Activité : Enrichir la supervision avec les packs

Présentation

Ouvrez les différents fichiers, regardez leur contenu pour comprendre leurs relations et effectuer les actions permettant de superviser **localhost** en tant que **hôte Linux**.

Pour l'instant la supervision se contente de tester la présence de notre hôte avec un **ping**. Mais il est possible d'obtenir beaucoup plus d'informations sur les hôtes supervisés.

La communauté des développeurs de Shinken met à disposition un certain nombre de solutions pour superviser une machine Linux.

```
root@ctShinken:~# su - shinken
shinken@cdShinken:~$ shinken search linux
glances (david-guenault) [pack,system,linux,glances] : Standard check through checkglances.py and
glances server
linux-snmp (naparuba) [pack,linux,snmp] : Linux checks based on SNMP
linux-ssh (naparuba) [pack,linux,ssh] : Linux checks based on SSH without any script on distant server
pack-glances (david-guenault) [pack,system,linux,glances] : Standard check through checkglances.py and
glances server
raspberrypi (frescha) [pack,linux,raspberrypi,server,os] : Standard checks
varnish-ssh (kokosny) [pack,linux,varnish,ssh] : varnish checks based on ssh
```

Dans un premier temps, vous allez mettre en oeuvre **linux-snmp**.

Pour cela il faut télécharger le **pack** nécessaire :

- contenu du dossier **$ETC/packs**.

```
shinken@cdShinken:~$ ls /etc/shinken/packs/
readme.cfg
```

- contenu du fichier **readme.cfg** :

```
#In this place you will find all your packs downloaded from shinken.iowebsite.
#
#you can freely adapt them to your own need
```

Le dossier est vide et prêt à recevoir les packs.

Installation du pack linux-snmp

```
shinken@cdShinken:~$ shinken install linux-snmp
Grabbing :linux-snmp
OK linux-snmp
```

Un dossier est créé dans le répertoire **/etc/shinken/packs/**:

```
shinken@cdShinken:~$ ls /etc/shinken/packs/
linux-snmp readme.cfg
shinken@cdShinken:~$ ls /etc/shinken/packs/linux-snmp/
commands.cfg discovery.cfg linux-snmp.pack  services    templates.cfg
```

Qu'est ce qu'un pack ?

Un **pack** est un ensemble de fichiers de configuration et/ou de commandes développés par la communauté des utilisateurs et permettant la **supervision** des hôtes correspondant à des modèles génériques (Linux par exemple).

La lecture du fichier **commands.cfg** montre les commandes qui seront appliquées sur les hôtes de type Linux.

```
# ------------------------------------------------------------------
#
#      Linux standard check
#
```

```
# -------------------------------------------------------------
#
define command {
command_name check_linux_load
command_line  $PLUGINSDIR$/check_snmp_load.pl -H $HOSTADDRESS$ -C $_HOSTSNMPCOMMUNITY$ -f -w
$_HOSTLOAD_WARN$ -c $_HOSTLOAD_CRIT$ -T netsl -o $_HOSTSNMP_MSG_MAX_SIZE$
}

define command {
command_name check_linux_disks
command_line  $PLUGINSDIR$/check_snmp_storage.pl -H $HOSTADDRESS$ -C $_HOSTSNMPCOMMUNITY$ -m
$_HOSTSTORAGE_PATH$ -f -w $_HOSTSTORAGE_WARN$ -c $_HOSTSTORAGE_CRIT$ -S0,1 -o $_HOSTSNMP_MSG_MAX_SIZE$
}

define command {
command_name check_linux_cpu
command_line  $PLUGINSDIR$/check_snmp_load.pl -H $HOSTADDRESS$ -C $_HOSTSNMPCOMMUNITY$ -f -w
$_HOSTCPU_WARN$ -c $_HOSTCPU_CRIT$ -o $_HOSTSNMP_MSG_MAX_SIZE$
}

# Added -g flag since all linux system used are 64bits.
define command {
command_name check_linux_network_usage
command_line  $PLUGINSDIR$/check_netint.pl -H $HOSTADDRESS$ -C $_HOSTSNMPCOMMUNITY$ -n
"$_HOSTNET_IFACES$" -g -2c -f -e -w $_HOSTNET_WARN$ -c $_HOSTNET_CRIT$ -q -k -y -M -B -m -P
"$SERVICEPERFDATA$" -T "$LASTSERVICECHECK$"  -o $_HOSTSNMP_MSG_MAX_SIZE$
}

define command {
command_name check_linux_memory
command_line  $PLUGINSDIR$/check_snmp_mem.pl -w $_HOSTMEMORY_WARN$ -c $_HOSTMEMORY_CRIT$ -- -v 2c -c
$_HOSTSNMPCOMMUNITY$ $HOSTADDRESS$
}

define command {
command_name check_linux_logfiles
command_line  $PLUGINSDIR$/check_logfiles -f $_HOSTCHKLOG_CONF$
}

define command {
command_name check_linux_time
command_line  $NAGIOSPLUGINSDIR$/check_ntp_time -H $HOSTADDRESS$ -w $_HOSTNTP_WARN$ -c $_HOSTNTP_CRIT$
}
```

Ces commandes sont situées dans le répertoire **$PLUGINSDIR$** ou **$NAGIOSPLUGINSDIR$**, soit comme défini dans le fichier **$ETC/resource.d/path.cfg**

```
# Nagios legacy macros
$USER1$=$NAGIOSPLUGINSDIR$
$NAGIOSPLUGINSDIR$=/usr/lib/nagios/plugins

#-- Location of the plugins for Shinken
$PLUGINSDIR$=/var/lib/shinken/libexec
```

Le **modèle linux hérite** lui-même du modèle **generic*host, comme on le constate en ouvrant le fichier templates.cfg <code> # The LINUX template. define host { name linux-snmp use generic-host checkcommand checkping register 0 # We will show the linux custom view custom_views +linux SNMPCOMMUNITY $SNMPCOMMUNITYREAD$ _SNMPMSGMAXSIZE 65535 LOADWARN 2,2,2 LOADCRIT 3,3,3 STORAGEWARN 90 STORAGECRIT 95 CPUWARN 80 CPUCRIT 90 MEMORYWARN 80 MEMORYCRIT 95 NTPWARN 0.128 NTPCRIT 1 NETIFACES eth\d+|em\d+ NETWARN 90,90,0,0,0,0 NETCRIT 0,0,0,0,0,0 CHKLOGCONF $PLUGINSDIR$/logFileslinux.conf STORAGE_PATH / } define service { name linux-service use generic-service register 0 aggregation system } </code> Il utilise donc des commandes définies au niveau le plus haut mais il définit aussi ses propres commandes spécifiques dans son fichier commands.cfg. Voici de quelle manière les commandes du fichier commands.cfg sont appliquées au modèle Linux : * Une commande n'est pas appelée directement par l'hôte (à part le ping), * les commandes sont appelées via les services car ce sont des services qui tournent sur les OS. Le modèle Linux définit un service linux-service qui fait appel lui-même à un service générique generic-service. Exemple de définition d'un service spécifique cpu.fcg : <code> define service { servicedescriptionCpu use 20minlong,linux-service register 0 hostname linux-snmp checkcommand checklinuxcpu DETAILLEDESC Detect abnormal CPU usage _IMPACT Slow down applications hosted by the system _FIXACTIONS If recurrent situation then make performance audit } </code> Deux informations sont à identifier : * l'appel au modèle linux-service par la directive use, * l'établissement de la relation entre le service et le modèle d'hôte linux-snmp par la directive hostname. </code> Il reste maintenant à établir une relation entre notre hôte réel localhost et***

notre modèle linux-snmp. C'est dans le fichier $ETC/hosts/localhost.cfg que cette relation est établie : <code> define host { use linux-snmp,generic-host contactgroups admins hostname localhost address localhost } </code> Relancez le service shinken puis visualiser le résultat dans l'interface webui.

De nouveaux indicateurs sont présents mais cela ne fonctionne pas car **snmp** n'est probablement **pas configuré** sur l'hôte. ===== Installation et configuration de SNMP ===== * installation du paquet snmpd : <code shell> root@ctShinken:~# apt-get install snmpd </code>

> **Attention** il faut que dans le fichier **/etc/snmp/snmpd.conf** on écoute bien sur **localhost** car la variable $HOSTADDRESS$ prendra la valeur **localhost** qui sera traduite par 127.0.0.1 via /etc/hosts.

Il faut donc vérifier ou modifier le fichier de configuration pour que le serveur SNMP écoute sur le port 161 sur son adresse IP.

> Pour vérifier si le port est à l'écoute, 2 solutions sont possibles :
>
> - soit avec la commande **netstat –ulnp**,
> - soit avec la commande **nmap –sU localhost** si **nmap** est installé.

Pour cela, éditez le fichier **/etc/snmp/snmpd.conf** et modifiez la valeur du paramètre **agentAddress** avec la valeur **localhost** de votre serveur : <code> #agentAddress udp:127.0.0.1:161 agentAddress udp:localhost:161 </code> Il faut aussi modifier dans ce fichier les conditions d'accès à la MIB : * l'accès en lecture est défini par la directive **rocommunity. * par défaut, cet accès est restreint à une seule vue systemonly via l'option -V**. La vue **systemonly** est définie ainsi : <code> viewsystemonly included .1.3.6.1.2.1.1 viewsystemonly included .1.3.6.1.2.1.25.1 </code> L'accès par défaut en lecture, faisant référence à la vue **systemonly, est défini ainsi : <code> rocommunity public default -V systemonly </code> Cela restreint l'accès aux deux branches .1.3.6.1.2.1.1 (system) et .1.3.6.1.2.1.25.1 (hrSystem**) et uniquement celles-ci. Il est possible de modifier ces restrictions en créant par exemple une vue **all** qui sera utilisée ensuite à la place de **systemonly. Cela donne les modifications suivantes dans le fichier /etc/snmp/snmpd.conf : <code> view all included .1 rocommunity public default -V all </code> Toute la branche iso sera ainsi accessible en lecture. Il est nécessaire de redémarrer le service snmpd pour que le changement soit pris en compte. <code> root@ctShinken:~# service snmpd restart </code> Visualisez** le résultat dans webui :

Il y a davantage de retour d'informations. La configuration n'est pas terminée. ===== Test des commandes du modèle linux-snmp ===== Le test manuel des commandes présentes dans le modèle (template**) linux va permettre de voir celles qui ne fonctionnent pas correctement. ==== Rappel sur le test d'une commande : ==== Il faut : * regarder quel est l'exécutable lancé par la commande, * se positionner sur le dossier * et lancer la commande. ==== Pour la commande la commande check*linux*cpu : ==== Visualiser cette commande dans le fichier /etc/shinken/packs/linux-snmp/commands.cfg : <code> define command { command*name*check*linux**cpu command*line $PLUGINSDIR$/check*snmp*load.pl -H $HOSTADDRESS$ -C $*HOSTSNMPCOMMUNITY$ -f -w $*HOSTCPU*WARN$ -c $HOSTCPU_CRIT$ } </code> Il faut remplacer toutes les variables par leurs valeurs. Ces valeurs se Où trouver les valeurs ? On va trouver les valeurs critiques $*HOSTCPU*CRIT$ et warning $*HOSTCPU*WARN$ du cpu dans le fichier /etc/shinken/packs/linux-snmptemplates.cfg. <code> # The LINUX template. ... # We will show the linux custom view custom_views +linux _SNMPCOMMUNITY $SNMPCOMMUNITYREAD$ *LOAD*WARN 2,2,2 *LOAD*CRIT 3,3,3 *STORAGE*WARN 90 *STORAGE*CRIT 95 *CPU*WARN 80 *CPU*CRIT 90 *MEMORY*WARN 90,20 *MEMORY*CRIT 95,50 *NET*WARN 90,90,0,0,0,0 *NET*CRIT 0,0,0,0,0,0 } </code> Ainsi on peut taper la commande : <code shell> shinken@ctShinken:~$ /var/lib/shinken/libexec/check*snmp*load.pl -H localhost -C public -f -w 80 -c 90 Can't locate utils.pm in @INC (@INC contains: /var/lib/shinken/libexec /etc/perl /usr/local/lib/perl/5.14.2 /usr/local/share/perl/5.14.2 /usr/lib/perl5 /usr/share/perl5 /usr/lib/perl/5.14 /usr/share/perl/5.14 /usr/local/lib/site*perl .) at /var/lib/shinken/libexec/check*snmp*load.pl line 22. BEGIN failed–compilation aborted at /var/lib/shinken/libexec/check*snmp_load.pl line 22. </code> On s'aperçoit que la commande fait appel à un fichier utils.pm qu'elle ne trouve pas. Ce fichier appartient en fait aux plugins nagios et la commande le cherche dans le répertoire courant. Il est nécessaire de créer un lien symbolique du répertoire libexec vers le répertoire nagios/plugins (voir http://fr.wikipedia.org/wiki/Lien_symbolique). <code shell> root@ctShinken# ln -s /usr/lib/nagios/plugins/utils.pm /var/lib/shinken/libexec/utils.pm </code> Testez à nouveau la commande : <code shell> shinken@cShinken:~$ /var/lib/shinken/libexec/check*snmp*load.pl -H localhost -C public -f -w 80 -c 90 Argument “v6.0.1” isn't numeric in numeric lt (<) at /var/lib/shinken/libexec/check*snmp*load.pl line 685. 1 CPU, load 14.0% < 80% : OK | cpu*prct*used=14%;80;90 </code> A priori cela fonctionne même si une erreur non bloquante est relevée dans le script. Regardez dans webui :**
* l'erreur sur **TimeSync** est due à l'absence de définition d'un serveur NTP, * l'erreur sur **Log*File*Health** est due à l'absence du paramétrage de l'emplacement des logs.

> **A faire :**
>
> - **Mettre en place NTP** et vérifier sa prise en charge par Shinken :
>   - pour vous aider : https://computerz.solutions/debian-8-shinken-corriger-lerreur-ntpq-not-found/
> - **Enlever la référence** à la commande gérant les logs dans le modèle linux-snmp
> - et vérifier le résultat sur webui.

==== Je reviens au menu Shinken ====

- Supervision des services avec Shinken