Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences
Western Switzerland

Introduction aux plugins Nagios

Novembre 2010

LESCOURT Adrien

Table des matières

1 Introduction

Nagios est un logiciel de surveillance de systèmes et de réseaux. Il permet par exemple de monitorer toute une liste d'ordinateurs et d'avoir pour chacune d'entre elle une vue de son état. Cette vue s'effectue via une interface Web et permet donc une vision d'ensemble des systèmes monitorés. Le serveur Nagios possède donc une large panoplie de services, telle que la surveillance de disques durs, de mémoires, de services réseaux...

Le travail de mon camarade M. Rossier explique en détail les principes ainsi que le fonctionnement d'un serveur Nagios. Pourtant Nagios permet d'aller plus loin en proposant un système de Plugins. Grace à cette fonctionnalité, l'utilisateur peut alors étendre les services proposés par défaut dans Nagios en y ajoutant des fonctions de surveillance personnalisées. Ce document à pour but de permettre à son lecteur de comprendre le principe du fonctionnement de base d'un plugin. Il pourra alors à son terme rapidement mettre en oeuvre son propre plugin. Cependant, les notions abordées ici sont le fruit d'une étude de plusieurs documentations extérieures et ne sont en aucun cas à considérer comme références.

Nous aborderons tout d'abord les mécanismes d'un plugin Nagios avant de détailler sa configuration. Nous expliquerons ensuite les moyens de communications pouvant être mis en oeuvre entre le serveur et hôtes surveillé, avec l'exemple d'un serveurs ESXi.

2 Mécanismes

2.1 Présentation

Contrairement à de nombreux autres logiciels de monitoring et de supervision, Nagios ne possède pas de système interne pour la surveillance, mais des programmes externes appelés Plugins. Grace à cette architecture, il est possible de contrôler tout ce que nous désirons.

Le mécanisme de base du fonctionnement d'un Plugin est le suivant :

1. Nagios exécute un plugin défini pour une machine donnée
2. Le plugin procède à la surveillance qui lui incombe (par exemple l'espace mémoire encore disponible)
3. Nagios récupère le ou les résultats, les interprète et agit en conséquence (ne fait rien, alerte, envoie d'un courrier électronique...)

Ces plugins sont donc bien séparés du coeur de Nagios, mais interagissent avec lui. Ils peuvent donc être des scripts, ou même des programmes exécutables implémentés dans la plupart des langages de programmation (C/C++, Java, Python, Bash...). Nous verrons que pour Nagios, quoi que fasse le plugin, seul le résultat est important.

2.2 Règles

Pour pouvoir communiquer avec le serveur Nagios, il faut respecter un certain nombre de règles et de contraintes.

2.2.1 Messages Nagios

Tout d'abord, une des premières fonctionnalité de Nagios est d'afficher un message indiquant l'état d'un service qui s'exécute sur une machine distante. La communication entre le Plugin et Nagios se fait via la sortie standard. Le message doit faire moins de 80 caractères et Nagios n'affichera que la première ligne de ce message. Il est donc nécessaire d'avoir un message concis et résumé, qui indique la nature ou la raison du problème.

Si toutefois l'utilisateur veut afficher plus d'information, il peut spécifier le flag verbose -v 2 qui va indiquer que le message est composé de plusieurs lignes.

2.2.2 Code de retour

Le serveur Nagios divise l'état des hôtes qu'il surveille en quatre catégories : OK, Warning, Critical, Unknown. Pour qu'un plugin indique au serveur l'état de la machine qu'il surveille, il doit spécifier le

code de retour de programme de la manière suivante :

- `0 - Status OK` : Le plugin a vérifié le service et il semble que tout fonctionne bien.
- `1 - Status Warning` : Le plugin a vérifié le service mais certains seuils "warning" ont été dépassés.
- `2 - Status Critical` : Le plugin n'a pas pu vérifier le service, ou alors un seuil "critical" à été dépassé.
- `3 - Status Unknown` : Problème dans la configuration ou les arguments du plugin. Le plugin n'aura alors exécuté aucune action.

Nous aborderons plus loin comment définir les seuils de chaque plugin.

2.3 Exemple

A partir des règles précédentes, nous pouvons écrire tous les plugins que nous voulons. Nous pouvons par exemple établir un squelette de plugin en Python.

```
#!/usr/bin/python
import sys
args = sys.argv[1:] #Obtention des arguments

if isCritcal():
    print "CRITICAL - " + afficheRaison()
    sys.exit(2)
if isWarning():
    print "WARNING - " + afficheRaison()
    sys.exit(1)

print "OK - Tout va bien"
sys.exit(0)
```

Pour que ce programme soit exécutable, il ne faut pas oublier de lui donner les droits nécessaires :

```
chmod +x monfichier.py
```

3 Configuration

Maintenant que nous avons établi les principes fondamentaux pour établir un plugin, nous allons voir comment configurer le serveur Nagios pour qu'il l'utilise. Tout au long de ce chapitre, nous nous baserons sur le serveur Nagios du laboratoire établi sur un Ubuntu server ;

3.1 L'arborescence

Nous ne détaillerons pas ici tous les fichiers nécessaire à la configuration de Nagios, mais uniquement ceux que la configuration d'un plugin implique.

Tout d'abord les plugins doivent être sauvegardés dans le répertoire `/usr/lib/nagios/plugins`. Un serveur Nagios bien configuré doit avoir défini une variable `$USER1$` dans le fichier de configuration `/etc/nagios3/ressources.cfg`

Nous avons donc un plugin de test appelé `testPlugin.py` dans `/usr/lib/nagios/plugins` qui simule une surveillance quelconque.

Ensuite, nous utiliserons uniquement deux fichiers de configuration :

- `/etc/nagios3/commands.cfg` qui contiendra la définition du plugin
- `/etc/nagios3/conf.d/labotd_test_machine.cfg` qui correspond au fichier de configuration d'une machine de test.

3.2 Définition du plugin

3.2.1 Command

La définition du plugin dans Nagios est très simple. Il suffit d'ajouter une `command` dans le fichier `/etc/nagios3/commands.cfg`

```
define command{
    command_name    testPlugin
    command_line    usr/lib/nagios/plugins/testPlugin.py
}
```

Cependant, ici le path vers le dossier de plugins est défini directement. Puisque nous avons une variable `$USER1$` correspondant à ce path, nous pouvons l'utiliser comme ceci :

```
define command{
    command_name    testPlugin
    command_line    $USER1$/testPlugin.py $ARG1$ $ARG2$ $ARG3$
}
```

Notons que nous avons aussi ajouté trois champs : *ARG*1 *ARG*2 *ARG*3. Ils correspondent aux arguments que peut prendre le plugin.

3.2.2 Service

Maintenant que la commande est définie, il faut créer un `service`. Il est important de bien comprendre la différence entre la `command` et le `service`. En effet, un plugin peut prendre plusieurs arguments en paramètres. Ces arguments vont dans la plupart des cas définir les seuils ou les intervalles de valeurs dans lequel notre machine doit s'exécuter.

Imaginons par exemple un parc d'ordinateurs comprenant 30 machines. Celles ci sont divisées en deux catégories : des machines anciennes et des machines récentes. Si l'on veut que le serveur Nagios monitore l'espace disque disponible, il faut que le plugin s'adapte aux hôtes qu'il inspecte. Si les machines anciennes ont des disques de 30Go et les nouvelle de 500Go, nous allons définir par exemple que le seuil minimum critique est inférieur à 3Go pour les disques de 30Go et inférieur à 20Go pour ceux de 500Go.

Quand on crée une `command`, on définit uniquement un script ou un exécutable avec son nombre d'arguments. Un service correspond alors à une configuration de ce script/exécutable avec les valeurs désirées en argument et la machine cible. Si l'on reprend notre exemple précédent, nous établirons alors une `command` et plusieurs `services` : la moitié avec comme arguments une valeur de 3Go, et l'autre moitié avec une valeur de 20Go.

Pour l'exemple utilisé au laboratoire, nous allons créer un service dans le fichier `/etc/nagios3/conf.d/labotd_test_machine.cfg`

```
define service{
    use                     generic-service
    host_name               testmachine
    service_description     plugin de test
    check_command           testPlugin
}
```

- `use` correspond au template que nous voulons utiliser. `generic-service` est le point de départ de la plupart des plugins.
- `host_name` correspond à la machine sur laquelle nous voulons exécuter le plugin.
- `service_description` est un bref descriptif du service
- `check_command` est le nom de la `command` que nous avons préalablement créé.

Ici nous avons défini un service sans arguments. La syntaxe pour en ajouter est la suivante :

```
define service{
    use                     generic-service
    host_name               testmachine
    service_description     plugin de test
    check_command           testPlugin!argument_1!argument_2
}
```

Nagios propose toute une liste d'options qui peuvent être ajoutées au `service` que nous ne détaillerons pas dans ce document. Ces paramètres sont plus propre au fonctionnement de Nagios qu'à la réalisation d'un plugin.

3.2.3 Définition du host

Nous n'entrerons pas dans les détails de la configuration des hosts, mais il semble important d'indiquer que la `testmachine` a été préalablement définie dans le même fichier de configuration `labotd_test_machine.cfg` de la manière suivante :

```
define host{
    use                     generic-host
    host_name               testmachine
    alias                   machine de test
    display_name            machine de test
    address                 Ip_du_host
    parents                 nagios
}
```

Une fois que les fichiers de configuration ont été établis, il est nécessaire de redémarrer le serveur Nagios pour que les modifications soient prisent en compte.

```
/etc/init.d/nagios3 restart
```

4 Remote connection

Lorsque l'on définit un plugin, Nagios ne s'occupe pas de transférer le programme vers la machine cible. Le serveur exécute lui même les instructions du plugin. Cela implique que pour obtenir les informations provenant du hôte désiré, il faut que le plugin implémente un mécanisme de communication.

4.1 SSH

Une des premières solution disponible consiste à écrire un plugin en Bash qui se connecte via `ssh`. Il faudra tout d'abord que le plugin prenne comme argument l'adresse ip de la machine distante. Puis deux solutions sont possible pour effectuer la connexion `ssh`. Soit on passe aussi le username et le password en argument du plugin, soit on génère une paire de clés privé/public entre le serveur Nagios et la machine monitorée. Bien évidemment, la première solution est la moins sécurisée puisqu'il faudra utiliser le mot de passe en clair dans la configuration du service.

Nous allons donc générer la paire de clés depuis le serveur Nagios.

```
ssh-keygen  <enter> <enter> <enter>
ssh-copy-id root@ip_machine_destination
```

Une fois que le script s'est connecté, les commandes seront exécutés sur l'hôte désiré.

Cependant un problème subsiste : la sortie standard ainsi que les codes de retour du programme seront sur le poste distant. Nagios ne sera alors plus informé de ces retours. Pour palier à ce problème, il faudra rediriger ces deux flux sur le tunnel `ssh`.

Il existe un plugin qui réalise ce traitement : `check_by_ssh`. En réalité, ce plugin un peu particulier prend en paramètre l'adresse de la machine distante, ainsi qu'un autre plugin Nagios. Il exécutera donc le script qu'il à reçu en paramètre sur la machine distante, et retransmet les informations au serveur Nagios. Pour définir la `command`, il faut procéder alors de la manière suivante :

```
define command {
        command_name    check_ssh_testPlugin
        command_line    $USER1$/check_by_ssh -H $HOSTADDRESS$ -C "$USER1$/testPlugi
}
```

Grâce à cela, nous avons créé un plugin nommé `check_ssh_testPlugin`, qui exécutera les fonctionnalités de `testPlugin` sur le host distant.

Un lien vers les détails de l'utilisation de ce plugin se trouve dans les références.

4.2 API de communication

L'autre méthode la plus répandue pour monitorer les machines distante consiste pour le programme à utiliser ses propres mécanismes de communication. Dans ces cas, la machine à surveiller fonctionnera comme un serveur que le plugin Nagios ira interroger. Bien évidement, il existe une multitude de ces mécanismes de communications. Pour rester cohérent avec le reste du cour de virtualisation, nous allons aborder l'API de VMWare pour communiquer avec un serveur ESXi.

5 Nagios et ESXi

5.1 Via ESX CLI

A l'aide du plugin `check_by_ssh`, il est possible d'avoir un script Bash qui fonctionne sur un serveur ESXi. Une fois sur ce serveur, nous pourrons utiliser toutes les commandes mises à disposition par VMWare pour monitorer le serveur. Voici une liste de ce qu'il sera possible de surveiller avec cette méthode, ainsi que quelques commandes intéressantes :

- La configuration du serveut ESXi
  - `esxcfg-info` : information sur le serveur
  - `vmware -l` : surveiller si la version est à jour
- La configuration des services actifs
  - `service -status-all` : Lister le status de tous les services
- La configuration du réseau et du firewall
  - `esxcfg-vswitch -l` : Lister les VSwich
- La configuration du stockage
  - `esxcfg-volume -l` : Lister les volumes snapshot
  - `vmkping` : Tester le VMKernel
- La configuration des VMs
  - `vmware-cmd -l` : Lister les VMs enregsitré
  - `vmware-cmd /vmfs/volumes/datastore/NAME/NAME.vmx getstate` : Connaitre l'état d'une VM

5.2 Via VMWare VI

L'API VMWare VI est une API qui utilise le VI SDK de VMWare. Elle permet entre autre de faciliter les communications entre un poste et un serveur ESXi, via une haute couche d'abstraction.

Voici un exemple de programme qui se connecte à ESXi, et retourne les informations suivantes :

- WARNING si aucune VM ne tourne sur le serveur
- WARNING si une des VM ne prend pas en charge le `multiple snapshot`
- OK si tout va bien

```
import java.net.URL;
import com.vmware.vim25.*;
```

```
import com.vmware.vim25.mo.*;

public class TestESXiNagois {

    public static void main(String[] args) throws Exception {

        // Connexion a ESXi
        ServiceInstance si = new ServiceInstance(new URL("https://ipESXI/sdk"),
                             "userName", " Password", true);

        // Recuperation des machines virtuelles
        Folder rootFolder = si.getRootFolder();
        ManagedEntity[] mes = new InventoryNavigator(rootFolder).
                             searchManagedEntities(" VirtualMachine");

        //Si aucune VM ne tourne, informe Nagios avec Warning
        if(mes==null || mes.length ==0) {
            System.out.println("WARNING - Aucune VM");
            System.Exit(1);
        }

        // Parcours des VMs
        for(int i=0; i<mes.length;i++) {
            VirtualMachine vm = (VirtualMachine) mes[i];
            VirtualMachineCapability vmc = vm.getCapability();
            vm.getResourcePool();

            //Si une VM ne supporte pas le Multiple Snapshot
            //Retourn un Warning et le nom de la VM en question
            if (!vmc.isMultipleSnapshotsSupported())
            {
                System.out.println("WARNING - Muliple Snapshot not supported
                                    by " + vm.getName());
                System.Exit(1);
            }
        }
        si.getServerConnection().logout();

        //Si tout va bien
        System.out.println("All VMs OK");
        System.Exit(0);
    }
}
```

Enfin, pour faire fonctionner ce plugin, il faudra lui ajouter les jar `dom4j-1.6.1.jar` et `vijava2u120091204.jar`.

```
javac -classpath dom4j-1.6.1.jar;vijava2u120091204.jar TestESXiNagois.java
```

Host	Service	Status	Last Check	Duration	Attempt	Status Information
clavister	Test DHCP	OK	2010-11-11 11:10:00	19d 13h 48m 7s	1/2	OK: ReÃ§u 2 DHCPOFFER(s), 1 de 1 serveurs ont rÃ©pondus, bail maxim
dns1_unige	Test DNS www.google.com	OK	2010-11-11 11:12:15	0d 0h 42m 3s	1/4	DNS OK: 0.005 secondes de temps de rÃ©ponse. www.google.com renv 209.85.135.103,209.85.135.104,209.85.135.105,209.85.135.106,209.85.1
dns2_unige	Test DNS www.google.com	OK	2010-11-11 11:13:31	0d 0h 55m 47s	1/4	DNS OK: 0.005 secondes de temps de rÃ©ponse. www.google.com renv 74.125.43.103,74.125.43.104,74.125.43.105,74.125.43.106,74.125.43.147
dns3_unige	Test DNS www.google.com	OK	2010-11-11 11:12:47	0d 0h 1m 31s	1/4	DNS OK: 0.005 secondes de temps de rÃ©ponse. www.google.com renv 74.125.39.103,74.125.39.104,74.125.39.105,74.125.39.106,74.125.39.147
dns4_unige	Test DNS www.google.com	CRITICAL	2010-11-11 11:11:03	112d 0h 34m 48s	4/4	Le domaine www.google.com n'a pas Ã©tÃ© trouvÃ© par le serveur
dns_DMZ	Test DNS labotd	OK	2010-11-11 11:11:18	26d 19h 49m 14s	1/2	DNS OK: 0.004 secondes de temps de rÃ©ponse. www.tdeig.ch renvoie
dns_Jenny	Test DNS www.google.com	OK	2010-11-11 11:13:00	9d 8h 41m 18s	1/4	DNS OK: 0.004 secondes de temps de rÃ©ponse. www.google.com renv 209.85.135.103,209.85.135.104,209.85.135.105,209.85.135.106,209.85.1
esxiCedric	test BashEsxiPlugin	CRITICAL	2010-11-11 11:14:00	19d 22h 42m 13s	1/4	(null)
fileserver1	Espace disque	OK	2010-11-11 11:11:34	143d 1h 57m 48s	1/4	DISK OK
fileserver2	Espace disque	OK	2010-11-11 11:11:50	143d 1h 57m 18s	1/4	DISK OK
ftp_Jenny	Test FTP	OK	2010-11-11 11:12:06	26d 19h 44m 36s	1/4	FTP OK - 0.001 second response time on port 21 [220 Microsoft FTP Serv
nagios	Current Users	OK	2010-11-11 11:12:22	143d 1h 56m 48s	1/4	UTILISATEURS OK - 1 utilisateurs actuellement connectÃ©s sur
	Espace disque	OK	2010-11-11 11:12:37	33d 19h 40m 10s	1/4	DISK OK
	Test la synchronisation NTP	OK	2010-11-11 11:11:00	8d 22h 33m 18s	1/4	NTP OK: DÃ©calage -6.048113573e-05 secs
pki_DMZ	Test HTTP	OK	2010-11-11 11:12:53	33d 21h 44m 22s	1/4	HTTP OK: HTTP/1.1 200 OK - 1439 bytes in 0.002 second response time
printer_HP4600	Test Status	OK	2010-11-11 11:10:00	0d 20h 49m 18s	1/4	Imprimante ok - ("Pr.t")
ssl_DMZ	Test HTTPS	OK	2010-11-11 11:13:09	33d 20h 55m 49s	1/4	HTTP OK: HTTP/1.1 200 OK - 736 bytes in 0.031 second response time
testmachine	test AdriPlug	WARNING	2010-11-11 11:13:25	33d 22h 51m 46s	1/4	WARNING - TEST
web_DMZ	Test HTTP	OK	2010-11-11 11:13:41	33d 21h 46m 13s	1/2	HTTP OK: HTTP/1.1 200 OK - 19942 bytes in 0.005 second response time

FIGURE 5.1 – Exemple de 2 plugins : TestAdriPlugin et testBashEsxiPlugin

6 Références

http://nagiosplug.sourceforge.net/developer-guidelines.html
http://www.redwaratah.com/wiki/index.php?title=Anatomy_of_a_Nagios_check_written_in_Python
http://wiki.monitoring-fr.org/
http://nagiosplugins.org/man/check_by_ssh
http://www.virtual-node.net/2010/07/23/utiliser-la-cli-dans-esx/