fichier de configuration et l'interface auxquels doit se référer Digitemp :

```
digitemp -i -c /usr/local/etc/digitemp.conf -s /dev/ttyS0
```

(pour liaison série seule)
ou

```
digitemp -i -c /usr/local/etc/digitemp.conf -s /dev/ttyUSB0
```

(pour un capteur liaison série relié à l'ordinateur par l'intermédiaire d'un adaptateur Série/USB).

➔ Vérifier la lecture de la température de la sonde :

```
digitemp -c /usr/local/etc/digitemp.conf -a
```

➔ Créer le script bash (get_temperature) chargé de récupérer la température mesuré par le capteur dans /usr/local/bin/ :

```
#!/bin/bash
# get_temperature: polls the temperature sensor array and leaves
# the temperatures in a state file. Called by cron every 5
minutes.

TMPFILE=/tmp/temperature.XXXXXX
STATEFILE=/var/lib/temperature/current
DIGITEMP=/usr/local/bin/digitemp
DIGICONF=/usr/local/etc/digitemp.conf

# Abort after first script error.
set -e

# Get a unique temporary tamper-proof file name.
tmp=$(mktemp $TMPFILE)

# Create a full poll list of the temperature array. This takes up
to
# 5 seconds per sensor, and therefore must be done to a (slowly
growing)
# temporary file.
$DIGITEMP -c $DIGICONF -a -q > $tmp

# 'Atomically' move the freshly created state file in place.
mv $tmp $STATEFILE
chmod 755 $STATEFILE
```

➔ Créer dans /etc/cron.d/ le script (appelé get_temperature également) permettant de lancer le script précédant de façon périodique (5 minutes dans cet exemple) :

```
# Poll the temperature sensor array every five minutes.
*/5 * * * * root /usr/local/bin/get_temperature

/* Optional*/
# Append the last poll to the history file after each whole hour.
4 * * * * root cat /var/lib/temperature/current >>
/var/log/temperature.log
```

➔ Créer le script que Nagios utilisera pour interagir avec le fichier cron et suivez les instructions de l'en-tête :

```
#!/usr/bin/perl -w
# check_temperature: Nagios wrapper script around digitemp.
```

```
# Used to monitor a couple of 1-wire temperature sensors and to
raise an
# alarm when one of them reports a temperature outside a
predefined band.
# For more information: http://www.hoppie.nl/tempsens .

# Permission is hereby granted, free of charge, to any person
obtaining a copy
# of this software and associated documentation files (the
"Software"), to
# deal in the Software without restriction, including without
limitation the
# rights to use, copy, modify, merge, publish, distribute,
sublicense, and/or
# sell copies of the Software, and to permit persons to whom the
Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO
EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS
# IN THE SOFTWARE.
#
# Howto Install in Nagios (tested with v1.3)
#
# 0. Make sure that Digitemp works and temperatures are written to
the file:
#    /var/lib/temperature/current
#    as described at http://www.hoppie.nl/tempsens/ .
#
# 1. Copy this Perl script to /usr/local/nagios/libexec/ or
wherever you have
#    placed your Nagios plugins.
#
# 2. Add a command to /usr/local/nagios/etc/checkcommands.cfg like
this:
```

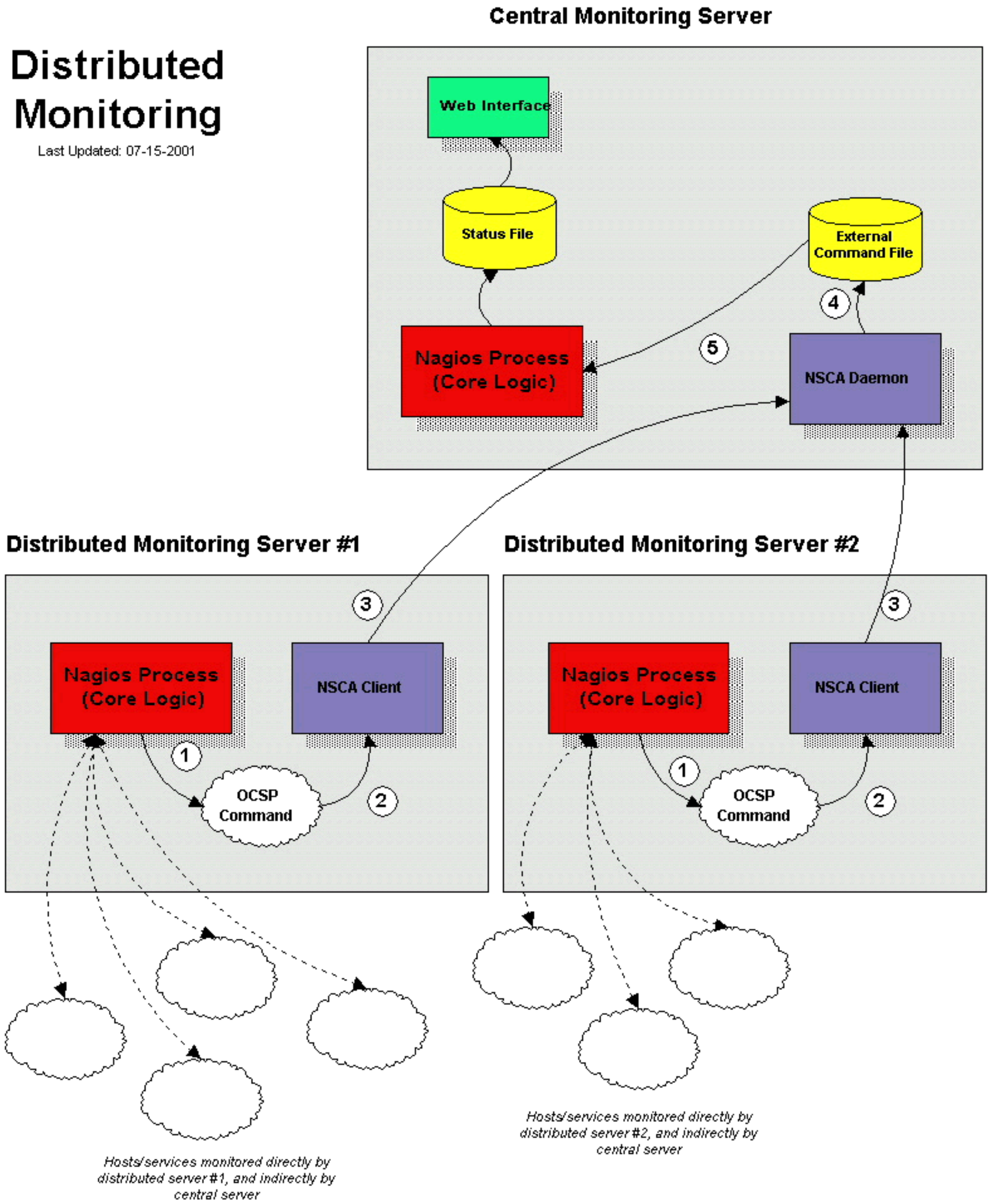
```
#
#    # DigiTemp temperature check command
#    define command {
#      command_name check_temperature
#      command_line /usr/local/nagios/libexec/check_temperature \
#                   -s $ARG1$ -t $ARG2$ -w $ARG3$ -c $ARG4$
#      (fold the above into one line)
#    }
#
# 3. Tell Nagios to monitor the temperature by adding a service
line like
#    this to your service.cfg file:
#
#    # DigiTemp Temperature check Service definition
#    define service {
#        use                             generic-service
#        host_name                       kermit
#        service_description             Temperature
#        is_volatile                     0
#        check_period                    24x7
#        max_check_attempts              3
#        normal_check_interval           5
#        retry_check_interval            2
#        contact_groups                  home-admins
#        notification_interval           240
#        notification_period             24x7
#        notification_options            w,u,c,r
#        check_command                   check_temperature!1!25!3!
5
#    }
#
#    In this example,
#      1 is the sensor # (as reported by digitemp -a) to monitor
#     25 is the target (ideal) temperature
#      3 is the warning temperature deviation from the target
#      5 is the critical temperature deviation from the target

#
===================================================================
=========

# Modules to use.
use strict;
use Getopt::Std;

# Define all our variables.
use vars qw($temperature_state
            $opt_s $opt_t $opt_w $opt_c
            $sensor $target $warn_dev $crit_dev
```

```
            $temperature
            %exit_codes);

# Place to look for the temperature state file (may be customised
here).
$temperature_state = '/var/lib/temperature/current';

# Predefined exit codes for Nagios.
%exit_codes   = ('UNKNOWN' ,-1,
                 'OK'      , 0,
                 'WARNING' , 1,
                 'CRITICAL', 2,);

# Get the options
if ($#ARGV le 0)
{
  &usage;
} else {
  getopts('s:t:w:c:');
}

# Shortcircuit the switches
if (!$opt_w or $opt_w == 0 or !$opt_c or $opt_c == 0)
{
  print "*** You must define WARNING and CRITICAL deviation
bands!\n";
  &usage;
}

# Check if levels are sane
if ($opt_w >= $opt_c)
{
  print "*** WARNING deviation band must not be greater than
CRITICAL!\n";
  &usage;
}

# Default sensor to read is #0
if(!$opt_s)
{
  $sensor = 0;
} else {
  $sensor = $opt_s;
}

# Default target temperature is 20 (degrees Centigrade).
if(!$opt_t)
{
  $target = 20;
```

```
} else {
  $target = $opt_t;
}

$warn_dev = $opt_w;
$crit_dev = $opt_c;

# Read the output from digitemp, as plain temperature Centigrade.
# In order to avoid race conditions, a separate cron job usually
placed in
# /usr/local/bin/get_temperature polls digitemp and leaves the
most recent
# poll list in /var/lib/temperature/current . We just need to
fetch the
# correct line out of this file to get the digitemp output.

# Does the file exist and is it readable?
if (!-r $temperature_state) {
  print "Could not read file $temperature_state; aborting.\n";
  exit $exit_codes{'UNKNOWN'};
}

open( DIGITEMP, $temperature_state );
$temperature = -9999;
while( <DIGITEMP> )
{
  chomp;
  # Select only the correct line.
  if( $_ =~ /Sensor $sensor/i )
  {
    # Extract the temperature in Centigrade. Allow more than 10
sensors and
    # negative temperatures.
    /Sensor [0-9]+ C: ([0-9.-]+)/;
    $temperature = $1;
    last;
  }
}
close( DIGITEMP );

# Was the requested sensor temperature available in the state
file?
if( $temperature==-9999 )
{
  # No!
  print "No data found for sensor #$sensor\n";
  exit $exit_codes{'UNKNOWN'};
}
```

```
# Now for the real checks.
if( $temperature and abs($target-$temperature) >= $crit_dev )
{
  print "Temperature CRITICAL - Sensor #$sensor = $temperature
C\n";
  exit $exit_codes{'CRITICAL'};
} elsif ($temperature and abs($target-$temperature) >= $warn_dev )
{
  print "Temperature WARNING - Sensor #$sensor = $temperature
C\n";
  exit $exit_codes{'WARNING'};
} elsif( $temperature ) {
  print "Temperature OK - Sensor #$sensor = $temperature C\n";
  exit $exit_codes{'OK'};
} else {
  print "Error parsing result for sensor #$sensor\n";
  exit $exit_codes{'UNKNOWN'};
}

# Show usage
sub usage()
{
  print "check_temperature v1.1 - Nagios Plugin\n";
  print "Copyright 2006 Jeroen Hoppenbrouwers <hoppie\@hoppie.nl>\
n";
  print "More info: http://www.hoppie.nl/tempsens/\n";
  print "See source for License and Nagios config example.\n\n";
  print "Usage:\n";
  print " check_temperature -s <sensor> -t <target> -w <warn> -c
<crit>\n\n";
  print "Options:\n";
  print " -s n              DigiTemp Sensor #, default 0\n";
  print " -t temperature    Target temperature in Centigrade,
default 20\n";
  print " -w deviation      Temperature deviation from target to
warn (required)\n";
  print " -c deviation      Temperature deviation from target when
critical (required)\n\n";
  print "Output:\n";
  print " UNKNOWN/-1, OK/0, WARNING/1, CRITICAL/2\n";

  exit $exit_codes{'UNKNOWN'};
}
```

La supervision répartie :



Le but de la supervision répartie de Nagios est d'alléger le serveur central en utilisant plusieurs serveurs (un central et des esclaves).

Le serveur central est celui qui est consulté par les administrateurs, il regroupe les informations de toutes les machines testées.

Les serveurs esclaves font des tests sur les machines qui leur ont été attribuées (comme un serveur normal). La seule différence est que le serveur esclave renvoie les résultats de tous les tests au serveur central qui écoute le réseau grâce au démon NSCA.

Concurrence :

Cacti :

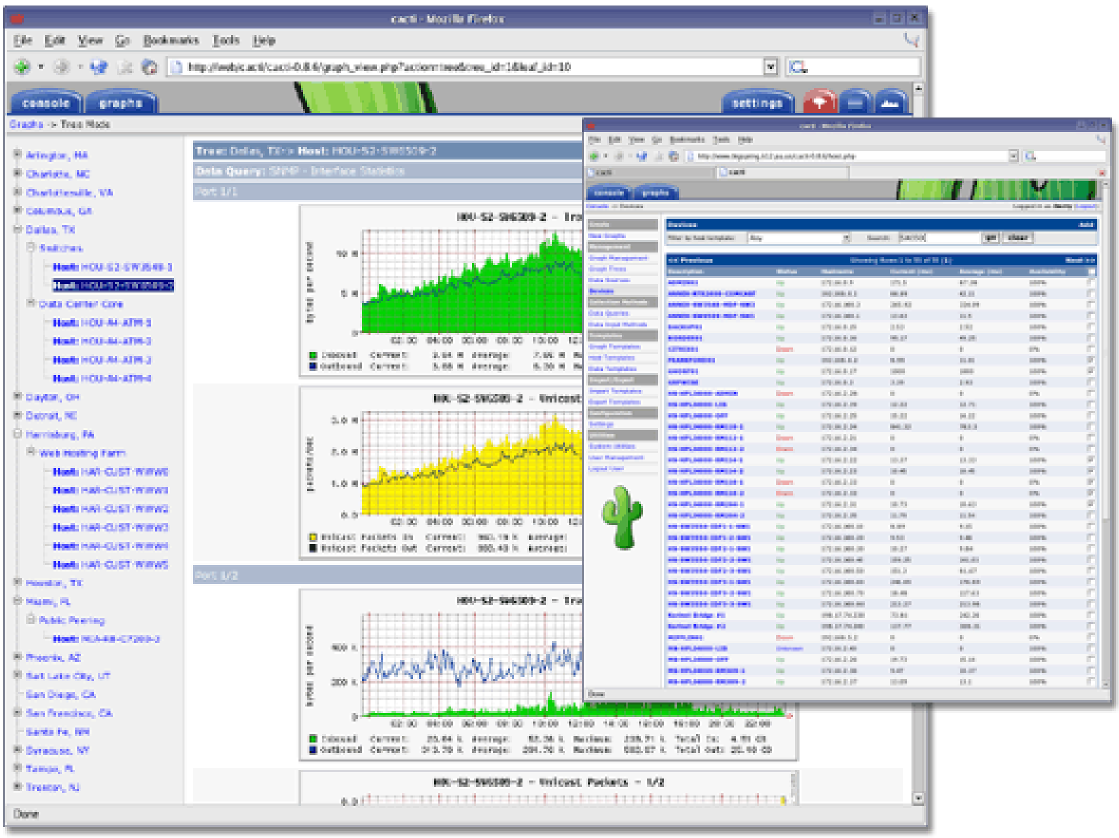
Cacti est un logiciel de supervision réseau . Il fonctionne grâce à un serveur web équipé d'une base de données et du langage PHP.

Il permet de représenter graphiquement divers statuts de périphériques réseau utilisant SNMP ou encore grâce à des scripts (Bash, PHP, Perl, VBs...) pour avoir par exemple l'espace disque restant ou bien la mémoire utilisée, la charge processeur ou le ping d'un élément actif. Les données sont récoltées auprès des différents agents SNMP (ou auprès des scripts locaux) grâce à un script php. Pour de meilleures performances un exécutable, nommé cactid, peut également effectuer les interrogations.

L'intérêt de ce logiciel réside principalement dans son principe de « modèles » (Templates) qui permet de créer de manière générique les graphiques afin de pouvoir les réutiliser. De manière générale, « tout » est modèle sous Cacti. Cela est avantageux lorsque de nombreuses données identiques doivent être observées, mais cela peut se révéler fastidieux à configurer lorsque les données sont hétérogènes.

Il est également possible d'effectuer des opérations simples (et des combinaisons d'opérations) avec les différentes données grâce une interface graphique

Cette solution n'est pas destinée à alerter en temps réel sur les dysfonctionnements d'un système mais bien de proposer une vision dans le temps de l'évolution d'indicateurs matériels et logiciels (trafic réseau, occupation des disques, temps de réponse, etc.).



Points forts :

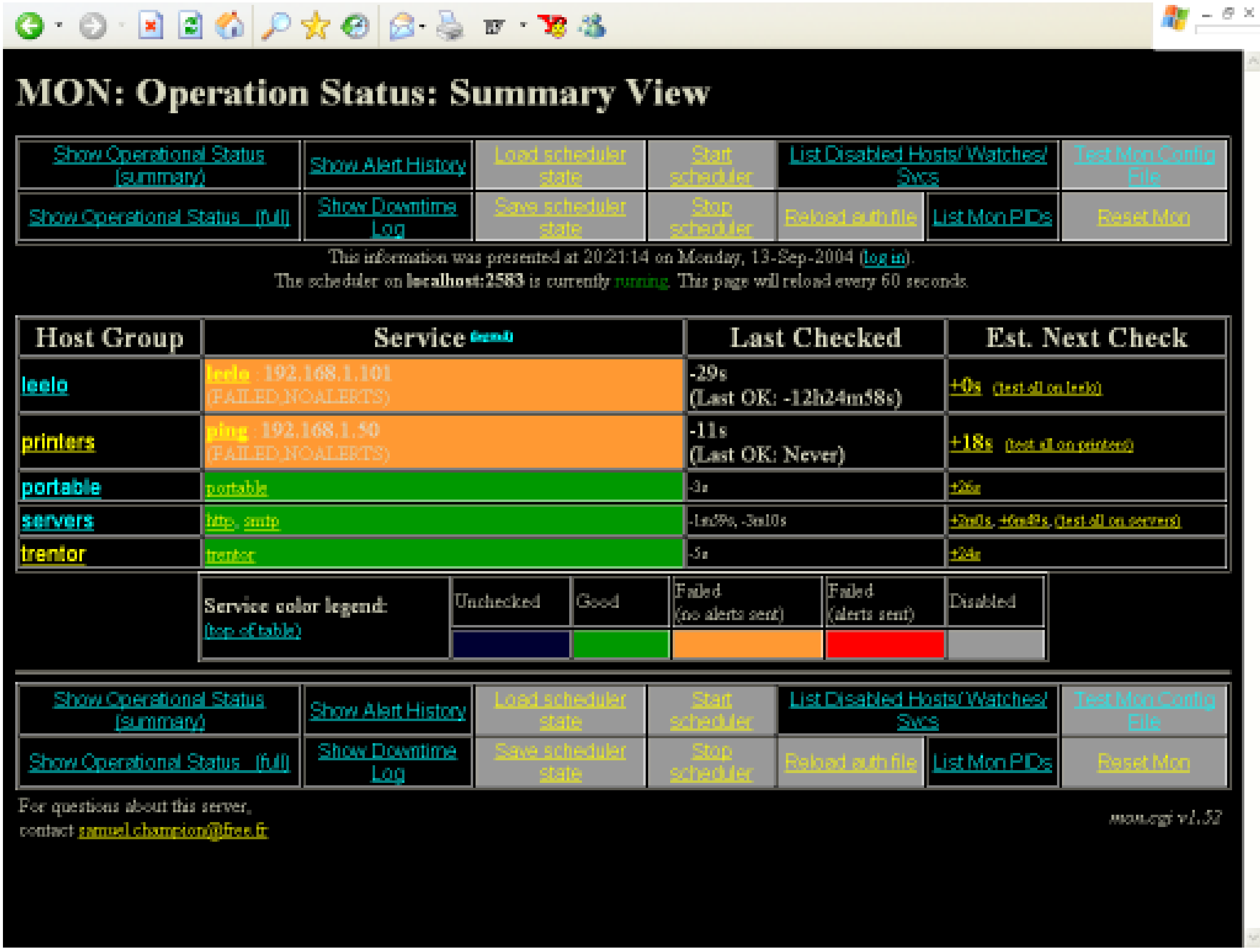
- Notion de template (tout est « template »);
- Graphiques complètement paramétrables (couleur, type d'affichage, valeurs affichées);
- Renseignement automatique (partitions, interfaces réseaux, ...);
- Très forte communauté;
- Nombreux plugins (pour ajouter des fonctionnalités, plugin Nagios pour Cacti);
- Facile à configurer.

Points faibles :

- Ne fais pas du temp reel;
- Utilisé pour voir les varations de temperature (etc...) dans un certain delai;
- Les alertes ne sont pas aussi bien developpé que Nagios.

MON :

MON est un outil de supervision qui vérifie les services et envoi des alertes lors d'événements prévus. Sa configuration s'effectue à l'aide d'un seul fichier qui utilise un langage facile à prendre en main. C'est un logiciel qui garde de grandes possibilités du fait de sa simplicité. Le projet qui s'occupe de ce programme n'est plus actif depuis juin 2007.



Points forts :

- Facilité d'installation et de configuration;
- Toutes les actions sont des scripts (Tests, Notifications, ...).

Points faibles :

- Communauté inexistante;
- Aucun Plugins;
- Interface web minimaliste.

Ces deux « utilitaires » ont des points forts, mais aussi des points faibles, Nagios quant à lui rassemble la majorité de ces points forts sans pour autant cumuler les points faibles. Dans cette optique nous appuyons la demande d'utilisation de Nagios à l'UFR de Sciences.

Les problèmes rencontrés et les solutions apportées :

Les problèmes rencontrés :

Nous avons rencontré des problèmes au niveau de l'organisation du travail, et de la répartition des tâches. En effet nous ne nous rendions pas vraiment compte de l'ampleur du travail à effectuer. Ainsi la répartition des tâches n'a pas été simple à mettre en place.

Les solutions trouvées :

La répartition des tâches a été repensée, et modifiée en prenant en compte tous les aspects du problème, c'est à dire en fonction des compétences de chacun et de la difficulté de chaque jalons du projet.

Ce que le projet et ces problèmes nous ont apportés :

La mise en place et l'organisation du projet nous ont permis de mettre un pied dans le monde du travail. En effet, la réalisation d'un projet nous a obligé à répartir les tâches de façon intelligente.

Sources :

- http://fr.wikipedia.org/
- http://mon.wiki.kernel.org/
- http://nagiosexchange.org/
- http://nagios.org
- http://fannagioscd.sourceforge.net/
- http://centreon.com
- http://hoppie.nl/tempsens/

Annexe 1 (Nagios.cfg) :

Paramètres du fichier nagios.cfg classés par ordre alphabétique :

accept_passive_service_checks	Accepte/Refuse les contrôles passifs au redémarrage de Nagios.
admin_email	C'est l'adresse mail de l'administrateur local de la machine (i.e. celle sur laquelle Nagios tourne). Cette valeur peut être utilisée dans les commandes de notification grâce à la macro $ADMINEMAIL$.
admin_pager	C'est le numéro du pager (ou la passerelle pager-email) de l'administrateur de la machine locale (i.e. celle sur laquelle Nagios tourne). Le numéro ou l'adresse de pager peut être utilisé dans les commandes de notification grâce à la macro $ADMINPAGER$.
aggregate_status_updates	Aggrégation des données d'état. Cette fonction est désactivé par défaut à cause de la charge CPU qu'elle entraîne.
cfg_file	Chemin d'un fichier de configuration des objets Nagios. Il est possible d'ajouter plusieurs fois ce paramètre pour charger tous les fichiers nécessaires.
cfg_dir	Chemin d'un répertoire de configuration des objets Nagios. Il est possible d'ajouter plusieurs fois ce paramètre pour charger tous les fichiers nécessaires. Exemple : il peut être préférable de créer un fichier par équipement à surveiller. Dans ce fichier seront réunis la définition d'hôte et les définitions de services associés à cet hôte.
check_external_commands	Active/désactive la vérification des commandes externes. Il faut l'activer pour passer des commandes à travers l'interface Web ou à l'aide de scripts externes.
check_for_orphaned_services	Cette option vous permet d'activer ou désactiver la vérification des contrôles de service orphelins. Les contrôles de service orphelins sont des contrôles ayant été exécutés et supprimés de la file des évènements, mais dont les résultats n'ont pas été remontés depuis longtemps.
check_service_freshness	Cette option détemine si Nagios va controler ou non périodiquement la validité des données d'un

	service. L'activation de cette option aide à controler que les controles de services passifs sont reçus en temps et heure.
command_check_interval	Interval de temps avant que Nagios re-vérifie les commandes externes en attentes. • Nombre positif : L'interval sera en unité de temps. Voir "interval_length" • Nombre positif suivi d'un "s" : L'interval sera en secondes • "-1" : Avec cette valeur, Nagios vérifiera les commandes en attentes aussi souvent que possible
command_file	Fichier de stockage des commandes en attentes. Il faut voir ce fichier comme un spooler d'impression, chaque commande envoyé par des programmes externes ou la couche web, est ajouté et attend d'être exécutée.
comment_file	Chemin du fichier de stockage, des commentaires entrés dans la couche Web pour des hôtes et services.
date_format	Cette option spécifie le format de date que Nagios utilisera dans l'interface web • us : MM/DD/YYYY HH:MM:SS • euro : DD/MM/YYYY HH:MM:SS • iso8601 : YYYY-MM-DD HH:MM:SS • strict-iso8601 : YYYY-MM-DDTHH:MM:SS
downtime_file	Chemin du fichier de stockage du planning d'indisponibilité des hôtes et services.
enable_event_handlers	Active/désactive les gestionnaires d'évènements au redémarrage de Nagios.
enable_flap_detection	Fonction expérimental de detection des hôtes et services ayant des réactions aléatoires lors des contrôles.
enable_notifications	Active/désactive l'envoi d'une notification au redémarrage de Nagios.
event_handler_timeout	C'est le nombre maximal de secondes pendant lequel Nagios laissera tourner un gestionnaire d'événement. Si un gestionnaire d'événement dépasse cette limite il sera tué et une alerte sera journalisée.
execute_service_checks	Active/désactive la vérification de service au redémarrage de Nagios.

freshness_check_interval	Cette option détermine l'intervalle de temps entre deux controles de validité des données d'un service. Si vous avez désactivé ce service, avec l'option (de controle de validité des données d'un service), cette option n'a pas d'effet.
global_host_event_handler	Nom court d'une commande (gestionnaire d'évènement) qui sera exécuté à chaque changement d'état pour les hôtes.
global_service_event_handler	Nom court d'une commande (gestionnaire d'évènement) qui sera exécuté à chaque changement d'état pour les services.
high_host_flap_threshold	Seuil supérieur pour la détection de l'oscillation d'un hôte
high_service_flap_threshold	Seuil supérieur pour la détection de l'oscillation d'un service.
host_check_timeout	C'est le nombre maximal de secondes pendant lequel Nagios laissera tourner un contrôle d'hôte. Si le contrôle dépasse cette limite, il est tué et un état CRITICAL est retourné, et l'hôte sera supposé être dans l'état DOWN. Une erreur de dépassement de délai est également journalisée
illegal_macro_output_chars	Cette option vous permet de spécifier les caractères illégaux qui seront filtrés dans les macros, avant qu'elles soient utilisées dans les notifications, les gestionnaires d'évènements et autres commandes. Ceci n'affecte pas les macros utilisées dans les controles des services ou des hotes. Vous pouvez choisir de ne pas filtrer les caractères donnés en exemple ci-dessus, mais je ne vous recommande pas de le faire. Quelques uns d'entre eux sont interprètés par le shell (par exemple, le `) et peuvent poser des problèmes de sécurité.
illegal_object_name_chars	Cette option vous permet de spécifier quels sont les caractères illégaux dans les noms d'objets, tels que hotes, services et autres. Nagios vous autorisera la plupart des caractères dans les définitions d'objets, mais je recommande de ne pas utiliser les caractères ci-dessus. Le faire est s'exposer à des problèmes dans l'interface web, les notifications de commandes, etc.
inter_check_delay_method	Selection d'une méthode de planification pour l'exécution du contrôle des services. • n = Ne pas utiliser de délai [no delay] - ordonnancer le lancement de toutes les

	contrôles maintenant (i.e. en même temps !) • d = Utiliser un délai "irréfléchi" [dumb] d'1 seconde entre les contrôles de service • s = Utiliser un calcul de délai "débrouillard" [smart] pour répartir également les contrôles de service (par défaut) • x.xx = Utiliser le délai fourni de x.xx secondes Attention, le paramètre "no delay" va déclencher des pics d'utilisation du processeur
interval_length	Définition en secondes, de ce que sera une unité de temps. Les unités de temps sont souvent utilisées dans les paramètres de définitions des hôtes et services.
lock_file	Fichier contenant le numéro de processus en cours quand Nagios est lancé en tant que Daemon (-d)
log_archive_path	Chemin d'un répertoire qui sera utilisé pour stocker les logs après rotation.
log_event_handlers	Active/désactive la journalisation pour lors de l'exécution des gestionnaires d'évènements.
log_external_commands	Active/désactive la journalisation des commandes externes.
log_file	Chemin du journal de logs, principal de Nagios. **Attention : ce paramètre doit être le premier dans le fichier nagios.cfg afin de journaliser les erreurs rencontrées.**
log_host_retries	Active/désactive la journalisation des tentatives de re-vérification des hôtes.
log_initial_states	Active/désactive la journalisation de l'état initial des hôtes et services.
log_notifications	Active/désactive la journalisation des notifications.
log_passive_service_checks	Active/désactive la journalisation des vérifications passives de services.
log_rotation_method	Paramètre pour définir la fréquence de rotation des logs. • n = Aucune [None] • h = Toutes les heures [Hourly] • d = Tous les jours [Daily] • w = Toutes les semaines [Weekly] • m = Tous les mois [Monthly]

log_service_retries	Active/désactive la journalisation des tentatives de re-vérification des services.
low_host_flap_threshold	Seuil inférieur pour la détection de l'oscillation d'un hôte
low_service_flap_threshold	Seuil inférieur pour la détection de l'oscillation d'un service.
max_concurrent_checks	Nombre de contrôles maximum qui peuvent s'exécuter simultanément. Mettre "1" pour désactiver la parallèlisation des contrôles et "0" pour n'imposer aucune limite. A ajuster en fonction de la capacité de la machine sur laquelle est installé Nagios.
nagios_group	Compte de groupe Unix sous lequel tournera le processus Nagios.
nagios_user	Compte d'utilisateur Unix sous lequel tournera le processus Nagios.
notification_timeout	C'est le nombre maximal de secondes pendant lequel Nagios laissera tourner une commande de notification. Si une commande de notification dépasse cette limite elle sera tuée et une alerte sera journalisée
obsess_over_services	Cette variable détermine si Nagios remontera les résultats de contrôles de service et lancera la commande de remontée de contrôle de service que vous avez défini. Si vous ne faites pas de supervision répartie, n'activez pas cette option
ocsp_command	Cette option définit la commande à lancer après chaque contrôle de service,ce aui peut etre utile dans une supervision répartie. Elles est exécutée après les éventuelles commandes de gestion d'événement ou de notification. L'argument commande est le nom court d'une définition de commande que vous avez définie dans le fichier de configuration des hôtes. Cette option sert dans le cadre de la supervision répartie. Le temps d'exécution maximal de cette commande est déterminé par la variable ocsp_timeout.
ocsp_timeout	C'est le nombre maximal de secondes pendant lequel Nagios laissera tourner une commande de remontée de contrôle de service. Si une commande dépasse cette limite, elle sera tuée et une alerte sera journalisée.
perfdata_timeout	C'est le nombre maximal de secondes pendant lequel Nagios laissera tourner une commande de traitement des données liées aux performance d'un hôte ou de traitement des données liées aux

	performances d'un service. Si une commande dépasse cette limite, elle sera tuée et une alerte sera journalisée.
process_performance_data	Cette valeur détermine si Nagios traitera les données liées aux performances des contrôles d'hôtes et de services.
resource_file	Chemin d'un fichier de ressource contenant entre autre le chemin des plugins et des macros utilisées dans les commandes.
retain_state_information	Active/désactive la conservation de l'état des hôtes et services entre deux démarrage de Nagios.
retention_update_interval	Fréquence en minutes à laquelle Nagios sauvegardera les données de mémorisation en situation normale.
service_check_timeout	C'est le nombre maximal de secondes pendant lequel Nagios laissera tourner un contrôle de service.Si le contrôle dépasse cette limite, il est tué et un état CRITICAL est retourné. Une erreur de dépassement de délai est également journalisée.
service_interleave_factor	Facteur permettant de modérer l'exécution des contrôles de services.
service_reaper_frequency	Fréquence en secondes des événements de "consolidation" des services.
sleep_time	Temps en secondes pendant lequel Nagios va rester en sommeil avant de relancer les séquences de vérifications des services, hôtes et l'exécution de commandes en attentes.
soft_state_dependencies	Active/désactive l'utilisation des états "soft" lors du contrôle des dépendances de services.
state_retention_file	Chemin du fichier de stockage des données d'états. Utilisé quand le paramètre "retain_state_information" est activé.
status_file	Chemin du fichier utilisé par Nagios pour stocker l'état actuel des hôtes et services.
status_update_interval	Fréquence en secondes à laquelle Nagios mettra à jour les données d'états. Ce paramètre est sans effet quand aggregate_status_updates est désactivé.
temp_file	Chemin d'un fichier temporaire généré et utilisé par Nagios pour stocker certaines valeurs.
use_agressive_host_checking	Paramètre qui était surtout valable pour les anciennes versions de nagios. Laissez "0" = désactivé
use_retained_program_state	Active/désactive le chargement de variables à partir du fichier de mémorisation.

use_syslog	Active/désactive l'utilisation de syslog. Il ne s'agit que des logs standards de Nagios, les autres journaux comme celui des états ne sont pas concernés par cette option.