Nagios

Pythagore F.D.
CT/030425/1/061129

Supervision : définitions

Supervision

Définiton de « supervision réseau »

La supervision réseau (network monitoring) se traduit par l'utilisation d'un système qui vérifie constament l'état d'un réseau d'ordinateurs interconnectés et qui notifie l'administrateur réseau concerné en cas de problème, via email ou autre moyen. La supervision est donc une sous-partie de la gestion de réseau.

Exemple pratique

Pour déterminer l'état d'un serveur web, un système de supervision pourra envoyer périodiquement une requète HTTP pour vérifier l'accès à une page web. Pour un serveur mail, le système peut envoyer une requète SMTP d'envoi de courriel à une boîte appropriée et récupérer ce même courriel via les protocoles IMAP ou POP3.

Gestion des évennements

Les tests donnant lieu à des erreurs (connexion perdue, message d'erreur du service...) conduisent le système à réagir avec une action appropriée prédéfinie par l'administrateur. Ce peut être une notification ou encore une activation d'un service de secours (failover).

Extensions courantes

La mesure du traffic réseau ou des performances de resources distribuées est une extension qui n'est pas toujours incluse dans les systèmes de supervision. Dans ce cas, elle sera a déléguée à des outils spécifiques et adaptés.

Services réseau

Qu'est-ce qu'un service réseau

On attribue le terme « service réseau » à toute fonction de haut niveau accessible par un protocole de communication réseau abstrait et qui est maintenue par une machine appelée serveur. En général, un service réseau concerne la couche applicative du modèle réseau ARPA.

L'internet

Il existe une multitude de services réseau dédiés à l'internet. Les plus populaires sont sans doute le service HTTP, utilisé pour les échanges de documents (le « web ») et le service SMTP, utilisé pour l'échange de courrier électronique (l'« e-mail »).

Normalisations

Le site de l'IETF rassemble les documents qui proposent et définissent des protocoles réseaux standards.
Voir le site : http://www.ietf.org/rfc.html.

Supervision

Objectifs

Connaître à distance et de manière automatique l'état des objets et ressources nécessaires au bon fonctionnement du système. Etre notifié d'un problème ou d'une panne.

Méthode

On peut surveiller avec différents niveaux de granularité. Exemple : « présence d'une machine sur le réseau » ou encore « taux de remplissage d'un disque en particuler ».

Le première tâche consiste à définir les objets à superviser, par exemple :

- services (exemple : http, smtp...)
- ports de transport (exemple : TCP 80, TCP 25)
- périphériques (exemple : espace disque occupé, état de l'imprimante...)
- unité centrale (exemple : charge du processeur, température de l'UC...)

Pour chaque objet surveillé, on définira la manière d'obtenir l'information et les seuils qui doivent déclencher une alerte.

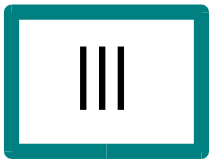
Techniques

Il existe différentes techniques de prélèvement d'informations sur les ressources à surveiller. On retiendra principalement trois méthodes valables pour la majorité des sytèmes informatiques :

- prélèvement par snmp (simple network management protocol). SNMP est un protocole standard de gestion des ressources à distance. Il est décrit dans une série de RFC dont les premiers (historiques) sont les 1155 à 1157 pour la version 1. SNMP utilise le transport UDP sur les ports 161-162.

- commandes de vérification. Certaines commandes usuelles nous renseignent sur l'état des ressources locales (ex: df, du, who...) et peuvent être exécutées à distance via telnet ou ssh. On peut utiliser l'ordonnanceur du système *cron* pour les lancer périodiquement puis analyser leur résultat. De plus, *cron* envoie systématiquement un email d'alerte si le code de retour des commandes est non nul.

- outils spécifiques de supervision. Des outils de supervision ergonomiques peuvent utiliser des méthodes de surveillance ad-hoc. Ils génèrent un historique de l'état des ressources et peuvent proposer un service d'alerte élaboré.



Présentation Nagios

Généralités

Historique

1999	NetSaint version 0.0.1
2002	NetSaint version 0.0.7 devient Nagios 1.0 pour éviter la confusion avec le produit « Saint ».
2005	Nagios version 2.0

Développement

Nagios est développé sous licence GPL (http://www.gnu.org/copyleft/gpl.html).
Son développement est supervisé par l'initiateur du projet, Ethan Galstad.

Portages

Nagios a été porté sur les plate-formes suivantes :

- plate-forme d'origine : Linux
- Unix propriétaires : AIX, HP-UX, Solaris, Irix, Compaq Tru64 UNIX (Digital UNIX), SCO Unixware
- Unix open-source : FreeBSD, NetBSD, OpenBSD, FREESCO

Présentation

Qu'est-ce que Nagios

Nagios est un superviseur de machines (les *hôtes*) et de services.
Il fonctionne en tant que démon Unix. A intervalles réguliers, il lance des sondes qui testent l'état des hôtes.
Les remontées d'alertes se font d'une manière définie par l'utilisateur (e-mail, message, SMS ou autre).

Nagios présente une interface graphique accessible par un navigateur web. Elle permet de :

- visualiser l'état du réseau de façon graphique à travers un carte de statut (*statusmap*)
- soumettre des requêtes (commandes externes ou *passive checks*)
- générer des rapports d'activité (par hôte, par service...)
- visualiser les historiques
- modifier la file d'attente des vérifications

Ce que n'est pas Nagios

Dans son fonctionnement standard, Nagios ne propose pas d'analyse quantitative des ressources car il ne sait traiter que leurs *états*. Nagios n'est pas non-plus un analyseur de vulnérabilité.

Fonctionnalités standards

Supervision

Nagios peut être utilisé pour superviser les éléments suivants :

- services réseau (HTTP, SSH, POP3...)
- ressources système (charge CPU, utilisateurs connectés...)
- état du matériel (température...)
- état des périphériques (i2c...)

Quelques autres possibilités

- décrire un réseau de façon hiérarchisée (groupes d'hôtes, de services, dépendances ...)
- exécuter des commandes externes (contrôlable via une fifo)
- modification des files d'attentes des vérifications à effectuer
- suspendre des vérifications
- persistance des statuts entre les redémarrages de Nagios
- authentification simple à travers le serveur web, et définitions de rôles selon les utilisateurs
- appliquer des mesures correctives grâce à un gestionnaire d'événements (*event handler)*
- intégration aisée dans d'autres outils de gestion de parc
- développer ses propres plugins avec n'importe quel langage (script-shell, perl, C ...)

Les plugins

Nagios est constitué de trois parties coopératives entre elles :

- un coeur, le serveur nagios : c'est un ordonnanceur performant lancé sous forme de daemon unix
- des "exécutants", les plugins (stockés dans nagios/libexec/ ou nagios/plugins/)
- une interface de contrôle (sous forme d'application CGI)

Les plugins sont des programmes externes au serveur, qui font des requêtes vers les hôtes, et transmettent au serveur un code retour et d'éventuels messages courts.

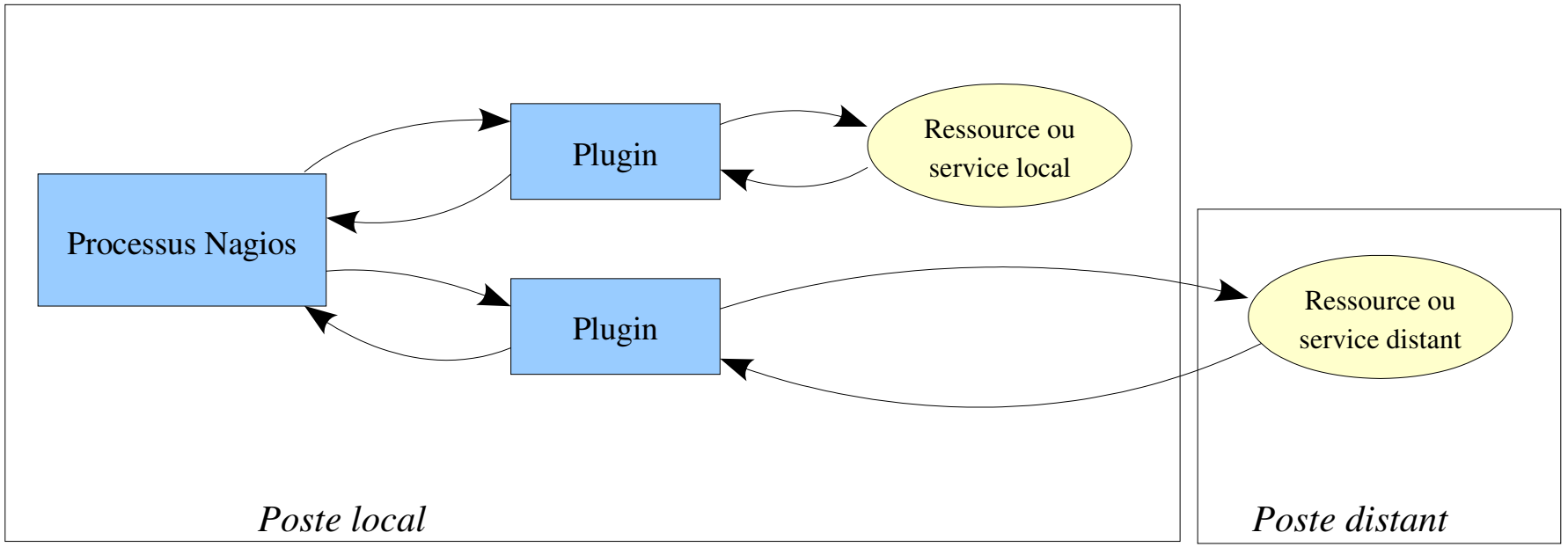
Par défaut, les plugins sont ordonnancés par le serveur nagios (*active check*) mais ils peuvent être exécutés à la demande ou bien en réponse à une alerte (*passive check*).

Quelques plugins standards :

```
check_http      vérifie la présence d'un serveur web
check_load      vérifie la charge CPU locale
check_ping      envoie une requête ping à un hôte
check_pop       vérifie la présence d'un serveur POP3
check_procs     compte les processus locaux
check_sensors   récupère des informations sur des capteurs industriels
check_smtp      vérifie la présence d'un serveur SMTP
check_snmp      envoie une requête SNMP (passée en argument) à un hôte
check_ssh       vérifie la présence d'un service SSH
check_tcp       vérifie l'ouverture d'un port TCP (passé en argument)
check_users     compte le nombre d'utilisateurs sur la machine locale
```

Les plugins

Nagios fait appel aux plugins, qui se chargent d'effectuer leur requête vers le poste local ou vers un poste distant. Les plugins renvoient ensuite la réponse et le code de retour au processus Nagios.



Les gestionnaires d'événements

Ce sont des scripts shell (par défaut stockés dans nagios/libexec/eventhandlers/ ou nagios/plugins/eventhandlers/ selon l'installation) qui permettent de corriger automatiquement un problème survenu sur un service.

Leur utilisation est facultative. Elle peut être utile dans le cas de gestion de « failover ».

Exemple :

```
code retour "warning"     recharger le service
code retour "critical"    lancer le service failover
code retour "ok"          ne rien faire
```

En général, il faudra donner à Nagios la possibilité d'exécuter des commandes sur des hôtes distants.

- Les processus correspondant à Nagios sont exécutés en tant que UID nagios / GID nagios : il faut mettre en place *sudo* pour permettre à l'utilisateur nagios de manipuler des scripts d'administration

Les extensions

De nombreux projets visant à rajouter des fonctionnalités non standard à Nagios ont vu le jour.
Le site nagiosexchange regroupe ces projets :

```
http://www.nagiosexchange.org/
```

Les plus intéressants sont :

- NRPE : Nagios Remote Plugin Execution, permet d'exécuter des plugins Nagios sur une machine distante. Utile pour les plugins qui doivent obligatoirement être placés sur la machine à interroger.
- NSCA : Nagios Service Check Acceptor, permet à Nagios d'intercepter des messages envoyés par une machine. Nécessite d'exécuter un démon NSCA sur le serveur Nagios (check passif)
- Nagios-popups : envoie les alertes sous forme de popups sur n'importe quelle machine. Cette extension utilise le service de messages Windows et xmessage sur Unix
- Apan, Nagiosgraph : affichent des graphes quantitatifs des ressources surveillées grâce à *rrdtool*
- Fruity, nagedit-nagview, NaWui : outils de configuration de Nagios
- Oreon, openQRM : interfaces web avancées de gestion réseau intégrant Nagios pour la supervision

Installation de Nagios

Introduction

La mise en place d'un moniteur Nagios nécessite :

- un serveur web (par exemple Apache)
- un serveur nagios
- des plugins nagios (qui peuvent être déployés sur des clients)
- d'éventuelles extensions

Ces quatres types de produits sont livrés séparément.

Enfin, pour des raisons de sécurité, le processus Nagios s'exécute en tant qu' « utilisateur nagios » et « groupe nagios » (par défaut). Ceci nécessite la création d'un utilisateur nagios et de son groupe sur les systèmes :

```
[root@servnagios /root]# groupadd -g 5666 nagios
[root@servnagios /root]# useradd -u 5666 -g nagios -d /usr/local/nagios nagios
```

Il est recommandé d'utiliser les mêmes UID/GID sur le serveur nagios et les clients nagios (en cas de mise en place des tcp-wrappers et identification avec identd).

Installation et mise à jour de Nagios

A partir de paquets

S'il existe des paquets pour votre distribution de Linux, l'installation et la mise à jour se font à partir de votre gestionnaire de paquets.

Sur Debian[1]

- installation :
```
[root@servnagios /tmp]# apt-get install nagios
```
- mise à jour :
```
[root@servnagios /tmp]# apt-get update && apt-get install nagios
```

Sur RedHat, Mandriva, SuSE et autres systèmes utilisant les packages rpm

- installation :
```
[root@servnagios /tmp]# rpm -i nagios-<version>.rpm
```
- mise à jour :
```
[root@servnagios /tmp]# rpm -U nagios-<version>.rpm
```

attention à la satisfaction de dépendances entre paquets

Tous les paquets RPM concernant Nagios pourront être trouvés sur le site suivant :

```
http://dag.wieers.com/home-made/apt/packages.php
```

1 Debian propose trois variantes de nagios équivalentes; la version standard est « nagios-text ».

Installation et mise à jour de Nagios

A partir des sources

On récupèrera les sources sur le site officiel de Nagios (http://www.nagios.org)

Pour l'affichage graphique (carte de statut ou *statusmap*), on aura besoin de la bibliothèque gd (http://www.boutell.com). Pour installer cette bibliothèque à partir de ses sources, les bibliothèques suivantes sont nécessaires :

- libpng-devel
- libjpeg-devel
- freetype-devel
- Xfree86-devel

Les fichiers .so (bibliothèques partagées) de gd sont installées par défaut dans /usr/local/lib.

vérifier que /usr/local/lib est présent dans le fichier de configuration de l'éditeur de liens /etc/ld.so.conf et mettre à jour la liste des liens avec ldconfig -v

- On doit d'abord décompresser l'archive téléchargée dans un répertoire temporaire (par exemple /tmp) :

```
[root@servnagios /tmp]# tar xzf nagios-<version>.tar.gz
```

Installation et mise à jour de Nagios

- Configuration des sources :

options utiles de configure (défaut entre crochets) :

--help	aide en ligne
--prefix=PREFIX	racine d'installation [/usr/local/nagios]
--exec-prefix=EPREFIX	racine d'installation pour les exécutables [=PREFIX]
--bindir=DIR	exécutables [EPREFIX/bin]
--sbindir=DIR	exécutables d'administation/sytème [EPREFIX/sbin]
--libexecdir=DIR	exécutables du programme [EPREFIX/libexec]
--datadir=DIR	données en lecture seule [PREFIX/share]
--sysconfdir=DIR	configuration [PREFIX/etc]
--localstatedir=DIR	fichiers journaux, de statut [PREFIX/var]
--libdir=DIR	bibliothèques partagées [EPREFIX/lib]
--with-nagios-user=<user>	propriétaire du processus nagios [nagios]
--with-nagios-grp=<grp>	groupe du processus nagios [nagios]
--with-init-dir=<path>	répertoire pour le script d'init
--with-template-extinfo	la configuration étendue utilise des modèles
--with-template-objects	la configuration des objets utilise des modèles
--with-cgiurl=<dir>	URL pour les cgi [/nagios/cgi-bin]
--with-htmurl=<dir>	URL pour html [/nagios]

Installation et mise à jour de Nagios

- Configuration des sources (suite) :

```
[root@servnagios /tmp]# cd nagios-<version>
[root@servnagios /tmp/nagios]# ./configure <options_eventuelles>
```

- Compilation de nagios et des CGIs :

```
[root@servnagios /tmp/nagios]# make all
```

- Installation dans le répertoire défini avec *configure* :

```
[root@servnagios /tmp/nagios]# make install
```

- Installation du script d'init :

```
[root@servnagios /tmp/nagios]# make install-init
```

Installation des plugins

Les plugins nécessitent l'installation d'openssl et d'un client SNMP (ucd-snmp par exemple).

A partir de packages

Les plugins doivent *a priori* être préparés pour l'architecture sur laquelle ils s'exécuteront, aussi on a tout intérêt à les installer à partir de packages téléchargés sur nagios.org :

Exemple :

```
[root@servnagios /tmp/]# rpm -i nagios-plugins-<version>.rpm
```

Si vous effectuez l'installation à partir de packages attention aux dépendances, nagios-plugins nécessite, entre autres, plusieurs modules perl. Tous les packages peuvent être trouvés sur le site http://dag.wieers.com/home-made/apt/packages.php

A partir des sources

La démarche est la même que pour la compilation/installation de nagios.
Les plugins sont installés par défaut dans PREFIX/libexec

ces plugins pourront être appelés à être déployés sur des clients, ce point sera abordé dans le chapitre déploiement

Installation des plugins

Attention, en fonction de la distribution de Linux utilisée et de la version des plugins un bogue peut apparaître, qui empêche la bonne interprétation du résultat des pings par le plugin check_ping.
Ceci est dû à une erreur de localisation (il en est de même pour le système Solaris de Sun).

Pour résoudre le problème il faut modifier dans les sources (qui sont situées dans le répertoire *plugins* du répertoire d'installation) le fichier *check_ping.c*.

Remplacer la ligne :

```
setlocale (LC_ALL, "");
```

par :

```
setlocale (LC_ALL, "C");
```

puis compilez et finissez normalement l'installation.

Démarrage de Nagios

Serveur Nagios

Le moniteur de surveillance se démarre à l'aide du script d'init /etc/rc.d/init.d/nagios.
Ce script peut être appelé en plus des options standards (start|sop) avec

- restart
- reload (à appeler lorsque l'on a modifié les fichiers de configuration)
- status

Nagios peut être lancé automatiquement au démarrage, il suffit de le rajouter dans les services à démarrer pour le runlevel souhaité :

Exemple pour le runlevel 3 :

```
[root@servnagios /etc/rc3/d/]# ln -s ../init.d/nagios S99nagios
```

Démarrage du serveur Web

Si vous utilisez Apache, il est lancé par le script d'init /etc/init.d/httpd.

Utilisation de nagios

Page d'accueil

La page d'accueil de Nagios est normalement accessible via l'url : *http://<adresse_serveur_nagios>/nagios*

Le menu de gauche permet d'accéder à tous les scripts cgi de supervision du réseau et à la documentation de Nagios.

Pour revenir à la page d'accueil depuis une autre page, cliquer sur « Home ».

Les CGI

Nagios fait appel à des scripts CGI pour toutes ses opérations de supervision et de génération de rapports. Les informations données par les CGI sur l'état du réseau dépendent des droits que possède l'utilisateur connecté.

Selon la configuration de Nagios, les opérations faisant appel à des commandes externes peuvent ne pas être autorisées.

Vue d'ensemble de l'état du réseau

La CGI « Tactical Overview » permet d'avoir une vue globale de l'état du réseau. Elle permet de détecter rapidement des pannes de réseau et de surveiller l'état des hôtes et services.

L'état général de tous les hôtes et services est indiqué par l'affichage « Network Health ».

Il est possible de cliquer directement dans la page sur les problèmes (qui apparaissent sur fond rouge) pour obtenir plus de détails.

Détail des hôtes et services

Les pages « Service Detail » et « Host Detail » affichent la liste de tous les hôtes et services surveillés ainsi que leur état. Si un hôte ou un service rencontre un problème, regarder les informations données par la colonne « Status Information » pour déterminer la cause de la panne.

Exemple de liste des services et de leur status :

Les pages « Service Problems » et « Host Problems » affichent les mêmes informations mais en cachant les hôtes et services dont le status est « OK ».

Les liens « Hostgroup xxx » et les liens « Servicegroup xxx » permettent d'accéder aux informations de status des hôtes et services en les regroupant selon les groupes définis par les fichiers de configuration.

Carte du réseau

Les liens « Status Map » et « 3-D Status Map » permettent d'afficher une représentation graphique du réseau, respectivement en 2D et en 3D. Ceci permet notamment de voir l'état des hôtes, les dépendances, et d'accéder directement aux détails pour un hôte donné en cliquant dessus.

Status Map

Ce CGI crée une carte de tous les hôtes du réseau surveillés par Nagios, en utilisant la librairie gd. Pour changer de type de représentation, utiliser la liste déroulante « Layout Method ». Les représentations suivante sont disponibles :

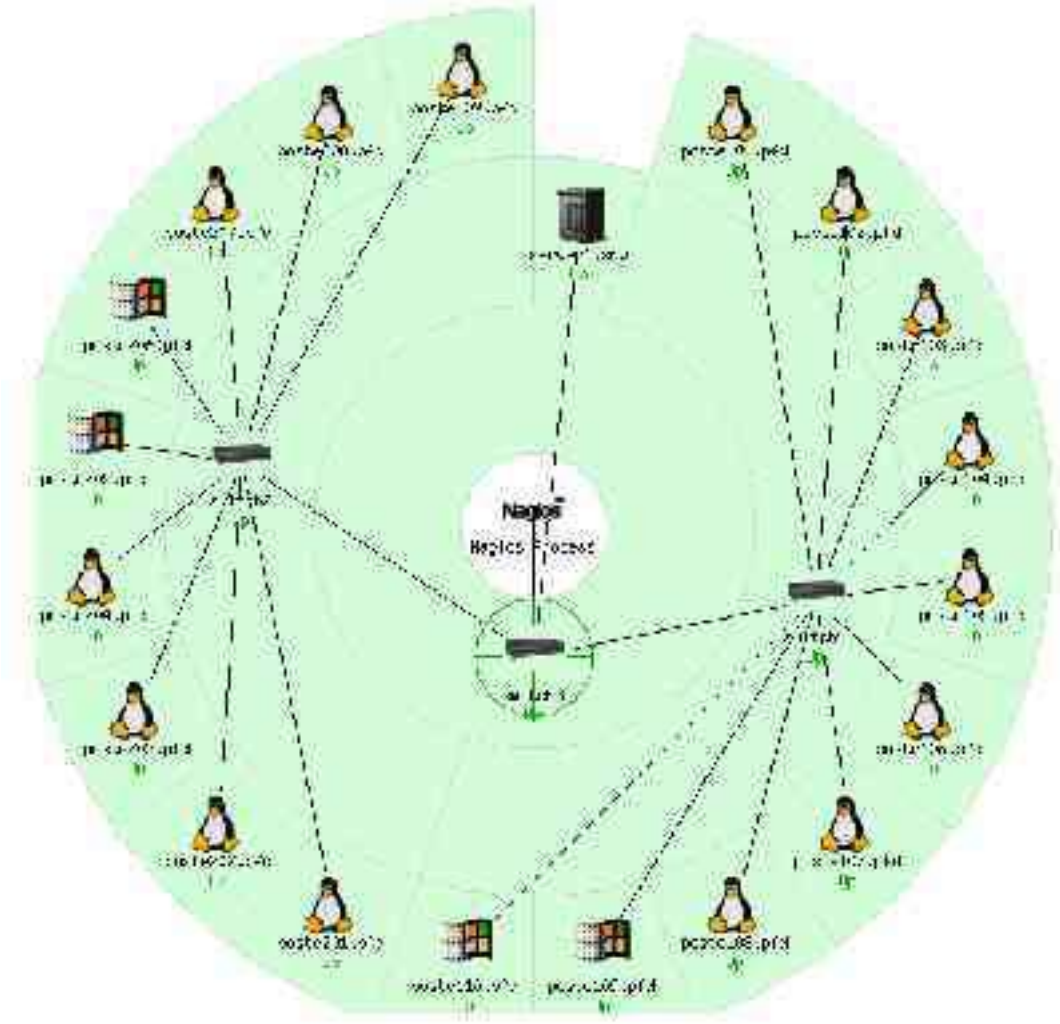
- User-supplied coords : coordonnées utilisateur
- Depth layers : représentation par couches
- Collapsed tree : arbre fermé
- Balanced tree : arbre équilibré
- Circular : représentation circulaire
- Circular (Marked Up) : circulaire avec zones colorées
- Circular (Balloon) : autre représentation circulaire

3-D Status Map

Ce CGI crée la représentation des hôtes en 3D VRML. La visualisation n'est possible que si un navigateur VRML (Virtual Reality Modeling Language) est installé sur le système.

Carte du réseau

Exemple de rendu du CGI statusmap :



Détection des pannes réseau

La page « Network Outages » répertorie la liste des problèmes sur les hôtes qui provoquent des pannes réseau.
Ceci permet sur de grands réseaux de détecter rapidement la source du problème, mais n'est pas d'une grande utilité pour les réseaux de petite taille.

Les hôtes sont triés dans la liste an fonction de la gravité de la panne causée, les problèmes les plus importants apparaissent en haut de liste.

Recherche d'un hôte et commentaires

Recherche d'un hôte

Le champ « Show Host » de la colonne de menu permet d'accéder directement au détail d'un hôte en saisissant son nom ou son adresse.

Commentaires

Le lien « Comments » permet de lister les commentaires associés aux hôtes et aux services, d'en supprimer ou d'en ajouter de nouveaux.

Arrêt d'hôtes et de services

Si nécessaire, il est possible d'indiquer à Nagios des périodes durant lesquelles un hôte ou un service sera arrêté.
Nagios ne lancera alors aucune vérification sur un hôte ou service déclaré arrêté.

Pour programmer des arrêts d'hôtes ou de services, cliquer sur « Downtime » dans le menu. Remplir les champs nécessaires et valider (bouton « Commit »).

Il y a deux types d'arrêts programmés :

- « Fixed » indique que Nagios ne doit plus effectuer de vérification pour cet hôte ou service entre deux instants précis,
- « Flexible » indique que Nagios doit attendre que l'hôte ou service soit dans un autre état que « OK » pour le considérer arrêter.

Dans ce cas, la période d'arrêt est indiquée dans le champ « If Flexible, Duration: ».

Les dates sont au format :

```
mm/dd/yyyy hh:mm:ss
```

Processus Nagios, performances

Le lien « Process Info » permet d'accéder à la page de gestion du processus Nagios.
Diverses informations sont affichées dans la section « Process Information », comme la durée depuis laquelle le processus est lancé, son PID, les paramètres activés, etc.

La colonne de droite (« Process Commands ») permet d'arrêter ou de redémarrer Nagios, et d'activer ou de désactiver plusieurs paramètres.

Le lien « Performance Info » permet d'accéder aux informations de performance des vérifications effectuées par Nagios.

Liste des vérifications programmées

Pour accéder à la liste des vérifications programmées, cliquer sur « Scheduling Queue ». La liste de toutes les vérifications apparaît, avec pour chacun l'hôte et le service concerné, l'heure de la dernière vérification effectuée par Nagios ainsi que l'heure de celle à venir.

Pour supprimer un évènement programmé, cliquer sur l'icône : 

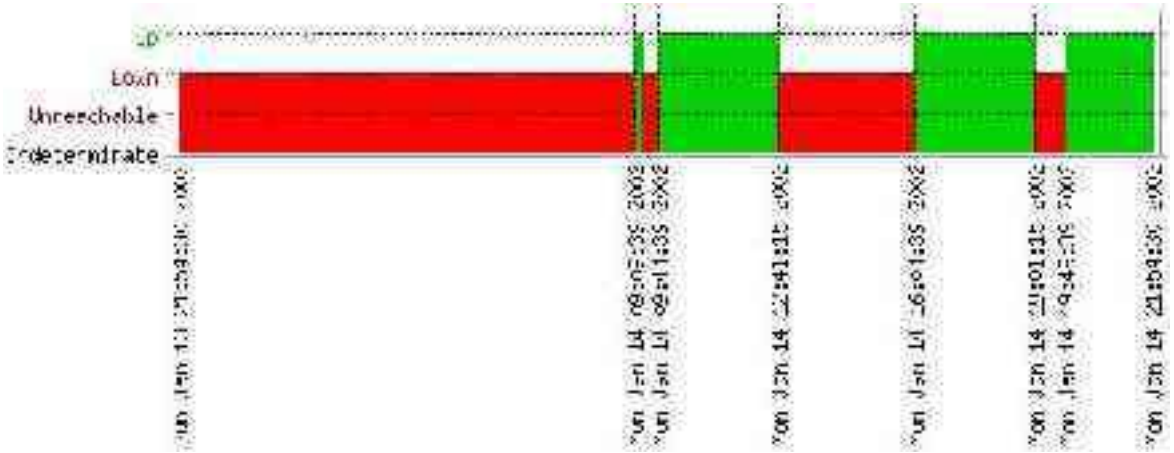
Pour modifier l'horaire de la prochaine vérification, cliquer sur l'icône : 

Rapports

La section « Reporting » du menu permet de générer automatiquement rapports sous diverses formes.

Trends

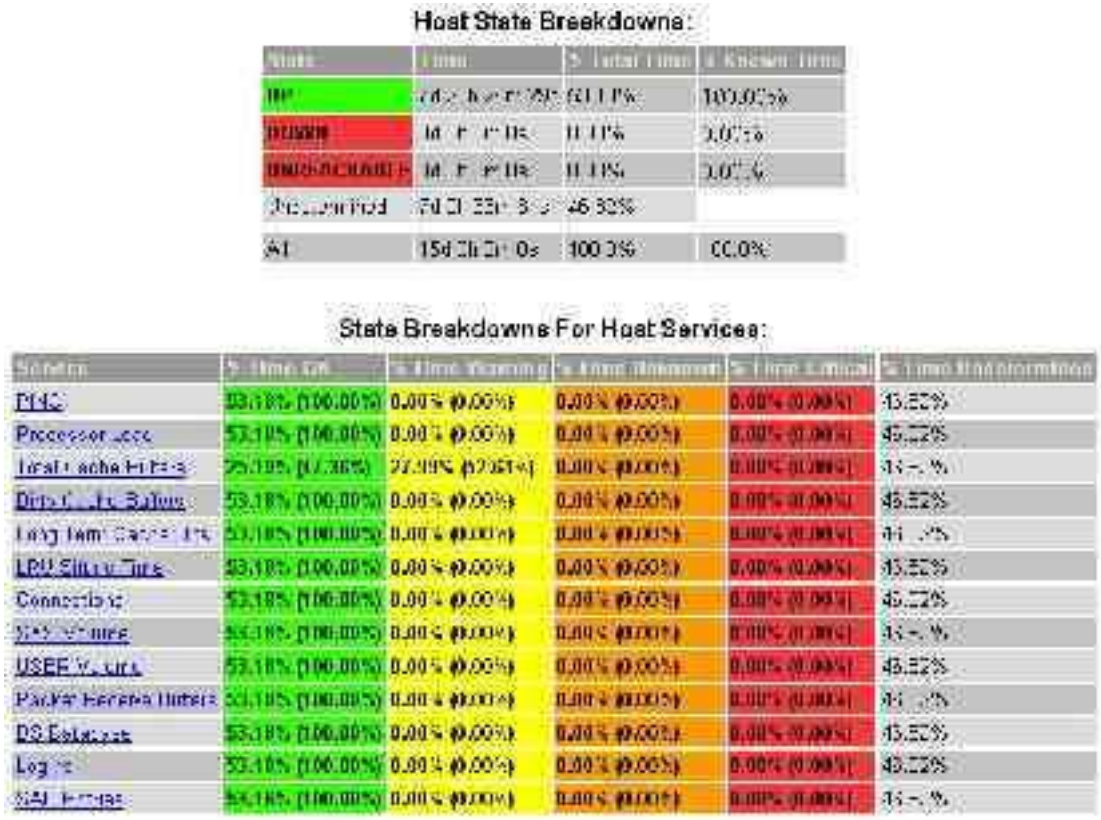
Permet de créer un graphique de l'état d'un hôte ou d'un service sur une période définie (utilise la librairie gd).



Rapports

Availability

Ce CGI est utilisé pour obtenir un rapport sur la disponibilité des hôtes et services sur une période définie.



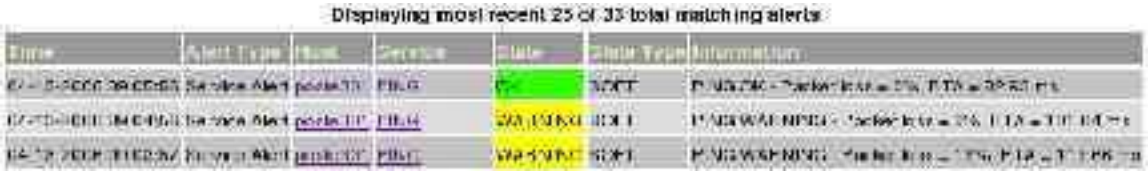
Rapports

Alert Histogram

Idem à « Availability », mais le rapport est sous la forme d'un histogramme.

Alert Summary

Ce CGI fournit des rapports concernant les alertes déclenchées par les hôtes et services.



Notifications

Dresse la liste des notofications, c'est-à-dire des alertes envoyées aux personnes, le plus souvent par mail.

Rapports

Event Log

Ce CGI affiche le fichier de log dans une page HTML. Les évènements sont regroupés par heure et des icônes indiquent pour chacun quel type d'évènement il d'agit (warning, notification, information etc).

Alert History

Permet de filtrer le fichier de log pour n'afficher, par exemple, que les évènements concernant les services, ou uniquement les alertes de type « Host unreachable » (Hôte indisponible).

Les paramètres de filtrage se situent en haut à droite de la fenêtre.

Lecture de la configuration

View config

Ce CGI permet de voir le contenu des objets définis dans les fichiers de configuration.
Les objets peuvent être les hôtes, les groupes d'hôtes, les contacts, les groupes de contacts, les services etc.

Il est, par contre, impossible d'éditer la configuration via l'interface web. Il existe cependant un projet récent de réécriture totale de l'interface de Nagios en php qui prévoit cette fonctionnalité assez demandée (voir le site web http://nagios-php.sf.net).

Il existe aussi des systèmes de gestion réseau qui intègrent le coeur de Nagios pour la fonction de supervision. Ils possèdent leur propre interface web de contrôle et permettent entre autres de définir les hôtes et services à surveiller directement depuis leur interface.

On pourra retenir les systèmes de gestion suivants :

- Oreon : http://www.oreon-project.org/
- OpenQRM : http://www.openqrm.org/

Configuration de Nagios

Configuration de Nagios

La configuration d'un système de supervision basé sur nagios s'articule autour de quatre grands axes :

Serveur de supervision

- configuration du serveur nagios lui-même (l'ordonnancement des tests)
- configuration du serveur web (l'accès au contrôle depuis un navigateur)

Hôtes clients

- configuration des extensions (nrpe, nsca)
- configuration des tcp wrappers

Démarrage / Arrêt

Typiquement, nagios sera lancé par les commandes d'init SYSV.

Exemple de commandes SYSV:

```
# /etc/init.d/nagios help
Usage: /etc/init.d/nagios {start|stop|status|reload}
# /etc/init.d/nagios start
Starting service nagios:                              [  OK  ]
```

Configuration de Nagios

Remarque

Nagios est très sensible à la syntaxe de ses fichiers de configuration et ne tolère aucune erreur.
Les fichiers de définition d'objets sont tous interdépendants.

Lors de l'élaboration des fichiers de configuration, on peut vérifier la syntaxe avec :

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

Fichiers d'exemples

Sur le serveur, des fichiers d'exemples sont crées dans /etc/nagios. Il est recommandé de les sauvegarder avant de travailler :

```
[root@servnagios /etc/nagios]# mkdir samples ; cp *.cfg samples
```

Les fichiers de configuration

Les fichiers de configuration se situent dans le répertoire /etc/nagios. Ils doivent être indiqués dans le fichier de configuration principal nagios.cfg pour être pris en compte :

```
cfg_file=/etc/nagios/checkcommands.cfg
cfg_file=/etc/nagios/hosts.cfg
cfg_file=/etc/nagios/services.cfg
...
```

Il obéissent tous à la même syntaxe et peuvent contenir les mêmes paramètres.
Il n'est pas obligatoire d'utiliser un grand nombre de fichiers pour la configuration, mais le découpage en plusieurs fichiers rend la lecture de la configuration plus aisée.

configuration générale

- cgi.cfg le fichier de configuration pour les scripts CGI
- nagios.cfg le fichier de configuration principal de nagios
- resource.cfg ressources diverses et variables d'environnement

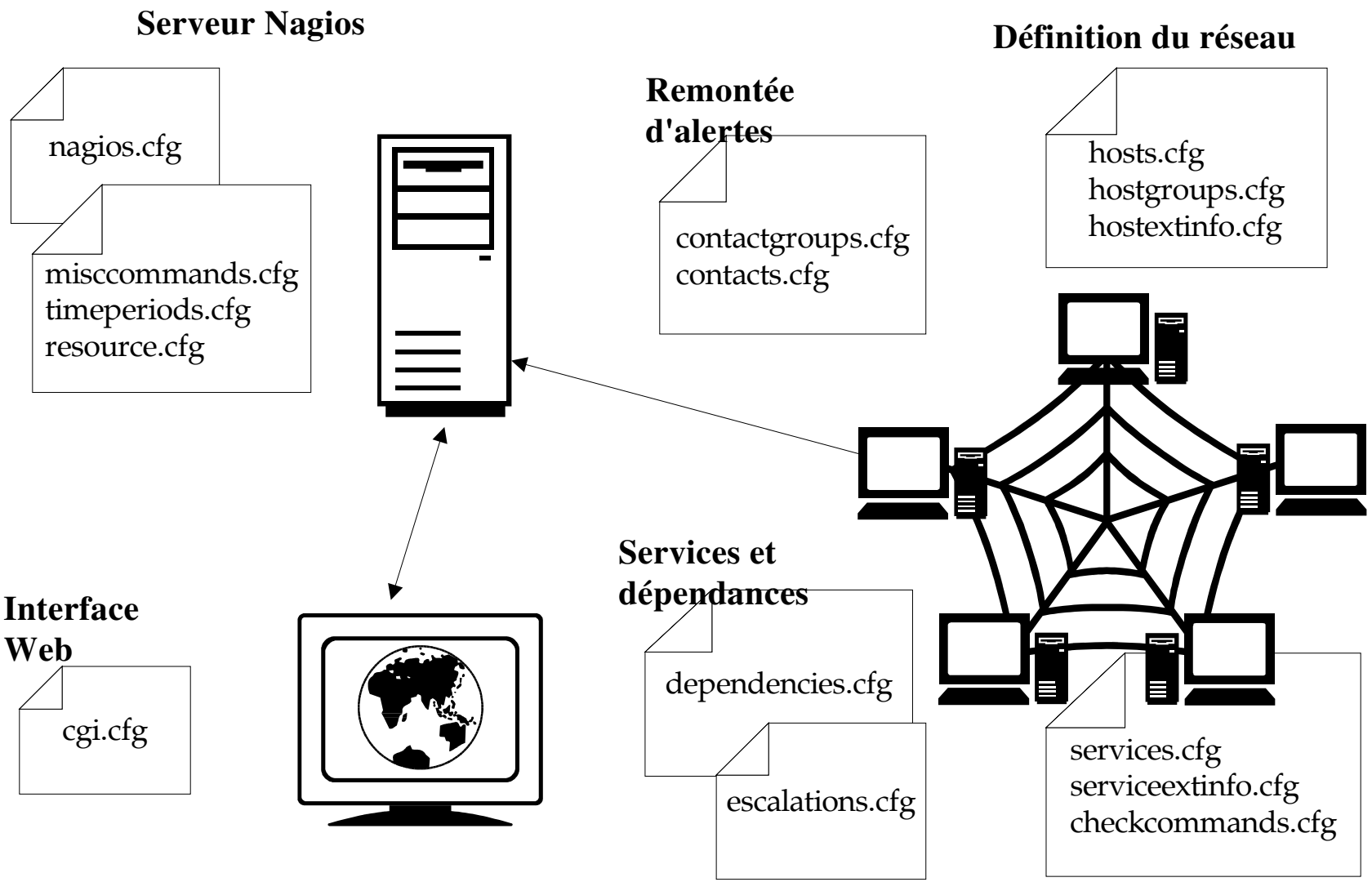
Les fichiers de configuration

configuration d'objets

Ceux-ci utilisent une configuration *basée sur modèle*. On entend par objet toutes les données telles que les services à surveiller, les hôtes... dont la configuration peut être extrêmement souple :

• hosts.cfg	définition du réseau, machine par machine, avec les ascendants
• hostgroups.cfg	regroupement d'hôtes par fonctions (dmz, production...)
• hostextinfo.cfg	détails sur les hôtes (icônes, descriptions...)
• dependencies.cfg	dépendances entre hôtes et services
• escalations.cfg	escalades de notification
• contactgroups.cfg	groupes de contact pour les alertes (admin, pupitreur...)
• contacts.cfg	personnes à contacter pour les alertes
• checkcommands.cfg	commandes de supervision (check_ftp, check_pop...)
• misccommands.cfg	commandes diverses (expédition de courrier...)
• services.cfg	services supervisés par nagios (POP, FTP...)
• servicegroups.cfg	groupes de services (seulement Nagios v2)
• serviceextinfo.cfg	détails sur les services (icônes...)
• timeperiods.cfg	périodes des supervision (24h/24, tous les jours...)

Les fichiers de configuration



Principe des modèles de configuration

Nagios utilise des fichiers de configuration basés sur des modèles (template-based configuration). Ce type de configuration consiste à définir des modèles pour des objets génériques, puis à les surcharger, les compléter, ou créer d'autres modèles.

Le squelette d'un modèle est le suivant :

```
define type_objet{
        name              <nom_du_modèle>  ; nom de l'objet
        <paramètre1>      <valeur1>
        <paramètre2>      <valeur2>
        ...
        register          0                ; on n'enregistre pas la définition
                                           ; car c'est un modèle
        }
```

Principe des modèles de configuration

Pour utiliser un modèle, il suffit de l'invoquer avec le mot-clé *use* :

```
define type_objet{
        name            <nom_de_l_objet> ; on surcharge le nom de l'objet
        use             <nom_du_modèle>  ; on utilise un modèle déjà défini
        <paramètre1>    <valeur1>
        <paramètre2>    <valeur2>
        ...
#pas de register, implicitement : register 1
        }
```

Deux types de commentaires sont possibles, "#" en premier caractère de ligne, sinon ";"

Des exemples concrets seront vus avec les fichiers de configuration.

Outils graphiques de configuration

Il existe plusieurs projets basés sur Nagios qui permettent une configuration facilitée, entièrement via une interface web. La configuration s'effectue ainsi de façon plus aisée et intuitive qu'en éditant tous les fichiers de configuration avec un éditeur de texte.

On pourra retenir les outils suivants :

- Nagedit-Nagview (Stable) : http://nagedit-nagview.sf.net/
- Fruity (Beta) : http://fruity.sf.net/
- NaWui (Alpha) : http://stigma.dyndns.org/NaWui

VII

Configuration d'Apache

Fichiers de configuration

Fichiers à modifier (apache)

Vérifier que le fichier de configuration d'Apache (httpd.conf) inclut celui qui concerne nagios.

Exécution des CGI

Le fichier nagios.conf devra contenir les instructions nécessaires pour accéder aux scripts CGI.

Exemple de configuration apache pour les CGI de l'interface de contrôle Nagios:

```
ScriptAlias /nagios/cgi-bin "/usr/lib/nagios/cgi"
<Directory "/usr/lib/nagios/cgi">
        Options ExecCGI
        AllowOverride none
        Order deny, allow
        Deny from all
        Allow from 127.0.0.1
        Auth Name "Accès Nagios"
        AuthType Basic
        AuthUserFile "/etc/nagios/htpasswd.users"
        Require valid-user
</Directory>
```

Dans cet exemple, nous n'autorisons l'accès aux CGI que depuis la machine locale.

Fichiers de configuration

Accès aux pages html

Le fichier de configuration apache nagios.conf devra contenir une déclaration de type Directory pointant sur le chemin réel de des fichiers html de Nagios.

Exemple de configuration apache pour les pages html de l'interface de contrôle Nagios :

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

<Directory "/usr/share/nagios">
        Options none
        AllowOverride none
        Order deny, allow
        Deny from all
        Allow from 127.0.0.1
        AuthName "Accès Nagios"
        AuthType Basic
        AuthUserFile /etc/nagios/htpasswd.users
        Require valid-user
</Directory>
```

Dans cet exemple, nous n'autorisons l'accès que depuis la machine locale.

Authentification

Nous avons spécifié que les visiteurs doivent s'authentifier pour pouvoir accéder à l'interface de contrôle.

Il est nécessaire de créer le fichier htpasswd.users qui contiendra la liste des personnes autorisées à accéder à l'interface web de Nagios. On utilisera pour cela l'utilitaire htpasswd fourni avec apache.

Exemple d'utilisation de htapsswd pour le compte « nagiosadmin » :

```
# htpasswd -c /etc/nagios/htpasswd.users nagiosadmin
```

D'autres utilisateurs peuvent être ajoutés par la suite. Pour cela, on utilisera le même utilitaire mais sans l'option '-c'. Attention, le fichier htpasswd.users doit être accessible en lecture par l'utilisateur apache seulement, et fermé à tout autre utilisateur.

Test de l'interface

Une fois Apache configuré, testez puis rechargez sa configuration :

```
# apachectl -t && /etc/init.d/httpd reload
```

Testez enfin l'accès à l'interface web de Nagios avec un navigateur quelconque.
Dans notre cas, l'adresse est http://127.0.0.1/nagios/

Le navigateur devra présenter une invitation à s'authentifier. Les identifiants sont ceux qui ont été générés précédemment avec l'utilitaire htpasswd.

- Attention : Si l'on utilise les fonctionnalités SELinux, il faudra en plus paramétrer le contexte de sécurité permettant au processus httpd d'apache d'accéder correctement aux CGI de Nagios.

VIII

Fichier de configuration principal

Le fichier nagios.cfg

Voici un exemple de fichier nagios.cfg, le fichier de configuration principal de Nagios. Chaque paramètre est commenté sur la partie gauche.

```
# emplacement du fichier journal      log_file=/var/log/nagios/nagios.log

# emplacement des fichiers de         cfg_file=/etc/nagios/checkcommands.cfg
# configuration d'objets              cfg_file=/etc/nagios/misccommands.cfg
                                      cfg_file=/etc/nagios/contactgroups.cfg
                                      cfg_file=/etc/nagios/dependencies.cfg
                                      cfg_file=/etc/nagios/escalations.cfg
                                      cfg_file=/etc/nagios/hostgroups.cfg
                                      cfg_file=/etc/nagios/hosts.cfg
                                      cfg_file=/etc/nagios/services.cfg
                                      cfg_file=/etc/nagios/timeperiods.cfg

# emplacement du fichier de           resource_file=/etc/nagios/resource.cfg
# configuration des ressources

# journal des états                   status_file=/var/log/nagios/status.dat

# utilisateur et groupe du            nagios_user=nagios
# propriétaire du processus           nagios_group=nagios
```

nagios.cfg

```
# utilisation de commandes          check_external_commands=0
# externes. Certaines opérations
# accessibles depuis l'interface
# web utilisent des commandes
# externes (ex: redémarrage du
# processus nagios, modification
# de la file d'attente des
# vérifications, etc)

# intervalle de vérification des    command_check_interval=-1
# commandes externes. Mettre cette
# valeur à -1 pour que nagios
# vérifie le fichier des commandes
# externes aussi souvent que
# possible

# fifo de commandes externes (doit  command_file=/var/log/nagios/rw/nagios.cmd
# être présent et accessible en
# écriture si les commandes
# externes sont activées!)

# journal de commentaires           comment_file=/var/log/nagios/comments.dat

# suspension de monitoring          downtime_file=/var/log/nagios/downtime.dat

# fichier verrou                    lock_file=/var/run/nagios.pid

# fichier temporaire                temp_file=/var/log/nagios/nagios.tmp
```

nagios.cfg

```
# rotation des journaux              log_rotation_method=d
# (découpage) :
# n = aucune, h = horaire,
# d = quotidienne (à minuit)
# w = hebdomadaire (samedi à
# minuit), m = mensuelle (minuit
# du dernier jour)

# chemin d'archivage des logs        log_archive_path=/var/log/nagios/archives

# utilisation des journaux système   use_syslog=1

# options diverses des journaux      log_notifications=1
                                     log_service_retries=1
                                     log_host_retries=1
                                     log_event_handlers=1
                                     log_initial_states=0
                                     log_external_commands=1
                                     log_passive_service_checks=1

# permet de définir une commande     #global_host_event_handler=somecommand
# appelée à chaque changement        #global_service_event_handler=somecommand
# d'état d'un hôte ou d'un service
```

nagios.cfg

```
# méthode de calcul du délai entre    service_inter_check_delay_method=s
# deux vérifications d'hôtes ou de
# services :
# n = pas de délai (déconseillé!)
# d = "dumb",  délai de 1
# seconde
# s = "smart",  étalement optimisé
# des vérifications
# x.xx = délai de x.xx secondes

# délai en minutes entre le moment    max_service_check_spread=30
# où le programme démarre et le
# moment où tous les services
# doivent avoir été vérifiés

# idem mais pour les hôtes            host_inter_check_delay_method=s
                                      max_host_check_spread=30

# facteur d'imbrication des           service_interleave_factor=s
# vérifications des services :
# s = "smart" (voir plus haut)
# x = facteur (1 = non entrelacé)

# nombre de vérifications en          max_concurrent_checks=0
# parallèle (0 = pas de limite)
# (1 = séquenciel)
# fréqence en seconde avec            service_reaper_frequency=10
# laquelle nagios traite les
# résultats des services qui ont
# été vérifiés
```

nagios.cfg

```
# pause entre services mis en file    sleep_time=0.25
# d'attente (en secondes) pendant
# laquelle nagios sera en sommeil


# timeout pour différentes            service_check_timeout=60
# commandes (en secondes)             host_check_timeout=30
                                      event_handler_timeout=30
                                      notification_timeout=30
                                      ocsp_timeout=5
                                      perfdata_timeout=5

# conservation de l'information de    retain_state_information=1
# statut (entre deux relances du
# service)

# fichier de sauvegarde du statut     state_retention_file=/var/log/nagios/retention.dat

# fréqence (en minutes) à laquelle    retention_update_interval=60
# nagios sauvegarde le status des
# hôtes et services

# détermine si nagios positionnera    use_retained_program_state=0
# diverses variables d'état du
# programme à partir des valeurs
# enregistrées dans le fichier de
# mémorisation
```

nagios.cfg

```
# nombre de secondes de l'unité
# de temps de base, utilisée
# dans toues les définitons
# de vérification / notification
interval_length=60

# vérification "agressive" (plus
# lente, mais plus fiable)
use_agressive_host_checking=0

# vérification des services au
# (re)démarrage
execute_service_checks=1

# accepter les vérifications
# passives
accept_passive_service_checks=1

# notifications au (re)démarrage
enable_notifications=1

# activer le gestionnaire
# d'évenement
enable_event_handlers=1

# détermine si nagios doit traiter
# ou non les données de
# performances des hôtes et
# services
process_performance_data=0

# indique dans quels fichiers les
# données de performances doivent
# être stockées
#host_perfdata_file=/tmp/host-perfdata
#service_perfdata_file=/tmp/service-perfdata
```

nagios.cfg

```
# détermine si nagios remonte les     obsess_over_services=0
# résultats de contrôles de
# service et lance la commande
# définie avec le paramètre
# suivant. A activer seulement en
# cas de supervision répartie

# définit la commande à lancer       #ocsp_command=somecommand
# après chaque contrôle de
# service.

# détermine si la détection des      check_for_orphaned_services=0
# services qui ont perdu leurs
# dépendances est activée

# si cette option est activée        check_service_freshness=1
# nagios vérifie que les checks
# passifs sont biens reçus

# fréquence en secondes avec         freshness_check_interval=60
# laquelle la vérification
# précédente est effectuée

# idem mais pour les hôtes           check_host_freshness=0
                                     host_freshness_check_interval=60
```

nagios.cfg

```
# autoriser une écriture              aggregate_status_updates=1
# asynchrone des changements de       status_update_interval=15
# statut (meilleures performances)

# permet d'activer la détection du    enable_flap_detection=0
# flapping, qui survient si un
# hôte ou un service change d'état
# trop souvent

# seuils de détection du flap pour    low_service_flap_threshold=5.0
# les hôtes et services               high_service_flap_threshold=20.0
                                      low_host_flap_threshold=5.0
                                      high_host_flap_threshold=20.0

# format de date                      date_format=us

# caractères interdits pour les       illegal_object_name_chars=`~!$%^&*|'"<>?,()=
# objets

# caractères interdits pour les       illegal_macro_output_chars=`~$&|'"<>
# macros

# active ou désactive la prise en    use_regexp_matching=0
# charge des expressions
# régulières dans les fichiers de
# configuration d'objets
```

nagios.cfg (fin)

```
# si cette option est désactivée,     use_true_regexp_matching=0
# une chaîne de caractères est
# prise en compte comme une
# expression régulière uniquement
# si elle contient le caractère
# '*' ou '?'. cette option n'a
# d'effet que si le paramètre
# précédent est activé.

# adresse email et pager de           admin_email=nagios
# l'administrateur                    admin_pager=pagenagios

# détermine si le démon nagios est    daemon_dumps_core=0
# autorisé ou pas à créer un core
# dump (sauvegarde d'image
# mémoire). Déconseillé, à part
# pour un débuggage.
```

Configuration de l'interface web

Le fichier cgi.cfg

Ce fichier configure le comportement de l'interface web de Nagios.

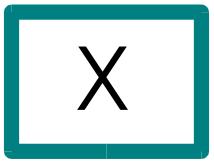
```
# emplacement du fichier de          main_config_file=/etc/nagios/nagios.cfg
# configuration générale

# chemin vers les fichiers html      physical_html_path=/usr/share/nagios

# chemin de nagios sur le serveur    url_html_path=/nagios
# web

# aide contextuelle pour les cgi     show_context_help=0

# commande de vérification du        nagios_check_command=/usr/local/nagios/libexec/check_nagios
# status du process nagios           /var/log/nagios/status.dat 5 '/usr/bin/nagios'

# indique si les cgi doivent         use_authentication=0
# utiliser un moyen
# d'authentification lorsqu'elles
# affichent des informations sur
# les hôtes et services

# utilisateur par défaut             #default_user_name=guest

# accès sytème/processus (vue)       authorized_for_system_information=nagiosadmin,theboss,jdoe

# utilisateurs ayant accès aux       authorized_for_configuration_information=nagiosadmin,jdoe
# informations de configuration
```

cgi.cfg

```
# accès au système/processus          authorized_for_system_commands=nagiosadmin
# (commandes)

# accès hôte/service (vue)            authorized_for_all_services=nagiosadmin,guest
                                      authorized_for_all_hosts=nagiosadmin,guest

# accès hôte/service (comandes)       authorized_for_all_service_commands=nagiosadmin
                                      authorized_for_all_host_commands=nagiosadmin

# image de fond de la statusmap       statusmap_background_image=smbackground.gd2

# aspect par défaut de la carte       default_statusmap_layout=5
# 0 = coordonnées utilisateur
# 1 = par couche
# 2 = arbre fermé
# 3 = arbre équilibré
# 4 = circulaire
# 5 = circulaire (zones colorées)

# aspect par défaut de la carte  en   default_statuswrl_layout=4
# wrl. 0, 2, 3 ou 4 (voir
# ci-dessus)
```

cgi.cfg

```
# définition de l'environnement wrl   #statuswrl_include=myworld.wrl
(pour les cartes VRML)
# définition du ping                  ping_syntax=/bin/ping -n -U -c 5 $HOSTADDRESS$

# taux de rafraîchissement des CGIs   refresh_rate=90
# (en secondes)

# optionnel, permet d'indiquer des    #host_unreachable_sound=hostdown.wav
# fichiers sons à utiliser lors de    #host_down_sound=hostdown.wav
# problèmes sur le réseau.            #service_critical_sound=critical.wav
                                      #service_warning_sound=warning.wav
                                      #service_unknown_sound=warning.wav
                                      #normal_sound=noproblem.wav
```



Configuration des hôtes et services

Hôtes

hosts.cfg

Ce fichier contient la description de base de tous les hôtes du réseau, avec les machines qui leur sont liées.

Exemple de définition générique d'un hôte :

```
define host{
        name                            generic-host
        Register                        0       ; ce n'est qu'un modele
        notifications_enabled           1       ; Notification activées
        event_handler_enabled           1       ; Un programme peut gerer l'evennement
        flap_detection_enabled          1       ;   Détecter   des   changements   trop
fréquemment
        process_perf_data               1       ; Sauvegarde des données detaillees de
                                                ; performances (utilisable par rrdtool)
        retain_status_information 1             ; Conserve les informations de status en
cas
                                                ; de redémarrage du programme
        retain_nonstatus_information      1       ; Conserve diverses informations
                                                ; en cas de redémarrage du programme
        }
```

Hôtes

Exemple d'hôtes connectés au switch sw0.pfd :

```
define host{
        use                     generic-host
        name                    commun
        parents                 sw0.pfd
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   480
        notification_period     24x7
        notification_options    d,u,r
        }
```

➢La directive « parents » liste les hôtes parents, généralement des switch, routeurs, passerelles, etc.

Exemple d'hôtes utilisant le modèle « commun » :

```
define host{
        use                     commun
        host_name               postepfd.pfd
        alias                   Serveur administratif
        address                 postepfd.pfd

define host{
        use                     commun
        host_name               calliope.delphes
        alias                   DNS formation
        address                 calliope.delphes
        }
```

Hôtes

Exemple d'hôte utilisant le modèle générique

```
define host{
        use                     generic-host
        host_name               sw0.pfd
        alias                   Switch pfd/delphes
        address                 sw0.pfd
        parents                 iris.pfd
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   60
        notification_period     24x7
        notification_options    d,u,r
        }
```

- On peut utiliser pour *address* un nom d'hôte logique (résolvable) ou directement une adresse IP.

Hôtes

hostgroups.cfg

Ce fichier définit des familles d'hôtes.

```
define hostgroup{
        hostgroup_name   exploitation
        alias            serveurs DMZ
        contact_groups   exploitation-admins
        members          servwas1.dmz,servwas2.dmz,mail.pythagore-fd.dmz
        }

define hostgroup{
        hostgroup_name   pythagore
        alias            Machines Pythagore
        contact_groups   pythagore-admins
        members          postepfd.pfd,servdeb.pfd,servtp5.pfd,iris.pfd
        }

define hostgroup{
        hostgroup_name   switches
        alias            Switches
        contact_groups   switch-admins
        members          sw0.pfd,switch2
        }
```

Hôtes

hostextinfo.cfg

Ce fichier définit des informations détaillées sur les hôtes. Les icônes des hôtes doivent être présentes dans le sous-répertoire « logos » du chemin des pages HTML (par défaut /usr/share/nagios/images/logos). Il est recommandé d'utiliser des images au format GD2 non compressé (on peut fabriquer ces images avec l'utilitaire *pngtogd2*). Les autres formats supportés sont JPEG, PNG et GIF.

Exemples :

```
define hostextinfo{
        name                    redhat
        icon_image              redhat.png
        icon_image_alt          station RedHat Linux
        statusmap_image         redhat.gd2 ; icône utilisée pour la carte
        register                0
        }
define hostextinfo{
        name                    servweb
        icon_image              globe.png
        icon_image_alt          serveur web
        statusmap_image         globe.gd2
        register                0
        }
define hostextinfo{
        use                     redhat
        host_name
poste102.delphes,poste105.delphes,poste205.delphes,poste102.delphes
        }
```

Hôtes

hostextinfo.cfg (suite)

 # ici, on n'utilise pas de modèle

```
define hostextinfo{
        icon_image              router40.png
        icon_image_alt          routeur pythagore
        gd2_image               router40.gd2
        host_name               iris.pfd
        }
```

 # utilisation simple de modèle

```
define hostextinfo{
        use                     servweb
        host_name               servwas1.dmz
        }
```

 # ici, on surcharge l'instruction "icon_image_alt"

```
define hostextinfo{
        use                     servweb
        icon_image_alt          server web backup
        host_name               servwas2.dmz
        }
```

Hôtes

escalations.cfg

Les escalades permettent de notifier différents contacts si un service ou un hôte surveillé persiste à ne pas fonctionner correctement. Une notification ne sera envoyée qu'à partir du moment où l'état est certifié (mode *hard*).

Exemple pour un service :

```
define serviceescalation{
        host_name                       linux1
        service_description             Current Users
        first_notification              5       ; notifier à partir de la 5e erreur
        last_notification               0       ; jusqu'à rétablissement du service
        contact_groups                  linux-admins,novell-admins      ; qui notifier
        notification_interval           0
        }
```

Exemple pour un groupe d'hôtes :

```
define hostgroupescalation{
        hostgroup_name                  novell-servers
        first_notification              2       ; notifier à partir de la 2e panne
        last_notification               5       ; jusqu'à la 5e
        contact_groups                  novell-admins,nt-admins  ; qui notifier
        notification_interval           10
        }
```

➢Il est possible de faire se chevaucher les plages d'escalades de notification

Services

services.cfg

Ce fichier définit les services à vérifier pour chaque hôte, en accord avec les définitions de commandes du fichier checkcommands.cfg. On peut répéter plusieurs fois un test avant de certifier qu'un service a changé d'état.

Définition des services (exemples)

```
define service{
        name                        generic ; Nom du modèle de service.
        register                    0       ; Ce n'est qu'un modèle.
        active_checks_enabled       1       ; Vérifications actives.
        passive_checks_enabled      1       ; Vérifications passives acceptées;
        parallelize_check           1       ; Parallèliser les tests.
        obsess_over_service         1       ; Pour la supervision distribuée (nsca).
        check_freshness             0       ; Dans ce cas, on ne vérifie pas si les
checks
                                            ; passifs sont reçus à la fréqence
voulue.
        notifications_enabled       1       ; Notifications activées
        event_handler_enabled       1       ; Exécution de commandes lors d'un
                                            ; changement d'état du service.
        flap_detection_enabled      1       ; Détection d'oscillation de l'état.
        process_perf_data           1       ; Prise en compte des données de
performance.
        retain_status_information           1    ; Conservation des informations de
status.
        retain_nonstatus_information        1    : Conservation des autres
informations.
        }
```

Services

```
define service{
        name                            commun
        use                             generic
        is_volatile                     0
        check_period                    24x7     ; Période de travail.
        max_check_attempts              6        ;  Tentatives  avant  de  passer
HARD
        normal_check_interval           5        ; intervalle de temps du test.
        retry_check_interval            1        ;   intervalle   de   temps   des
tentatives.
        contact_groups                  exploitation-admins
        notification_interval           120
        notification_period             24x7
        notification_options            c,r
        register                        0
        }
```

Exemple pour le service PING (utilise les valeurs par défaut du modèle 'commun') :

```
define service{
        name                            ping
        use                             commun
        service_description             PING
        check_command                   check_ping!100.0,20%!500.0,60%
        register                        0
        }
```

Services

Définition du service rusers :

```
define service{
        name                            rusers
        use                             commun
        service_description             Current Remote Users
        check_command                   check_remote_users!75!150
        register                        0
        }
```

Définition du service DNS :

```
define service{
        name                            dns
        use                             commun
        contact_groups                  pythagore-exploitation
        register                        0
        }
```

Services

Définition du service LPR :

```
#NB : cette définition surcharge celle de "commun"
#en cas d'arrêt du service, on invoque la commande restart_lp
define service{
        name                            lpr
        use                             commun
        service_description             Printer Status
        check_period                    workhours
        contact_groups                  imprimante-admins
        notification_interval           960
        notification_period             workhours
        notification_options            c,r
        check_command                   check_lp
        event_handler                   restart_lp
        register                        0
        }
```

- pour le *check_command*, le séparateur d'arguments est le point d'exclamation

Services

Utilisation des services (exemples)

Ici on complète la définition du service dns :

```
define service{
        use                     dns
        name                    dns-internet
        service_description     verification DNS
        host_name               iris.pfd
        check_command           check_dnsi!10.21.1.254
}
define service{
        use                     dns
        name                    dns-pfd
        service_description     verification DNS pfd
        host_name               iris.pfd
        check_command           check_dnsp!10.21.1.254
        }
```

Exemple d'utilisation de service sans redéfinition :

```
define service{
        use                     ping
        host_name               postepfd.pfd, calliope.delphes, poste102.delphes,
poste105.delphes, poste205.delphes, servport05.pfd, servport06.pfd, servdeb.pfd,
servtp5.pfd, iris.pfd, servsol2.pfd, servsol3.pfd, servaix.pfd, servhp.pfd,
servwas1.dmz, mail.pythagore-fd.dmz, servext.dmz, sw0.pfd, switch2,
laserps.delphes, laserps.pfd
        }
```

Services

Utilisation des services (suite)

Exemple de définition complète de service :

```
define service{
        use                             generic
        host_name                       servsol2.pfd
        service_description             /dev/hdb2 Free Space
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  pythagore-admins
        notification_interval           120
        notification_period             24x7
        notification_options            w,u,c,r
        check_command                   check_local_disk!20%!10%!/dev/hdb2
        }
```

Services

serviceextinfo.cfg

Ce fichier permet d'ajouter des informations supplémentaires sur les services (icônes, fichier de commentaires...) pour l'interface web de contrôle et notamment pour la carte (statusmap).

Exemples :

```
define serviceextinfo{
        host_name               host9
        service_description     PING
        icon_image              ping.gif
        }
define serviceextinfo{
        name sei1
        host_name               host1,host2,host3,host4
        service_description     TCP Wrappers
        icon_image              wrappers.gif
        }
define serviceextinfo{
        use sei1
        service_description     Security Alerts
        icon_image              security.gif
        }
define serviceextinfo{
        hostgroup               novell-servers
        service_description     LRU Sitting Time
        icon_image              cache.gif
        }
```

timeperiods.cfg

Ce fichier contient les définitions des alias horaires pour la planification de tâches.

Exemples :

```
define timeperiod{
        timeperiod_name  24x7
        alias            24 Heures sur 24 et 7 jours sur 7
        sunday           00:00-24:00
        monday           00:00-24:00
        tuesday          00:00-24:00
        wednesday        00:00-24:00
        thursday         00:00-24:00
        friday           00:00-24:00
        saturday         00:00-24:00
        }

define timeperiod{
        timeperiod_name  workhours
        alias            Heures de travail (normal)
        monday           09:00-17:00
        tuesday          09:00-17:00
        wednesday        09:00-17:00
        thursday         09:00-17:00
        friday           09:00-17:00
        }
```

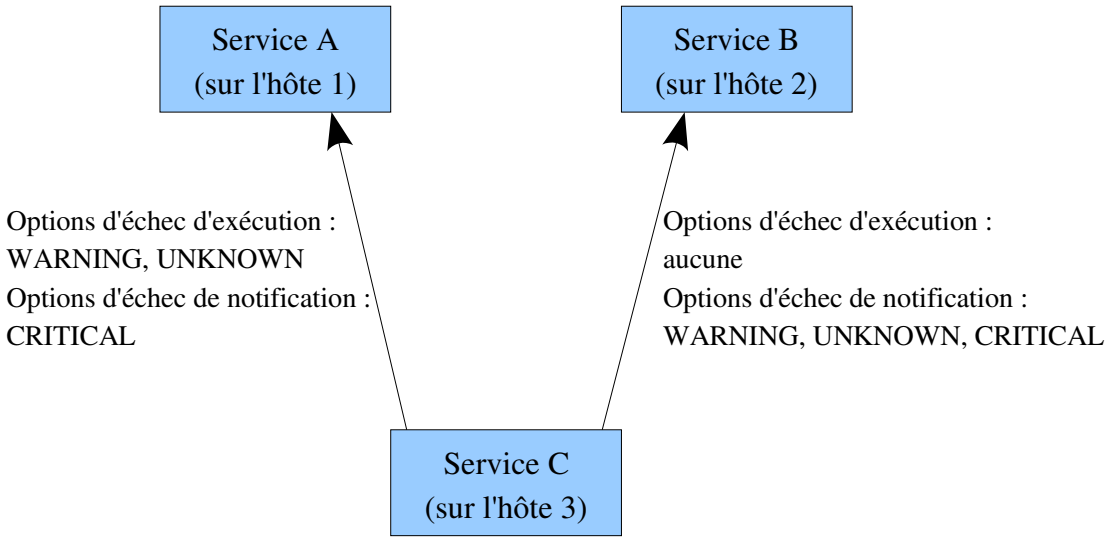
Dépendances entre hôtes/services

Il s'agit d'une fonctionnalité avancée de Nagios qui permet de contrôler le comportement d'hôtes et de services en fonction du status d'autres hôtes et services.

Dépendances entre services

Un service peut être associé à un ou plusieurs autres services, qui peuvent appartenir à des hôtes différents. Les dépendances entre services sont utilisées pour arrêter l'exécution et/ou les notifications d'un service si les services dont il dépend sont dans un état particulier (*ok*, *warning*, *unknown* ou *critical*).

Un exemple :



Dépendances entre hôtes/services

Dans l'exemple illustré par la figure précédente, si le service A est dans l'état « Warning » ou « Critical » alors l'exécution de la vérification du service C est reportée à plus tard.

Si le service A est dans l'état « Warning » ou que le service B est dans l'état « Warning », « Unknown » ou « Critical » alors les notifications pour le service C sont suspendues jusqu'au prochain test.

Les dépendances d'un service sont testées lors de chaque vérification du service ou envoi de notifications pour ce service, et si une des dépendances n'est pas satisfaite (les états des services concernés sont testés) la vérification ou notification ne s'effectue pas.

Dépendances entre hôtes

Les dépendances entre hôtes fonctionnent de la même manière qu'entre services à la différence près qu'elles ne fonctionnent que pour la suppression de notifications et pas pour la suppression des vérifications.

Fichier de configuration

La configuration des dépendances s'effectue dans un fichier de configuration d'objets, par exemple dans les fichiers services.cfg et hosts.cfg, ou dans un fichier séparé dependencies.cfg (le nom de ces fichiers de configuration doivent apparaîtrent dans nagios.cfg).
Chaque dépendance de service doit être définie par une section *servicedependency* et chaque dépendance d'hôte par une section *hostdependency*.

Dépendances entre hôtes/services

Exemple de configuration reprenant la figure précédente :

```
define servicedependency{
        host_name                       Hote1    ; Nom de l'hôte et du service
dont
        service_description             ServiceA ; le service dépend.
        dependent_host_name             Hote3    ; Nom du service dépendant et de
        dependent_service_description   ServiceC ; son hôte.
        execution_failure_criteria      w,u      ; Critères pour lesquels la
        notification_failure_criteria   c        ; vérification ou la
notification
}                                                ; seront annulés.

define servicedependency{
        host_name                       Hote2
        service_description             ServiceB
        dependent_host_name             Hote3
        dependent_service_description   ServiceC
        execution_failure_criteria      n                ; n = aucun critère
        notification_failure_criteria   w,u,c            ; warning, unknown,
critical
}
```

- A noter que les dépendances entre services ne s'héritent pas, à moins que la directive *inherits_parent* soit spécifiée et mise à 1. Les dépendances entre hôtes ne s'héritent jamais.

Dépendances entre hôtes/services

Exemple de dépendances entre hôtes :

```
define hostdependency{
        host_name                       Hote2
        dependent_host_name             Hote3
        execution_failure_criteria      d,u     ; d = down, u = unreachable
```

Dans cet exemple les notifications s'arrêteront pour l'hôte 3 si l'hôte 2 est arrêté (d, pour « *down* ») ou inaccessible (u, pour « *unreachable* »).

Critères d'échec d'exécution ou de notification:

- servicedependency : o = ok, w = warning, c = critical, u = unknown, n = none
- hostdependency : d = down, u = unreachable, r = recover

➢Par défaut, Nagios attend que ces critères soient certifiés en mode *hard*, c'est à dire après le nombre de tentatives spécifiées par *max_check_attempts*.

Dépendances de groupes

Ce qui est applicable aux hôtes et aux services l'est aussi aux groupes d'hôtes et aux groupes de services. Un groupe de service se défini par l'entité *servicegroup*. Un groupe d'hôte se définit par l'entité *hostgroup*.

Les dépendances entre groupes d'hôtes sont spécifiées dans des entités de type *hostgroupdependency* tandis que les dépendances entre groupes de services le sont dans des entités de type *servicegroupdependency*.

- Attention, les groupes de services ne sont pas supportés par les anciennes versions de Nagios (1.x).

XI

Autres fichiers de configuration

Ressources

resource.cfg

Ce fichier contient principalement des variables utilisateur ($USER1$ à $USER32$) et, en option pour Nagios version 1, la configuration de l'accès à une base de données des états des hôtes.

```
# $USER1$ : chemin des plugins     $USER1$=/usr/lib/nagios/plugins

# $USER2$ : chemin des event       #$USER2$=/usr/lib/nagios/plugins/eventhandlers
# handlers

# exemple d'utilisation des        #$USER3$=someuser
# variables pour stocker des       #$USER4$=somepassword
# mots de passes (masqués pour
# les CGIs)
```

Ressources

Base de donnée (Nagios 1.x)

Dans les anciennes versions de Nagios (1.x) les états des hôtes surveillés pouvaient optionnellement être gérés dans une base de donnée relationnelle (pgSQL ou Mysql). Ce n'est plus le cas depuis la version 2.

```
# DG EXTENDED DATA                #xeddb_host=somehost
                                  #xeddb_port=someport
                                  #xeddb_database=somedatabase
                                  #xeddb_username=someuser
                                  #xeddb_password=somepassword

# DB STATUS DATA (lecture seule   #xsddb_host=somehost
# pour les CGIs)                  #xsddb_port=someport
                                  #xsddb_database=somedatabase
                                  #xsddb_username=someuser
                                  #xsddb_password=somepassword

# DB COMMENT DATA (lecture seule  #xcddb_host=somehost
# pour les CGIs)                  #xcddb_port=someport
                                  #xcddb_database=somedatabase
                                  #xcddb_username=someuser
                                  #xcddb_password=somepassword

# DB DOWNTIME DATA (lecture       #xdddb_host=somehost
# seule pour les CGIs)            #xdddb_port=someport
                                  #xdddb_database=somedatabase
                                  #xdddb_username=someuser
                                  #xdddb_password=somepassword
```

Contacts

contactgroups.cfg

Ce fichier configure les groupes d'administrateurs à contacter pour les alertes (ces personnes sont définies dans le fichier contacts.cfg).

```
define contactgroup{
        contactgroup_name       exploitation-admins
        alias                   Administrateurs d'exploitation
        members                 nagios,jdoe
        }
define contactgroup{
        contactgroup_name       pythagore-admins
        alias                   Administrateurs Pythagore
        members                 nagios
        }
define contactgroup{
        contactgroup_name       switch-admins
        alias                   Etherswitch Administrators
        members                 nagios
        }
define contactgroup{
        contactgroup_name       imprimante-admins
        alias                   Printer Administrators
        members                 nagios
        }
```

Contacts

contacts.cfg

Personnes à contacter ainsi que leurs coordonnées.

```
define contact{
        contact_name                    nagios
        alias                           Nagios Admin
        service_notification_period     24x7
        host_notification_period        24x7
        service_notification_options    w,u,c,r
        host_notification_options       d,u,r
        service_notification_commands   notify-by-email,notify-by-epager
        host_notification_commands      host-notify-by-email,host-notify-by-
epager
        email                           nagios-admin@localhost.localdomain
        pager                           pagenagios-admin@localhost.localdomain
        }
```

Dans cet exemple, l'administrateur de Nagios est défini avec son adresse de couriel, quand il peut être contacté, et sur quels types d'évènements il devra être contacté.

Contacts

```
define contact{
        contact_name                    jdoe
        alias                           John Doe
        service_notification_period     workhours
        host_notification_period        workhours
        service_notification_options    c,r
        host_notification_options       d,r
        service_notification_commands   notify-by-email
        host_notification_commands      host-notify-by-email
        email                           jdoe@localhost.localdomain
        }
```

Dans cet exemple, la notification s'effectue par courriel, mais on peut utiliser tout autre type de notification pour peu que la machine possède les programmes adéquats, par exemple par messagerie instantanée IRC, par SMS, ou autre.

Les options de notifications *host_notification_options* et *service_notification_options* déterminent quelles raisons devront engendrer une notification.

- Pour les hôtes, les options de notification sont : d (Down), r (Recover) et u (Unreachable).
- Pour les services, les options de notification sont c (Critical), w (Warning), u (Unknown), o (Ok) et a (All).

Commandes

checkcommands.cfg

Ce fichier définit les commandes telles qu'elles seront utilisées par Nagios. Elles seront invoquées dans le fichiers services.cfg

Exemples simples :

```
define command{
        command_name     check_local_users
        command_line     $USER1$/check_users -w $ARG1$ -c $ARG2$
        }

define command{
        command_name     check-host-alive
        command_line     $USER1$/check_ping   -H   $HOSTADDRESS$   -w   3000.0,80%   -c
5000.0,100% -p 1
        }
```

command_name est le nom du test à effectuer. *command_line* est la commande réelle utilisant un greffon situé dans le répertoire des plugins.

➢Si *command_line* possède des arguments de type $ARG1$... alors les tests de services doivent appeler la commande avec des arguments de la sorte : *command_name!arg1!arg2*.

Commandes

Exécution de greffons à distance

Grâce à des greffons spéciaux qui ont la propriété d'exécuter d'autres greffons après s'être connectés sur une machine distante, il est possible de sonder l'état interne de ces machines. On peut utiliser pour cela les fonctionnalitées de NRPE, de telnet ou de ssh.

Exemple pour l'exécution distante avec NRPE :

```
#vérifie les utilisateurs distants à travers nrpe
define command{
        command_name     check_remote_users
        command_line     $USER1$/check_nrpe -H $HOSTADDRESS$ -c check_users
        }
```

Exemple d'un gestionnaire d'évenement :

```
#commande pour redemarrer un serveur d'impression
define command{
        command_name     restart-lp
        command_line     /usr/local/nagios/libexec/eventhandlers/restart-lp
$SERVICESTATE$ $STATETYPE$ $SERVICEATTEMPT$
        }
```

Commandes

misccommands.cfg

Ce fichier contient des commandes diverses, notamment pour les notifications d'alerte, ou la journalisation de données supplémentaires.

Exemples :

```
# envoi d'e-mail au sujet d'un service
define command{
        command_name     notify-by-email
        command_line     /usr/bin/printf "%b" "Alerte :
$NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHote: $HOSTALIAS$ ($HOSTADDRESS$)
\nEtat: $SERVICESTATE$\n\nDate: $DATETIME$\n\nInfos:\n $OUTPUT$" | /bin/mail -s
"[nagios]: alerte $NOTIFICATIONTYPE$" $CONTACTEMAIL$
        }

# broadcast TTYs au sujet d'un host (pas de destinataire particulier)
define command{
        command_name      host-notify-by-wall
        command_line      /usr/bin/printf "%b" "[nagios] alerte sur $HOSTNAME$
($HOSTADDRESS$) : Etat : $HOSTSTATE$\nMessage : $OUTPUT$\nDate : $DATETIME$" | wall
}

# recuperation d'informations quantitatives
define command{
        command_name     process-host-perfdata
        command_line     /usr/bin/printf "%b"
"$LASTCHECK$\t$HOSTNAME$\t$HOSTSTATE$\t$HOSTATTEMPT$\t$STATETYPE$\t$EXECUTIONTIME$\t
$OUTPUT$\t$PERFDATA$">>/usr/local/nagios/var/host-perfdata.out
        }
```

XII

Déploiement Nagios : les extensions

Introduction

Nagios peut se contenter de faire des requêtes distantes vers des clients, mais certains services ne sont accessibles que de *l'intérieur* (exemple: le nombre d'utilisateurs connectés).

Selon la configuration des machines du réseau, plusieurs possibilités sont à étudier :

Si chaque client offre un agent SNMP, nagios peut l'utiliser pour connaître l'état des ressources du client.

Si chaque client possède un service ssh, nagios peut exécuter les plugins du client via son plugin *check_by_ssh* et récupérer le résultat des tests.

Dans le cas où ni snmp ni ssh ne sont présents, on peut déployer sur les clients le serveur nrpe (Nagios Remote Plugin Execution) qui acceptera des demandes d'exécution distantes.

Une autre solution consiste à récupérer de manière passive les résultats d'exécution de plugins distants autonomes grâce au démon nsca (Nagios Service Check Acceptor).

Introduction

Principe de nrpe

- à l'aide du plugin check_nrpe, le démon nagios fait une requête vers un hôte à surveiller
- l'hôte reçoit la requête grâce à un démon nrpe
- nrpe exécute le plugin demandé puis renvoit sa réponse et son code retour
- le plugin check_nrpe de nagios récupère la réponse et l'analyse comme s'il s'agissait d'un plugin local

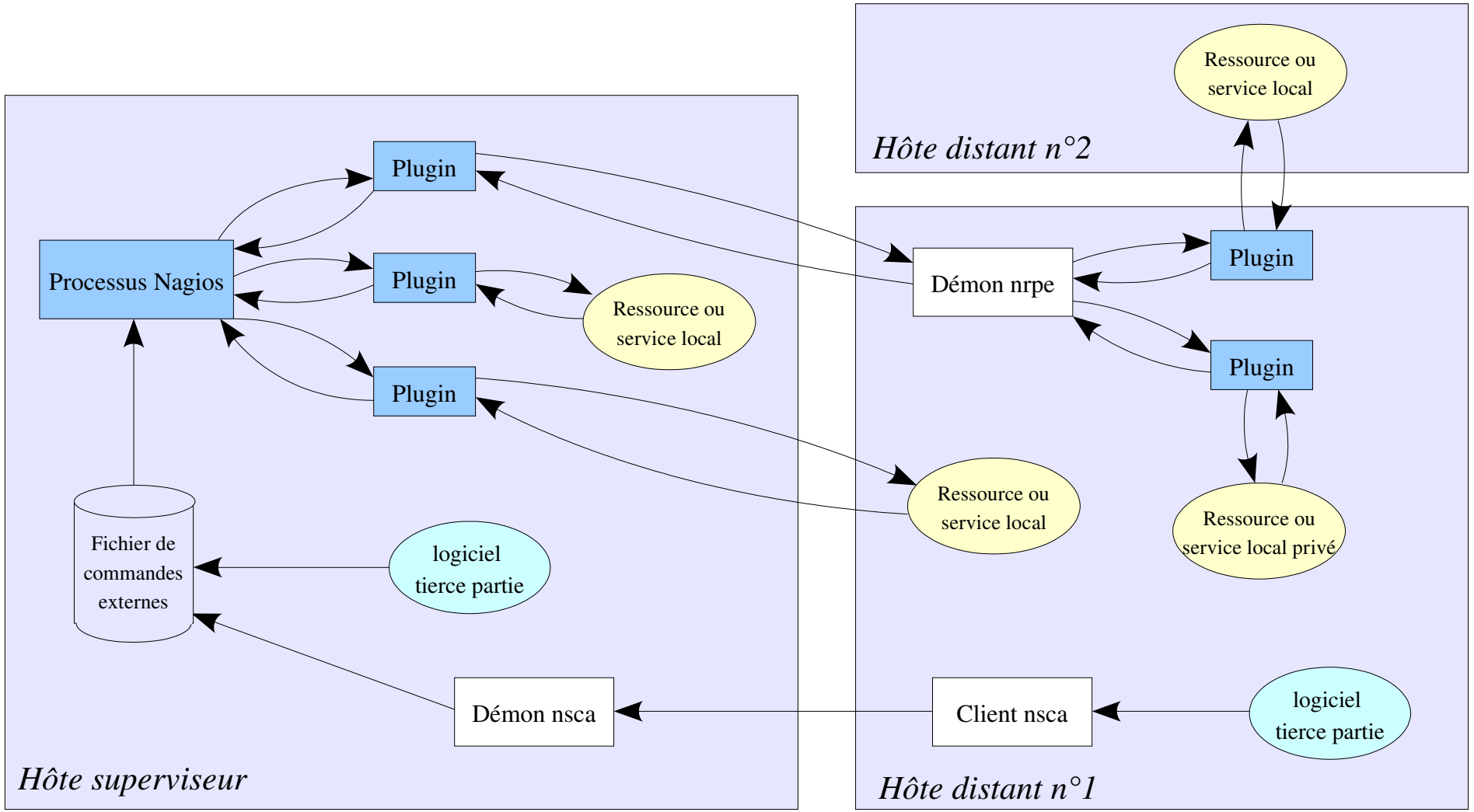
➢nrpe constitue une méthode de contrôle dite active, c'est la machine Nagios qui est initiatrice des tests

Principe de nsca

- l'hôte distant exécute un plugin en réponse à une alerte détectée par une commande externe
- le résultat et le code retour sont envoyés au serveur nagios via le plugin check_nsca
- nagios récupère le code grâce au démon nsca et le traite comme un résultat de *check actif*

➢ncsa est une méthode passive, ce n'est pas la machine Nagios qui est à l'origine des tests

Utilisation de Nagios avec nrpe et nsca



Utilisation de Nagios avec nrpe et nsca

Mode d'exécution

nrpe et nsca peuvent fonctionner en standalone ou bien à travers le daemon unix inetd (plus économique en ressources dans le cas d'un usage raisonable)

- nrpe n'est disponible que pour les hôtes Linux ou Solaris (utiliser snmp dans les autres cas).
- nsca supporte l'authentification des agents avant l'échange des données.

Pour le déploiement, nous prendrons l'exemple de l'extension nrpe.

Installation de nrpe

La démarche est similaire à l'installation de nagios.

La mise en place à partir de sources ne propose pas de routine d'installation, aussi faut-il copier les fichiers (configuration et exécutable) dans les emplacements *ad hoc*.

- Le support SSL est nécessaire car nrpe crypte ses connexions.

Installation des plugins

On peut se contenter d'installer seulement les plugins qui seront nécessaires sur l'hôte.

Configuration inetd

Par défaut, le démon nrpe écoute le port 5666.
Pour que inetd (ou xinetd) reconnaisse le service, il faut rajouter une entrée dans /etc/services :

```
nrpe    5666/tcp        # NRPE
```

Installation de nrpe

Le nom de l'entrée de /etc/services doit correspondre au service décrit dans le fichier de configuration d'inetd (ou xinetd). Un fichier d'exemple est fourni :

```
#/etc/xinetd.d/nrped
# default: on
# description: NRPE (Nagios Remote Plugin Executor)
service nrpe
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = nagios
        group           = nagios
        server          = /usr/local/nagios/bin/nrpe
        server_args     = -c /etc/nagios/nrpe.cfg --inetd
        log_on_failure  += USERID
        disable         = no
# adresse du serveur nagios
        only_from       = 192.176.21.3
}
```

Une fois l'installation terminée, on peut mettre à jour la configuration de xinetd en rechargeant ce service :

```
[root@servnagios /root]# /etc/init.d/xinetd reload
```

Configuration de nrpe

Elle réside dans le fichier /usr/local/nagios/etc/nrpe.cfg

➢Ce fichier doit être lisible par l'utilisateur unix « nagios »

Configuration du démon

Exemple de fichier nrpe.cfg :

```
# attention, les 5 premières directives ne sont utiles qu'en standalone :

# fichier dans lequel le demon nrpe écrit son pid     pid_file=/var/run/nrpe.pid
# port TCP du serveur (>1024)                         server_port=5666
# adresse du serveur                                  server_address=192.168.1.1
# hôtes autorisés à se connecter                      allowed_hosts=127.0.0.1
# propriétaire du processus                           nrpe_user=nagios
# groupe du processus                                 nrpe_group=nagios

# autoriser les arguments de commandes                dont_blame_nrpe=0
# debugage                                            debug=0
# timeout de commande                                 command_timeout=60
# fichiers/répertoires de configuration à inclure     #include=<mon_fichier>
                                                      #include_dir=<mon_repertoire>
                                                      #include_dir=<mon_autre_repertoire>
```

Configuration de nrpe

Configuration des commandes

Pour chaque commande qui sera appelée par le plugin *check_nrpe* de nagios, on trouvera une entrée dans le fichier nrpe.cfg. La syntaxe de configuration est semblable à l'ancienne syntaxe de NetSaint :

```
#Syntaxe : command[<command_name>]=<command_line>
```

#commandes sans arguments

```
command[check_users]=/usr/local/nagios/libexec/check_users -w 5 -c 10
command[check_load]=/usr/local/nagios/libexec/check_load -w 15,10,5 -c 30,25,20
command[check_total_procs]=/usr/local/nagios/libexec/check_procs -w 150 -c 200
```

 # commandes utilisant des arguments en ligne de commande (dont_blame_nrpe=1)

```
command[check_users]=/usr/local/nagios/libexec/check_users -w $ARG1$ -c $ARG2$
command[check_load]=/usr/local/nagios/libexec/check_load -w $ARG1$ -c $ARG2$
command[check_procs]=/usr/local/nagios/libexec/check_procs -w $ARG1$ -c $ARG2$ -s $ARG3$
```

Exemple, si nagios exécute la commande :

```
/usr/local/nagios/libexec/check_nrpe -H mon_hote -c  check_users
```

le démon nrpe de mon_hote lancera la commande :

```
/usr/local/nagios/libexec/check_users -w 5 -c 10
```

(voir l'exemple du service check_remote_users dans le chapitre configuration de nagios)

Exemple de déploiement

Nrpe devra être déployé sur un certain nombre de clients. Il serait fastidieux de répéter les mêmes opérations d'installation et de configuration sur chacun d'eux. Il est donc nécessaire d'automatiser cette tâche.

Création de scripts de déploiement

- On considère que les clients ont un démon ssh en activité, qui permet le dépôt et l'exécution sécurisée de scripts.

Création d'une archive (nrped.tar.gz) contenant le démon, elle devra contenir les fichiers suivants :

```
-rwx--x--x       root/root        1193     2003-04-23 11:07:25 /etc/init.d/nrped
-rw-------       root/root        448      2003-04-23 11:10:36 /etc/xinetd.d/nrped
-rw-------       nagios/nagios    5051     2003-04-23 11:08:11 /etc/nagios/nrpe.cfg
-rwxr-xr-x       root/root        138934   2003-04-23           11:08:28           /
usr/local/nagios/bin/nrpe
```

On encode cette archive en base alphanumérique pour l'inclure dans le script d'installation :

```
[/usr/local/nagios/sharedeploy]# uuencode nrped.tar.gz > nrped.uu
```

On procèdera de la même manière avec les plugins.

Exemple de déploiement

Script d'installation de nrpe et des plugins pour xinetd :

```
#!/bin/sh
#install_nrped

echo "création de l'utilisateur nagios"
groupadd -g 5666 nagios
useradd -u 5666 -g nagios nagios

echo "nrpe 5666/tcp #nrpe daemon" >> /etc/services

echo "extraction du daemon..."
(sed 's/^X//' << '_EOTAR_'
... archive de nrpe uuencodée ...
_EOTAR_
) | uudecode -o - | gzip -d -c | tar -P -xv
echo "fin de l'installation du daemon"

echo "extraction des plugins..."
(sed 's/^X//' << '_EOTAR_'
... archive des plugins uuencodée ...
_EOTAR_
) | uudecode -o - | gzip -d -c | tar -P -xv
echo "fin de l'installation des plugins"

echo "fin de d'installation"
/etc/init.d/xinetd reload

#EOF
```

Exemple de déploiement

Script de déploiement

```
#!/bin/sh
#deployer_nrpe

if [ $# -lt 1 ]; then
        echo usage : `basename $0` host [host...]
        exit 1
fi

install=/usr/local/nagios/share/deploy/install_nrped
install_sc=`basename $install`

for host in $*
do
        if ping -c2 $host 2>/dev/null 1>&2;   then
                echo déploiement sur $host
                scp $install root@$host:.
                ssh root@$host ./$install_sc
        else
                echo problème avec $host
        fi
done
#EOF
```

- On peut rajouter dans ce script des instructions pour patcher le fichier nrpe.cfg, et pour mettre à jour le champ only_from de /etc/xinetd.d/nrped

XIII

Configuration de l'ordonnanceur

Méthode d'ordonnancement

Pour chaque hôte ou service à surveiller, nous avons défini trois paramètres temporels du test :

- l'intervalle de contrôle *normal_check_interval*,
- l'intervalle de tentatives *retry_check_interval*
- la période effective du test *check_period*.

Nagios possède aussi une configuration concernant la méthode de répartition temporelle de tous les tests confondus. La configuration de l'ordonnanceur à ce sujet est dans le fichier *nagios.cfg*.

Nagios répartit automatiquement l'étalage des tests selon :

- La définition des trois paramètres temporels pour chaque service
- Le délai de lancement des tests indépendants : *inter_check_delay_method*
- Le facteur d'entrelacement des tests de service : *service_interleave_factor*
- Le nombre maximal de tests concurrents : *max_concurrent_checks*
- La fréquence de récupération des résultats de test : *service_reaper_frequency*

Nagios peut lui-même calculer certaines valeurs par défaut mais ne peut pas deviner la charge de la machine sur laquelle il repose.

Délai entre chaque test

Le délai entre chaque test indépendant est spécifié en secondes et centièmes de secondes ou bien avec des valeurs par défaut qui laisseront Nagios calculer une valeur intelligente. Il est défini par l'option *inter_check_delay_method* et généralement placé à « s ».

Paramètres de *inter_check_delay_method* :

- n : (none) aucun délai. Lancer tous les tests en même temps.
- d : (dumb) délai d'une seconde entre chaque service différent.
- s : (smart) calcul intelligent de répartition en fonction des *normal_check_interval*
- x.yz : délai de x.yz secondes entre chaque test indépendant.

➢Lancer tous les tests en même temps peut être problématique si la machine du superviseur ne peut supporter la charge.

Entrelacement des services

L'entrelacement des service est un facteur entier qui permet à Nagios de répartir le choix des services qui sont dans la file d'attente au lieu de les prendre les uns à la suite des autres, notamment lorsqu'il a la possibilité de les lancer de manière concurrente. Il est défini par l'option *service_interleave_factor* et généralement placé à « s ».

Paramètres de *service_interleave_factor* :

- s : (smart) calcul intelligent en fonction du rapport services / hôtes
- x : (1-n) choisir un test tous les x éléments de la file d'attente.

- La file d'attente des tests est classée selon le nom d'hôte auxquels ils sont associés. Donner à ce paramètre une valeur trop petite laissera donc Nagios tester la majorité des services d'un même hôte en même temps.
- Le calcul de la valeur « smart » est égal au plafond entier du rapport du nombre total de services sur le nombre total d'hôtes.

Nombre maximal de tests concurrents

Le nombre maximal de tests en concurrence permet de limiter le nombre de processus à lancer en parallèle et permettant ainsi de ne pas surcharger la machine de supervision. Il est défini par l'option *max_concurrent_checks* et généralement placé à « 0 ».

Paramètres de *max_concurrent_checks* :

- 0 : aucune limite
- 1 : pas de parallélisme
- n : n tests parallèles au plus.

➢Pour déterminer correctement le nombre maximal de tests en concurrence, il faut se baser sur le délai *inter_check_delay_method*, la valeur du *service_reaper_frequency* et sur le temps moyen d'exécution d'un test. On peut évaluer ce nombre de manière optimiste en ajoutant 1 à la partie entière du rapport du temps moyen d'exécution d'un test par la valeur de délai entre chaque test.

Fréquence de récupération

Lorsque Nagios est autorisé à lancer des tests en parrallèle, il n'attend pas la fin d'un test pour en lancer un autre. Les résultats des tests sont donc placés dans une file que Nagios devra lire afin de pouvoir lancer les notifications et les gestionnaires d'évènements associés aux tests.

Le paramètre *service_reaper_frequency* détermine la période (en secondes) de lecture de la file des résultats des tests. Sa valeur est généralement placée à « 10 ».

Il est inutile de tenter de lire la file des résultats plus souvent qu'il ne peut y avoir de réponses nouvelles des tests.

Il faut penser que le temps de latence d'une réponse à un test est très variable selon le type de médium (local, bus I2C, réseau), l'emplacement des hôtes et la complexité du service.

Il est normal de prendre pour valeur de *service_reaper_frequency* la valeur moyenne des latences associées aux réponses des tests.

- Pour des services réseaux, les valeurs des délais d'attente maximale (timeout) des fonctions *gethostbyname()* ou *connect()* des systèmes Unix nous donnent un ordre de grandeur allant de 5 secondes à 2 minutes environ. Dans ce cadre, Il sera donc inutile de placer *service_reaper_frequency* à plus de 120 secondes.

XIV

Contrôle et débogage

Fichier de logs

Lorsque l'on a créé une configuration de supervision, il est nécessaire de contrôler que les résultats des greffons de supervision font correctement leur travail et que Nagios ne logue aucune erreur.

Selon les instructions du fichier nagios.cfg, Nagios placera l'historique de ses actions et les erreurs dans des fichiers de log. Ils sont une source importante d'information sur d'éventuels problèmes.

Exemple de log nagios:

```
# tail -2 /var/log/nagios/nagios.log
[1147798253] Warning: Return code of 127 for check of service 'PING' on host
'poste403.pfd11' was out of bounds. Make sure the plugin you're trying to run
actually exists.
[1147798253] HOST NOTIFICATION: nagios-admin;poste403.pfd11;DOWN;host-notify-by-
email;(No output!)
```

Ici, le code d'erreur de 127 rendu par l'essai d'exécution de *check_ping* est anormal. En effet, ce plugin doit renvoyer un code entre 0 (OK) et 3 (UNKNOWN). Dans cet exemple, Le code 127 est retourné par le système car le plugin n'est pas à l'endroit spécifié dans la configuration (erreur de chemin).

Commandes

vérifier les déclarations de commandes

Il est possible que l'on ait déclaré des commandes erronées qui appellent de manière incorrecte les greffons de Nagios, ou bien que nos propres extensions ne fonctionnent pas correctement.

Pour vérifier qu'une commande fonctionne bien, on peut lancer le greffon manuellement avec exactement les mêmes arguments que l'aurait fait Nagios à partir de la configuration de la commande.

Exemple de commande de test utilisant le greffon check_http :

```
define command{
        command_name    check_url
        command_line    $USER1$/check_http -I $HOSTADDRESS$ -p $ARG1$ -u $ARG2$ -w
$ARG3$ -c $ARG4$
}
```

Une erreur classique est une faute minime dans un nom de variable de Nagios (aussi appelée macro).

Exécution

vérification manuelle

Le lancement manuel d'un greffon ou toute autre extension avec les paramètres définis dans la configuration de Nagios nous assure qu'il fonctionne correctement ou non.

La majorité des greffons de Nagios possèdent une aide concise si on les appelle sans option :

Exemple d'aide d'un greffon :

```
# /usr/lib/nagios/plugins/check_http
check_http: Could not parse arguments
Usage: check_http -H <vhost> | -I <IP-address> [-u <uri>] [-p <port>]
       [-w <warn time>] [-c <critical time>] [...]
```

Il est important de prendre l'identité de nagios pour se mettre dans son environnement.

Exemple de lancement manuel correct (valeur de retour entre 0 et 2) :

```
# su - nagios
$ cd plugins
$ ./check_http -I 10.23.102.5 p 80 -u /index.html -w 1 -c 2
HTTP OK HTTP/1.1 200 OK - 445 bytes in 0.078 seconds
$ echo $?
0
```

Il est indispensable de donner les droits d'exécution des greffons corrects pour l'utilisateur nagios.

Options détaillées

Options des greffons de Nagios

Lorsqu'on appelle un greffon avec l'option « -h », il affichera une aide détaillée.

Exemple d'aide détaillée d'un greffon :

```
# /usr/lib/nagios/plugins/check_ping -h
check_ping (nagios-plugins 1.4.2) 1.49
Copyright (c) 1999 Ethan Galstad <nagios@nagios.org>
[...]
Options:
 -h, --help
    Print detailed help screen
 -V, --version
    Print version information
 -4, --use-ipv4
    Use IPv4 connection
 -6, --use-ipv6
    Use IPv6 connection
-H, --hostname=HOST
   host to ping
-w, --warning=THRESHOLD
   warning threshold pair
-c, --critical=THRESHOLD
   critical threshold pair
-p, --packets=INTEGER
   number of ICMP ECHO packets to send (Default: 5)
[...]
```

Débogage

Certains greffons peuvent être écrits en langage interprété (shell, perl,...) et peuve être facilement tracables lors de cas d'erreurs inconues.

En particulier, lorsque l'on développe ses propre sondes ou autres extensions, il est intéressant de pouvoir tracer son exécution.

Trace d'un script shell

Pour tracer un script shell, on peut utiliser l'option « x » de l'interpréteur :

Exemple de trace d'un gestionnaire pour le système d'impression :

```
# sh -x /usr/lib/nagios/plugins/eventhandlers/reload_cupsys CRITICAL SOFT  3
[...]
+ case "$3" in
+ echo 'CUPS 3rd SOFT CRITICAL ISSUE : Reloading local cupsys daemon.'
+ CUPS 3rd SOFT CRITICAL ISSUE : Reloading local cupsys daemon.
+ /etc/init.d/cupsys reload
+ reload_cupsys: line 39: /etc/init.d/cupsys: Permission denied
+ ERR=126
+ exit 126
```

Dans cet exemple, le script *reload-cups* a récupéré le code d'erreur de cupsys en l'affectant à la variable *ERR* (ERR=$?) puis il a renvoyé *$ERR*.

XV

Développement de plugins

Normalisation

Il est possible de développer des plugins pour Nagios mais ils doivent obéir à certaines règles. Les éléments imposés par les spécifications sont :

une option standard

- -h ou --help pour afficher un mode d'emploi du plugin

des codes retour standards

• OK	0	(la ressource est en bonne santé)
• WARNING	1	(la ressource a un problème)
• CRITICAL	2	(la ressource est dans un état critique)
• UNKNOWN	3	(le plugin ne peut déterminer l'état)
• DEPENDENT	4	(l'état de la ressource est lié à une autre)

Le code DEPENDENT est particulier, optionnel et rarement géré par un plugin lui-même.

Moyens mis en oeuvre

Langages

Les plugins de nagios peuvent être développés en divers langages, ce qui facilite leur développement :

- langage C
- Perl
- shell Unix
- ...

Le choix du shell peut se justifier du fait que chaque système possède au moins un interpréteur de commandes de type *sh* (Bourne-shell), et qu'il s'agit du langage classique pour l'administration système.

Utilitaires

Pour le shell, le fichier *utils.sh* contient :

- les définitions des codes retour (sous forme de variables STATE_xx)
- des fonctions standard (print_revision, support)

Ce fichier devra être « sourcé » au début des plugins écrits en shell.
Il existe aussi une version Perl de cet utilitaire.

Exemple de plugin

Ce plugin vérifie l'état d'un service LPD sur un hôte (non CUPS).

```
#!/bin/sh
```

#on source le fichier d'utilitaires

```
. `dirname $0`/utils.sh

SONDE=`basename $0`

usage()
{
        print_revision
        cat <<_EOF_

$SONDE queue host
        affiche l'état de la file d'attente
$SONDE -h ou --help
        affiche cette aide
_EOF_
}
```

#duplication du canal d'erreur vers le canal standard (debug)

```
exec 2>&1
```

Exemple de plugin (suite)

#si moins de deux arguments : on affichera l'aide

```
if [ $# -lt 2 ]
then
        QUEUE=-h
else
        QUEUE=$1
        HOST=$2
fi
```

#analyse des arguments et statut

```
case $QUEUE in
-help|-h)
        usage
        ;;
*)
```

 # récupération du statut avec lpr, on substitue les LF par ";" (check_nrpe ne sait lire qu'une ligne)

```
        STATUS=$(lpq -P$QUEUE@$HOST | head -4 | tr '\n' ';')

        if echo $STATUS | grep -i error >/dev/null
        then
```

Exemple de plugin (suite)

```
#la chaîne contient "error" : warning
                $ECHO "$STATUS"
                exit $STATE_WARNING
        elif echo $STATUS | grep -i "cannot open connection" >/dev/null
        then

#connexion impossible : statut critique
                $ECHO "$STATUS"
                exit $STATE_CRITICAL
        elif echo $STATUS | grep -i Queue: >/dev/null
        then

#file d'attente présente : statut ok
                $ECHO "lpr OK : $STATUS"
                exit $STATE_OK
        else

#cas alternatif : statut inconnu
                $ECHO "statut inconnu : $STATUS"
                exit $STATE_UNKNOWN
        fi
        ;;
esac
```

XVI

gestionnaires d'évennement

Traîtement d'erreur

Un gestionnaire d'évènement (event-handler) est un programme optionel qui a pour but d'effectuer diveres opérations découlant d'une erreur, voire même de tenter résoudre le problème de manière automatique.

- Un gestionnaire d'évennement est une *action* associée à la réponse d'un test.
- Il sera toujours lancé, même si l'état reporté ne change pas (valable aussi pour "OK").
- Plusieurs variables (macros) définissant l'évènement lui seront transmises via une commande.

Un gestionnaire d'évennement sera notament utile pour la récupération sur erreur. Il s'agira dans ce cas, de développer un programme procédant à des opérations systématiques et dont les effets sont maîtrisés.

On peut lier un event-handler à une ressource ou à un modèle, auquel cas toutes les ressources concernées peuvent bénéficier du traîtement.

Nagios permet de lier un event-handler pour un type host ou un type service.

Macros de type évènement

Les variables décrivant le résultat d'un test de ressource permettent aux gestionnaires d'évènements de réagir en conséquence.

Voici les valeurs que peuvent prendre ces variables :

- $HOSTSTATE$ = « UP » ou « DOWN »
- $HOSTSTATETYPE$ = « SOFT » ou « HARD » (déterminé par *max_check_attempts*)
- $HOSTATTEMPT$ = un nombre entier positif (nombre d'essais)
- $SERVICESTATE$ = « WARNING », « CRITICAL », « UNKNOWN » ou « DEPENDENT »
- $SERVICESTATETYPE$ = « SOFT » ou « HARD » (déterminé par *max_check_attempts*)
- $SERVICEATTEMPT$ = un nombre entier positif (nombre d'essais)

S'il s'agit de réagir sur un hôte à distance, l'adresse de l'hôte est disponible dans $HOSTADDRESS$.

Exemple : relancement httpd

On considère qu'un serveur web (type apache httpd) est surveillé sur un host et devra être redémarré si la sonde renvoie 3 fois de suite une ereur CRTICAL de type SOFT (juste avant de passer en HARD et de déclancher une notification).

Dans la définition d'une ressource de type *service* ou de type *host*, l'utilisation d'un gestionnaire d'évènement se fait par l'option *event_handler*. Ici, il s'agit d'un service.

Exemple de lisison de l'event-hdanler dans un service appelé HTTP (syntaxe Nagios) :

```
define service {
        service_description       HTTP
        host_name                 WWW
        [...]
        event_handler_enabled     1
        event_handler             handle-httpd-error
        [...}
}
```

➢On ne peut pas transmettre d'argument personnel ($ARG1$,...) à un gestionnaire d'évènement.

Exemple : relancement httpd (suite)

Il est donc nécessaire de déclarer une commande que l'on utilsera à nos fins.

Exemple d'appel du gestionnaire d'évennement (synaxe Nagios) :

```
define command {
        command_name    handle-httpd-error
        command-line    $USER1$/reload_httpd.sh    $HOSTADDRESS$    $SERVICESTATE$
$SERVICESTATETYPE$ $SERVICEATTEMPT$
}
```

Le fichier *misccommands.cfg* est destiné à contenir une telle définition.

Mise en place

Dans cet exemple, le shell script *reload_httpdsh* à développer deva :

- n'agir que si les arguments transmis après le $HOSTADDRESS$ sont "CRITICAL" "SOFT" "3"
- se connecter alors sur 212.21.34.65 de manière transparente
- posséder sur place les droits pour relancer le service httpd
- relancer le service et se déconnecter

Il est possible de répondre au problème de connexion/exécution avec divers outils Unix :

Exemple de préparation des clés ssh pour connexion non interactive :

```
$ who am i
nagios
$ ssh-keygen -t dsa -P ""
$ scp .ssh/id_dsa.pub  nagios@212.21.34.65:.ssh/authorized_keys
```

Exemple de commande ssh avec commande à distance :

```
$ ssh -x 212.21.34.65 'sudo /etc/init.d/httpd restart'
```

Exemple de configuration /etc/sudoers sur le host 212.21.34.65 (edition avec "visudo") :

```
nagios ALL = NOPASSWD: /etc/init.d/httpd restart
```

Exemple de vérification manuelle du script après développement :

```
# su - nagios
$ ./handlers/reload_httpd.sh 212.21.34.65  CRITICAL SOFT 3
ACK: probleme CRITICAL SOFT 3 sur 212.21.34.65 : tentative de redemarrage.
```

XVII

Superviseurs redondants

Méthodes de redondance

La machine de supervision n'est pas exempte de panne. Dans un environnement critique, il est indispensable de la surveiller depuis une autre machine qui prendra le rôle de superviseur de secours lorsqu'elle détectera un état critique de la première.

Méthodes

Il existe plusieurs méthodes pour rendre le contrôle redondant et permettre à un hôte Nagios de prendre le relai de la supervision si un autre tombe en panne.
Il peut, en effet, y avoir deux superviseurs en action dont un seul fera des notifications, ou bien un superviseur endormi qui sera réveillé en fail-over si le premier tombe. Il existe aussi pour chaque méthode plusieurs mises en œuvre possibles.

Redondance pure

La méthode la plus simple est de dupliquer les contrôles sur deux machines de supervision mais de n'activer les notifications par défaut que sur un seul hôte « maître » qui sera en outre surveillé par le second Nagios. L'avantage de cette méthode est :

- une relative simplicité de mise en œuvre (recopie)
- une duplication de l'interface de contrôle (web)
- un suivi de l'état des hôtes d'un superviseur à l'autre.

L'inconvénient majeur de cette méthode est la duplication des tests sur le réseau, ce qui va augmenter le trafic et doubler les requêtes de service sur les hôtes contrôlés.

Mode failover

On peut ne démarrer le superviseur de secours qu'en cas de panne du premier.

Il faudra, dans ce cas, utiliser l'ordonnanceur standard unix *cron* pour ne surveiller que l'état du superviseur « maître » avec un script lancé via « telnet » ou « ssh » ou via une utilisation autonome du greffon *check_nrpe*.
L'avantage de cette méthode est qu'il n'y a pas de duplication des tests de l'état des hôtes et des services.
L'inconvénient majeur est que le superviseur de secours ne connaît pas l'état initial des hôtes et services.

Si l'on utilise de manière autonome (en ligne de commande) le greffon *check_nrpe*, il faudra installer un service NRPE sur l'hôte « maître » configuré avec une commande qui donnera l'état demandé (utiliser par exemple *check_nagios*).

Mise en place

Nous allons décrir la mise en place de deux superviseurs ayant presque les mêmes configurations, à la différence, que seul l'un des deux enverra les notifications tandis que l'autre n'activera cette option que si le premier chute.

Superviseur Maître :

Le serveur maître n'a aucune connaissance du superviseur secondaire. Il est configuré comme s'il était le seul à tout contrôler. Aucune modification n'est à prévoir sur l'hôte maître.

Superviseur secondaire :

Le superviseur secondaire doit :

- avoir exactement la même configuration des hôtes et services à surveiller
- surveiller en plus, le serveur maître
- n'envoyer aucune notification par défaut
- activer l'envoi de toutes les notifications si le maître ne peut plus le faire
- désactiver l'envoi de toute notification si le maître reprend ses fonctions

Configuration du secondaire

Nous allons uniquement montrer les particularités à ajouter au superviseur secondaire.

nagios.cfg

Il faut désactiver toute notification par défaut, mais accepter la commande externe qui permettra de changer cela en cas de besoin.

Partie intéressante du fichier nagios.cfg :

```
enable_notification=0
check_external_commands=1
```

hosts.cfg

Il faut déclarer l'hôte du superviseur maître à surveiller ainsi qu'un gestionnaire d'évennement qui récupèrera le code d'erreur du test.

Partie concernée du fichier hosts.cfg :

```
define host {
        use                     generic-host
        host_name               master
        alias                   Superviseur Maitre
        address                 212.24.6.14      ; adresse IP du maitre
        check_command           check-host-alive
        max_check_attempts      2
        event_handler           handle-remote-nagios-host-event
}
```

Configuration du secondaire

services.cfg

Il faut déclarer le service de supervision de l'hôte maître ainsi qu'un gestionnaire d'évènement qui récupèrera le code d'erreur.

Exemple de service de test d'un daemon Nagios distant :

```
define service {
        use                     generic-service
        host_name               master
        service_description     NAGIOS
        check_command           check-remote-nagios
        max_check_attempts      2
        event_handler           handle-remote-nagios-service-event
}
```

misccommands.cfg

Il faut définir la manière d'utiliser notre gestionnaire d'évènement si panne de l'hôte « master ».

Exemple de commande pour handle-remote-nagios-host-event :

```
define command{
        command_name    handle-remote-nagios-host-event
        command_line    /
usr/lib/nagios/plugins/eventhandlers/manage_master_host_state           $HOSTSTATE$
$HOSTSTATETYPE$
        }
```

Configuration du secondaire

misccommands.cfg (suite)

Il faut définir la manière d'utiliser notre gestionnaire d'évènement sur le service *check-remote-nagios* qui est effectué pour l'hôte « master ».

Exemple d'utilisation du gestionnaire d'évennement :

```
define command{
        command_name    handle-remote-nagios-service-event
        command_line    /
usr/lib/nagios/plugins/eventhandlers/manage_master_service_state      $SERVICESTATE$
$SERVICESTATETYPE$
        }
```

Maintenant, il reste à écrire les programmes de gestion d'évènement spécifiques à notre problématique.

Leur but est de lancer la commande externe *enable_notifications* (script fourni avec les plugins de nagios).

Le script devra lancer la commande *disable_notifications* lorsqu'il aura détecté le retour à l'état normal du superviseur maître.

Gestionnaire : panne du superviseur

Nous avons déclaré deux gestionnaires : un pour la panne matérielle de la machine maître, et un autre pour la panne du daemon nagios sur le maître. Le premier script concerne la gestion de la panne de la machine.

manage_master_host_state

```
#!/bin/sh
# On recuper la gestion des notifications seulement si le maitre est DOWN + HARD
# args: HOSTSTATE puis HOSTSTATETYPE (note: on testera d'abord le type)
ERR=0
case "$2" in
HARD)
        case "$1" in
        DOWN)   # Nous recuperons la gestion des notifications.
                /usr/lib/nagios/plugins/eventhandlers/enable_notifications; ERR=$?
                ;;
        UP)     # Nous ne prenons plus en charge les notifications.
                /usr/lib/nagios/plugins/eventhandlers/disable_notifications;
ERR=$?
                ;;
        *)      # Inconnu. On ne fait rien.
                ;;
        esac
        ;;
SOFT)   # On attend confirmation de la panne (HARD)
        ;;
esac
exit $ERR
```

Gestionnaire : panne du service nagios

Si la machine maître fonctionne mais que le service nagios est hors fonction, il faut récupérer l'erreur.

manage_master_service_state

```
#!/bin/sh
# Recuperer la gestion des notifications seulement si le service est CRITICAL +
HARD
# args: SERVICESTATE et SERVICESTATETYPE (nte: on testera d'abord le type)
ERR=0
case "$2" in
HARD)
        case "$1" in
        CRITICAL)       # On prend en charge les notifications.
                        /
usr/lib/nagios/plugins/eventhandlers/enable_notifications; ERR=$?
                        ;;
        OK)             # On abandonne la gestion des notifications.
                        /
usr/lib/nagios/plugins/eventhandlers/disable_notifications; ERR=$?
                        ;;
        *)
                # On ne fait rien si WARNING, UNKNOWN ou autre.
                ;;
        esac
        ;;
SOFT)   # On attend confirmation de la panne du service.
        ;;
*)      # Inconnu. On ne fait rien.
        ;;
esac
```

```
exit $ERR
```

greffon de test du maître

Greffons check_nagios et check_by_ssh

Le greffon standard *check_nagios* permet de connaître l'état du superviseur Nagios sur la machine locale. Pour l'utiliser sur une machine distante, on pourra le lancer via le greffon *check_by_ssh* qui l'exécutera après avoir établi une connexion « ssh » sur la machine à surveiller.

Génération de clés ssh

Afin de lancer le test via ssh, le greffon doit pouvoir s'authentifier auprès de l'hôte maître sans que l'on ait à taper un mot de passe de connexion. Le service ssh permet de s'authentifier au moyen d'échange de clés uniques. Il est indispensable de générer le trousseau de clé sous utilisateur « nagios ».

Exemple de placement d'une clé publique

```
# su - nagios
# ssh-keygen -q -f .ssh/id_dsa -t dsa -P ""
# scp .ssh/id_dsa.pub 212.24.6.14:.ssh/authorized_keys
nagios@212.24.6.134's password: ********
```

Une fois la clé publique copiée, la prochaine connexion depuis cet hôte et vers l'hôte du maître se fera sans mot de passe tout en restant authentifiée.

Commande du test

Le greffon check_by_ssh n'est en lui-même incapable de tester quoi que ce soit. C'est un greffon qui demande en argument un autre greffon à exécuter via ssh et qui transmettra correctement son resultat.

checkcommands.cfg

Nous devons définir exactement la commande *check-remote-nagios* utilisée pour le service NAGIOS déclaré plus haut. Cette commande utilise les greffons *check_by_ssh* (en local) et *check_nagios* (à distance).

Exemple de test nagios distant:

```
define command {
        command_name    check-remote-nagios
        command_line    $USER1$/check_by_ssh       -H       $HOSTADDRESS$       -C
'/usr/lib/nagios/plugins/check_nagios  -F  /var/log/nagios/status.dat  -e  1  -C  /
usr/bin/nagios -d /etc/nagios/nagios.cfg'
        }
```

Fruity

Installation

Fruity permet, depuis un navigateur web, d'importer une configuration Nagios quelconque, de la modifier puis de l'exporter à nouveau. Fruity nécessite un serveur http, une base de donnée mysql et php pour fonctionner.

Téléchargement

Fruity est un logiciel libre. On peut le télécharger à cette adresse : http://fruity.sf.net
Il n'existe pas encore de paquet RPM mais son installation manuelle n'est pas complexe.

Installation

Un serveur apache avec php5 devra être installé. Il faudra aussi que les modules php-pear et php-mysql soient pré-installés (c'est souvent le cas). Enfin, une base de données Mysql devra être en fonction.

Procédure d'installation typique :

```
# cd /var/www/html
# tar zxf fruity-xxx.tar.gz
# mv fruity-xxx fruity
# chown -R apache:apache fruity
# ln -s /usr/share/nagios/images/logos fruity/logos
```

Dans cet exemple, nous considérons que /var/www/html est la racine des documents du serveur web.

Configuration

Fruity nécessite d'avoir une base mysql. On va donc créer une base appelée 'fruity' et accorder tous les droits pour l'utilisateur 'fruity' (par exemple).

Création de base mysql pour fruity :

```
# mysql -u root
> create database fruity;
> grant all on fruity.* to fruity@localhost identified by 'fruity';
> quit;
```

Ensuite, nous créons les tables de fruity grâce au script SQL fourni avec le logiciel.

Création des tables mysql pour fruity :

```
# mysql -u fruity -passowrd=fruity fruity < /var/www/html/fruity/sqldata/fruity-
mysql.sql
```

Il ne reste plus qu'à utiliser un navigateur web pour aller à l'adresse http://localhost/fruity/

Pour permettre à fruity de modifier la configuration de Nagios, il faudra autoriser apache à créer des fichiers dans le répertoire de configuration de Nagios.

Donner l'autorité à apache :

```
# chown -R apache /etc/nagios
```

Utilisation

Tout d'abord, il faudra importer la configuration de Nagios existante.

Cliquer sur « import » et entrer le chemin du fichier « nagios.cfg ».

La configuration de Nagios s'effectue avec le menu thématique qui est en haut de la page. Il reprend tous les types connus de Nagios.

La configuration ou création d'un service se fait en choisissant tout d'abord un hôte. Une fois le service créé, on peut l'associer à d'autres hôtes en les configurant un par un.

Une fois la configuration terminée, on l'exporte : cliquer sur « export » puis YES.
Des fichiers de sauvegarde seront aussi créés avec l'extension « .backup ».

➢ Attention, une configuration incomplète d'un hôte ou service ne sera pas vérifiée à l'export et Nagios refusera de redémarrer dans ce cas.

Pour que Nagios prenne en compte sa nouvelle configuration, on lui demande de la recharger.

Vérification d'une configuration Nagios :

```
# nagios -v /etc/nagios/nagios.cfg
```

Rechargement :

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

Cacti - Introduction

Cacti

Présentation

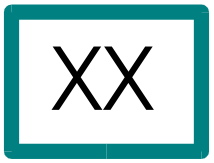
Cacti est un outil de mesure des ressources réseau basé sur RRDTool (Round Robin Database) et utilisant SNMP. Cacti n'interprète pas les résultats obtenus et n'est donc pas destiné à alerter quiconque en temps réel à propos de problèmes éventuels.

Cacti est écrit en PHP et nécessite un serveur web pour être utilisé. Il stocke toutes ses informations dans une base MySQL. Sa sonde SNMP standard se lance via l'odronnanceur Cron. Cacti est développé et distribué sous licence GPL.

Fonctionnalités

En plus du stockage d'informations et de la création de graphiques, voici les fonctionnalités de Cacti :

- Utilise différentes sources d'informations (scripts, fichiers, etc)
- Utilise des templates
- Organise les graphes de différentes manières (dont une sous forme d'arbre)
- Gestion des utilisateurs et des permissions (vues, création de graphe, etc...)



Installation

Prérequis

Cacti est disponible sur systèmes Windows, GNU/Linux, OpenBSD, NetBSD et FreeBSD. Les éléments suivants doivent être installés :

- PHP
- Apache
- MySQL
- RRDTool
- net-snmp

Nous allons détailler l'installation de RRDTool, net-snmp et Cacti sur Linux.

RRDTool

Vous pouvez effectuer l'installation depuis un paquet ou depuis les sources. Pour l'installation depuis les sources, on récupère une archive TAR :

```
$ tar xzf rrdtool-<version>.tar.gz
$ cd rrdtool-<version>
$ ./configure --prefix=/usr
$ make
```

puis, en root :

```
# make install
```

Prérequis

Outils SNMP

Le paquet se nomme « net-snmp », et l'installation depuis les sources se déroule de la manière suivante :

```
$ tar xzf net-snmp-<version>.tar.gz
$ cd net-snmp-<version>
$ ./configure -prefix=/usr
```

Les informations suivantes doivent alors être renseignées :

- Default SNMP Version : laisser la valeur par défaut (3)
- System Contact Information : l'adresse mail de l'administrateur
- System Location : lieu où se situe le système
- Logfile Location : laisser par défaut /var/log/snmpd.log
- snmpd persistent storage location : laisser par défaut /var/net-snmp

Continuer avec la commande :

```
$ make
```

puis, en root :

```
# make install
```

Installation

Paquet

L'installation depuis les paquets (« urpmi cacti », « apt-get install cacti »...) installe dans */var/www* les fichiers qui vont être utilisés par Apache et dans */usr/share/cacti* les autres fichiers.

Un fichier est ajouté à la configuration d'Apache : */etc/httpd/conf.d/cacti.conf* (en général).

Ceci permet l'accès à Cacti via l'url « http://<adresse_serveur_web>/cacti ».

Les versions packagées créent un utilisateur système *cacti*. Il faudra s'assurer que cet utilisateur aura le droit d'écrire dans le répertoire */var/www/cacti/rra*.

Installation

Sources

Pour installer Cacti depuis les sources, téléchargez la dernière version de Cacti puis décompressez l'archive dans le répertoire racine d'Apache (en général /var/www ou /var/www/html) :

```
# useradd -g apache -d /var/www/cacti -m cacti
# tar xzf cacti-<version>.tar.gz
# cp cacti-<version>/* /var/www/cacti && chown -R apache:apache /var/www/cacti
```

Ajouter dans le fichier de configuration d'Apache httpd.conf les lignes suivantes :

```
Alias /cacti /var/www/cacti
<Directory /var/www/cacti>
    Allow from all
</Directory>
```

Puis recharger la configuration d'Apache :

```
service httpd reload
```

Installation

Configuration de la base de données

Il est nécessaire de créer une base de données nommée « cacti » :

```
$ mysql -u root
> create database cacti;
> quit
```

Importer ensuite dans la base cacti le contenu de cacti.sql, fichier présent dans le répertoire /usr/share/cacti/cacti.sql ou /var/www/cacti selon la méthode d'installation :

```
$ mysql cacti < cacti.sql
```

Cacti s'identifie à MySQL sous le nom « cactiuser » (valeur dans /var/www/cact/include/config.php).
Il faut donc lui donner toutes les permissions pour la base de données « cacti » :

```
$ mysql -u root
> grant all on cacti.* to cactiuser@localhost identified by 'motdepasse_cacti';
> quit
```

Modifiez le fichier de configuration de Cacti /var/www/cacti/include/config.php (ou /etc/cacti.conf) pour mettre le mot de passe que vous avez attribué à l'utilisateur « cactiuser » :

```
$database_password = "motdepasse_cacti";
```

Installation

Configuration des droits

Lors d'une installation à partir des sources, attribuer les répertoire *rra/* et *log/* à l'utilisateur aproprié :

```
$ cd /var/www/cacti
$ chown -R apache:apache log/ rra/ #logs générés par les scripts php (user apache)
$ chown -R cacti rra/              #bases rrd générées par poller.php (user cacti)
```

Ajouter (ou vérifier) la ligne suivante au fichier /etc/crontab pour lancer la sonde toutes les 5 minutes :

```
*/5 * * * * cacti php /var/www/cacti/poller.php > /dev/null 2>&1
```

Interface web

Accéder à l'url « http://localhost/cacti » à l'aide d'un navigateur.

Si l'installation s'est bien déroulée, il restera quelques étapes à faire via l'interface web. Ensuite se connecter avec l'identifiant « admin » et le mot de passe « admin » pour accéder à la page d'accueil de Cacti. Immédiatement, Cacti vous demandera de donner un nouveau mot de passe pour l'utilisateur 'admin'.

Utilisation de Cacti

Page d'accueil

La page d'accueil de Cacti est normalement accessible par l'url « http://<adresse_serveur_web>/cacti ».

Deux onglets sont disponibles dans la partie supérieure de l'interface de Cacti : l'onglet « console » permet d'accéder au menu comportant toutes les actions réalisables avec Cacti, et l'onglet « graphs » permet de visualiser les graphiques qui ont été configurés.

Les graphiques

Quasiment tout ce qu'on trouve dans Cacti est lié à un graphique. Il est possible à tout moment d'afficher la liste de tous les graphiques disponibles en cliquant sur « Graph Management » dans la partie « Management » du menu.

Les graphiques de Cacti sont générés par rrdtool. L'abcisse d'un graphique représente toujours une plage de temps. Cacti montre plusieurs graphes par ressource: horaire, journée, mois et année. l'ordonnée d'un graphe est automatiquement mise à la bonne échelle en fonction des valeurs collectées.

Les graphiques de Cacti sont très proches de ceux qu'on peut obtenir manuellement avec RRDTool, Cacti permettant de créer facilement ces graphiques sans que l'utilisateur ait besoin de connaître parfaitement le fonctionnement de RRDTool.

Les hôtes

Avant toute chose, il est nécessaire d'indiquer à Cacti les hôtes à superviser. Le plus simple est que ces hôtes soient accessibles par SNMP. Dans Cacti, un hôte est appelé « device ».

Lister les hôtes

Cliquer sur « Devices » dans la partie « Management » du menu permet d'accéder à la liste des hôtes avec leur état.

Il est possible de filtrer cette liste à l'aide de la barre située en haut de page :

- « Type » pour n'afficher par exemple que les postes Windows,
- « Status » pour ne garder que les hôtes qui sont dans un état particulier,
- « Search » pour effectuer une recherche d'après la descrition et le nom d'hôte.

Actions sur les hôtes

Les actions possibles se trouvent dans la liste déroulante située en bas à droite. En cliquant sur le bouton « Go » à côté de la liste, l'action sera effectuée pour tous les hôtes que l'on a cochés.

Pour créer un nouvel hôte à surveiller, cliquer sur « Add » en haut à droite de la page.

Création d'un hôte

Seuls les champs « Description » et « Hostname » (qui doit être le nom d'hôte ou l'adresse IP) doivent obligatoirement être remplis pour créer un nouvel hôte.

Choisissez le « Host Template » qui correspond à l'hôte à ajouter, en sachant que le modèle « Generic SNMP-enabled Host » peut être choisi dans la plupart des cas, ou « none » si SNMP n'est pas utilisé.

Options snmp

- « SNMP Community » : communauté SNMP, pour la lecture.
- « SNMP Version » : version du protocole SNMP, utiliser la version 1 (sauf cas particulier)
- « SNMP Port » : port UDP du serveur SNMP
- « SNMP Timeout » : temps d'attente maximum en millisecondes d'une réponse SNMP

Après avoir confirmé la création de l'hôte, diverses statistiques sont affichées en haut de page (résumé de l'OID *.1.3.6.1.2.1* de l'hôte SNMP), sinon le message « SNMP error » indique un problème SNMP entre Cacti et l'hôte.
Il faudra configurer les agents SNMP pour autoriser la lecture de leurs informations par Cacti. La configuration se fait différament selon que l'on travaille avec SNMP v1, v2c ou v3.

Configuration basique d'un agent SNMP v1

Cacti devra être autorisé à lire le maximum d'informations sur les ressources des hôtes surveillés.
Voici un exemple de configuration du daemon *ucd-snmp* dans le fichier /etc/snmp/smpd.conf :

```
# snmpd.conf pour l'agent ucd-snmp [voir la page man snmpd.conf(5)]
syslocation      Labo (Salle 3)
syscontact       Root (root@machine.lan)
# 1. communaute / contexte de securite
com2sec  cacti    192.168.0.42     cactus
com2sec  normal   192.168.0.0/24   public
com2sec  anonyme  default          public
# 2. politique / groupe (remplacer v1 par v2c pour utiliser le protocole v2c)
#        name     model    class
group    intrus   v1       anonyme
group    invite   v1       normal
group    special  v1       cacti
# 3. vues accessibles
#        name             inc/excl         subtree
view     vuesysteme       included         .1.3.6.1.2.1.1
view     vuesysteme       included         .1.3.6.1.2.1.25.1.1
view     vuetotale        included         .1
# 4. contrôle d'acces
#        group    context  model    level    prefix   read             write    notif
access   intrus   ""       any      noauth   exact    none             none     none
access   invite   ""       any      noauth   exact    vuesysteme       none     none
access   special  ""       any      noauth   exact    vuetotale        none     none
# 5. support des cartes bcm5820
pass     .1.3.6.1.4.1.4413.4.1    /usr/bin/ucd5820stat
```

Table des matières