Les Sondes

Introduction

Shinken Enterprise s'appuie sur des programmes externes appelés "plugins de check ou sondes" pour pouvoir superviser une large variété d'éléments.

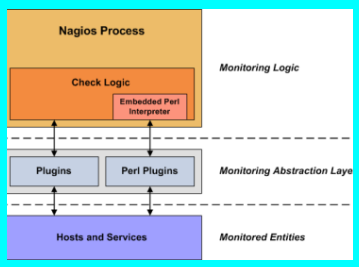
Ce sont les serveurs Shinken Poller et Réactionner qui exécuteront ces sondes. Ces deux daemons devront donc accéder à ces sondes directement sur leur système/serveur.

Un certain nombre de sondes sont inclues lorsque vous installez Shinken Enterprise, mais vous pouvez également en télécharger sur l'Internet.

Si vous avez quelques compétences en scripting, vous pouvez également créer vos propres sondes assez simplement, il suffira d'utiliser les codes retours correspondant.

Sommaire

Qu'est ce qu'une sonde ou un plugin?



Les sondes sont des **exécutables compilés** ou des **scripts** (Python, Perl, Shell, etc.) qui peuvent être lancés par une ligne de commande afin de vérifier le statut d'un hôte ou bien d'un check particulier sur un hôte. Shinken Enterprise utilise les retours de ces sondes pour déterminer le statut des éléments.

Shinken Enterprise lancera une sonde à chaque fois que ce sera nécessaire pour vérifier un statut. La sonde fait *quelque chose* (terme volontairement générique) pour procéder à la vérification en retournant simplement le résultat vers Shinken Enterprise qui prendra alors les actions nécessaires en fonction de ce retour (lancement d'événement, envoi de notifications, etc).

De la même manière que pour Nagios (cf l'image ci contre), les sondes agissent à un niveau d'abstraction entre la logique de supervision présente dans les démons de Shinken Enterprise et les hôtes supervisés.

Le gros intérêt de l'approche "sonde" est que l'**on peut superviser à peu près tout type d'élément tant qu'il est joignable sur un réseau.**

Si vous pouvez automatiser le processus de vérification d'un élément, alors vous pouvez le faire avec Shinken Enterprise.

Il existe plusieurs milliers de sondes (ou plugins) créées pour superviser des ressources du type charge du processeur, utilisation disque, ping, etc..

Si vous souhaitez superviser quelque chose d'autre, référez vous au paragraphe Plugins API et créez votre propre sonde, c'est aussi simple !

Quelles sondes sont disponibles ?

Shinken Enterprise est livré en standard avec un pack de sondes. Ces sondes sont placées dans le répertoire **/var/lib/shinken/libexec** et ces sondes sont utilisées par des commandes Shinken déjà créées lorsque vous installez Shinken Enterprise.

Si vous regarder comment est créé la commande depuis l'UI de configuration, vous verrez que la ligne de commande fait référence à la variable de remplacement **$NAGIOSPLUGINS$** , **$PLUGINSDIR$** ou encore **$USERPLUGINSDIR$**.

Ces variables sont globales, et déclarées depuis le fichier **/etc/shinken/resource.d/paths.cfg** et elle représente les chemins disques des sondes :

Variable	Chemin	Description
$NAGIOSPLUGINS$	/usr/lib64/nagios/plugins	ce répertoire contient un certain nombre de sondes issues de Nagios
$PLUGINSDIR$	/var/lib/shinken/libexec	ce répertoire contient des sondes adaptées et utilisées spécifiquement pour Shinken Enterprise. Attention, les sondes de ce répertoire seront régulièrement mises à jour, ne pas les modifier.
$USERPLUGINSDIR$	/var/lib/shinken-user/libexec	ce répertoire vous permettra de placer vos propres sondes. Ce répertoire ne sera jamais modifié lors de mises à jour.

Dans les répertoires **$NAGIOSPLUGINS$** et **$PLUGINSDIR$**, il y a des sondes pour surveiller tout type d'éléments et services.

Ils utilisent des protocoles classiques WMI, SNMP, SSH, TCP, UDP, ICMP, LDAP et plus encore

Cela permet de surveiller à peu près tout:

- Unix/Linux, Windows
- Routeurs, Switches
- services réseau : "HTTP", "POP3", "IMAP", "FTP", "SSH", "DHCP"
- Charge CPU , utilisation disque, utilisation mémoire..
- Applications, Bases de données, logs et plus.

Note Importante

Vous pouvez également créer votre propre variable globale qui représentera un chemin personnalisé.

Par exemple pour un Poller Windows, vous pouvez rajouter : **$WINPLUGINSDIR$=C:\shinken\libexec**

Droits et accès

Les sondes sont utilisées par les commandes Shinken. Commandes qui peuvent être rattachées directement à des hôtes et des checks (via leurs commandes de vérification), ou encore à des méthodes de notifications ou commandes lancée par le gestionnaire d'événements.

Suivant le cas, elles seront donc exécutées par le **Poller** (pour les hôtes et checks) ou par le **Reactionner** (pour les notifications mais aussi pour les gestionnaires d'évènements).

Le daemon correspondant exécutera alors la commande de son répertoire de sonde. Les processus Shinken des daemons, lancés en tant qu'utilisateur Unix "**shinken**", utiliseront donc cet utilisateur "**shinken**" pour l'exécution des sondes.

Pour la bonne exécution des sondes, il est donc important de s'assurer que l'utilisateur **shinken** puisse correctement accéder en exécution aux scripts/sondes.

Pour cela, si besoin, rajoutez les droits UNIX correspondants :

```
chmod 755 ma_sonde.py
```

ou

```
chmod +x ma_sonde.py
```

Aussi, les droits propriétaires des scripts peuvent rester root/root, il n'est pas nécessaire de donner le droit "ownership" à l'utilisateur shinken via la commande chown.

Comment utiliser une sonde ?

La plupart des sondes présentent des informations de type utilisation en les exécutant avec l'extension "-h" ou "--help" dans la ligne de commande.

Comme vu précédemment, les scripts sont exécutés via l'utilisateur UNIX **shinken**, nous vous conseillons donc d'effectuer vos tests à partir cet l'utilisateur :

```
su - shinken
```

Par exemple, si vous voulez savoir comment fonctionne la sonde **check_http** (sonde du répertoire $NAGIOSPLUGINS$ qui vérifie le statut du port HTTP d'un serveur) ou pour savoir quelles options sont possibles, vous devez lancer la commande suivante :

```
/usr/lib64/nagios/plugins/check_http --help
```

Une fois que vous savez comment s'en servir, vous pouvez alors tenter d'exécuter la sonde avec les paramètres souhaités :

```
/usr/lib64/nagios/plugins/check_http -H www.google.fr
```

Vous obtenez alors la réponse de la sonde :

```
HTTP OK: HTTP/1.1 200 OK - 11837 bytes in 0.313 second response time |time=0.312690s;;;0.000000 size=11837B;;;
0
```

Ici la ligne de retour contient deux parties séparées par le symbole pipe |.

Nous allons voir maintenant comment se compose le retour d'une sonde, de manière à ce que vous puissiez créer vos propres scripts de sonde.

Créer sa sonde (API)

Voyons maintenant comment créer vos propres sondes à placer dans le répertoire **$USERPLUGINSDIR$**.

Concept de sonde

Les scripts et les exécutables doivent faire au moins 2 choses :

- sortir avec l'une des nombreuses valeurs possibles en retour ("code retour")
- retourner au moins une ligne de texte vers "STDOUT"

Le fonctionnement interne de la sonde importe peu à Shinken Enterprise, c'est l'interface entre eux deux qui compte.

Votre sonde peut vérifier le statut d'un port TCP, lancer une requête sur une base de données, ou faire tout ce qui est nécessaire pour vérifier un élément sur un réseau LAN ou WAN ou encore sur Internet.

Les détails dépendront ce qui doit être vérifié.

Code retour

Shinken Enterprise détermine le statut d'un hôte ou d'un check en évaluant le code retour de la sonde.

La table suivante montre la liste des valeurs possibles, avec leur correspondance possible, suivant si la commande est attaché à un Check ou directement sur un hôte :

Code retour	Etat du Check	Etat de l'hôte
0	OK	UP
1	WARNING	DOWN
2	CRITICAL	DOWN
3	UNKNOWN	DOWN

Format de sortie des données de sonde

Au minimum, la sonde doit retourner une ligne de texte vers STDOUT.

La sonde peut également retourner en option des données de performance pour la métrologie.

Le format standard de sortie d'une sonde est donc le suivant :

TEXT OUTPUT | OPTIONAL_PERFDATA

Les données de performance sont optionnelles. Si une sonde retourne des données de performance, elles doivent être séparées du reste du texte par le symbole pipe **|** .

Exemples de retour de sonde

Voyons quelques exemples possibles.

Cas 1: une ligne de retour (texte seulement)

Imaginons une sonde qui retourne une ligne telle que décrite en dessous :

```
DISK OK - free space: /var 3326 MB (56%);
```

Si cette sonde est utilisée pour réaliser une vérification de service, la totalité de la ligne retour sera stockée dans **$SERVICEOUTPUT$** .

Cas 2: une ligne de retour (texte et données de performance)

La sonde peut retourner en option des données de performance utilisables par Graphite. Dans ce cas, les données de performance doivent être séparées par le symbole pipe " **|** " :

```
DISK OK - used space: /var 3326 MB (56%); | /var=3326
```

Si cette sonde est utilisée pour vérifier un service, la partie à gauche du symbole sera stockée dans **$SERVICEOUTPUT$** et la partie à droite du séparateur sera stockée dans **$SERVICEPERFDATA$**

Exemples de retour de sonde avec HTML/CSS

Il est tout à fait possible de retourner du HTML/CSS dans le retour des sondes.

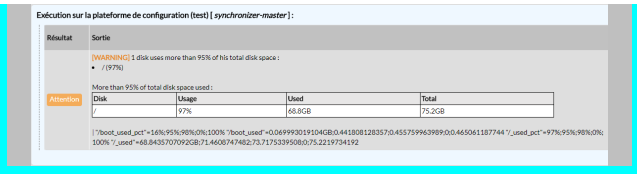
De cette manière, il est possible d'insérer du style et donc d'améliorer le retour des checks affiché dans un navigateur WEB, depuis l'UI de configuration (évaluation des checks) et depuis l'UI de visualisation (liste et panneau de détails).

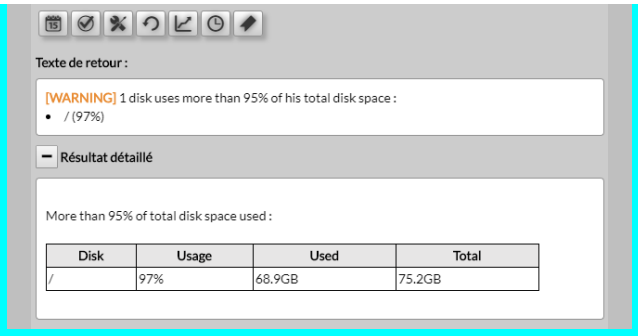
Voici un exemple :

```
<span style="color:#e48c19;font-weight: bold;">[WARNING]</span> 1 disk uses more than 95% of his total disk
space :<br/><li>/ (97%)</li>
<style type="text/css"> .disks-table, .disks-table td, .disks-table th { border: 1px solid #000000 !
important; border-collapse: collapse !important; color: #000000 !important; } .disks-table { width: 90% !
important; } .disks-table-th { background-color: #E8E7E7 !important; width: auto !important; max-width: 20% !
important; padding: 2px !important; word-break: break-word !important; background-color: #E8E7E7 !important;
text-align: center;}.disks-table-td { padding: 2px !important; width: auto !important; max-width: 20% !
important; font-weight: normal !important; word-break: break-word !important; background-color: #FFFFFF !
important; } .disks-host-command { font-style: italic !important; color: #7F7F7F !important; } .disks-table-
center { text-align: center; } </style><br/>More than 95% of total disk space used :<br/> <table class="disks
table"><tr><th class="disks-table-th">Disk</th><th class="disks-table-th">Usage</th><th class="disks-table-th"
>Used</th><th class="disks-table-th">Total</th></tr>
 <tr> <td class="disks-table-td">/</td> <td class="disks-table-td">97%</td><td class="disks-table-td">68.9GB<
/td><td class="disks-table-td">75.2GB</td></tr>
</table><br> | "/boot_used_pct"=16%;95%;98%;0%;100% "/boot_used"=0.069993019104GB;0.441808128357;
0.455759963989;0;0.465061187744 "/_used_pct"=97%;95%;98%;0%;100% "/_used"=68.8651046753GB;71.4608747482;
73.7175339508;0;75.2219734192
```

Depuis un navigateur, ce retour sera affiché comme suivant dans l'interface de configuration :

Depuis un navigateur, ce retour sera affiché comme suivant dans l'interface de visualisation :





Les données de performance

Une donnée de performance est composée au minimum de :

```
key=value
```

La valeur **key** est une suite de caractères qui permettra de donner un titre au graphique qui sera composé des données de performance.
La valeur **value** est un nombre représentant la donnée de performance à l'instant de la requête de la sonde.

> **Remarque**
>
> Pour éviter tout problème de création des graphiques, nous vous conseillons de ne pas mettre d'espaces dans la valeur **key**

Un exemple:
/var=3326

Ici, dans le cas de cette sonde, la valeur **key** représente le nom de la partition disque de serveur, et la valeur **value** représente un volume de cette partition.

Afin de rajouter des informations additionnelles, il est intéressant de rajouter d'autres options séparées par des points virgules :

key=valueUNIT;warning;critical;minimumvalue;maximalvalue

- UNIT : chaîne de caractères représentant l'unité de la valeur - les unités sont facultatives
- warning : seuil de warning affiché dans le graphique
- critical : seuil de critique affiché dans le graphique
- minimum value : Valeur minimale que pourra retourner la sonde
- maximal value : Valeur maximale que pourra retourner la sonde

Ces informations pourront être utilisées et affichées dans les graphiques accessibles depuis l'UI de visualisation via le widget Graphique.

Voici un exemple:

/var=3326MB;5800;5900;0;6000

Ici, la taille de la partition **/var**, récupérée par la sonde à l'instant T, est de 3326MB. Le seuil de warning est de 5800, et le seuil de critique est de 5900. La valeur minimum récupérée sera 0 et la valeur maximale sera de 6000 (car la partition fait 6GB).

Aussi, vous pouvez avoir une série de données de performance séparées par un espace :

```
Ok: all disks are in the limits | "/boot_used_pct"=14%;90%;95%;0%;100% "/_used_pct"=44%;90%;95%;0%;100%
```

Taille maximum du retour d'une sonde

Shinken Enterprise va seulement lire les premiers 64 KB de données qu'une sonde retourne. Cela évite de surcharger une base de données.

Exemple de sonde écrite en python

Voici un exemple très simple et basique d'une sonde écrite en Python afin de vérifier la présence du fichier /tmp/test_file :

```
#!/usr/bin/env python

import os
import sys

# Shinken return values
shinkenRetValOk = 0
shinkenRetValWarn = 1
shinkenRetValCritical = 2

try:
    os.stat("/tmp/test_file")

except:
    print "CRITICAL! Unable to open test_file"
    sys.exit(shinkenRetValCritical)

print "OK! The file /tmp/test_file exists"
sys.exit(shinkenRetValOk)
```

Exemple de sonde écrite en VBS pour un Poller Windows

Voici un exemple très simple et basique d'une sonde check_file_exists.vbs écrite en VBS afin de vérifier la présence d'un fichier passé en argument (avec retour d'une donnée de performance):

```
Dim strfilepath
Dim wsh
Dim Perf_Data

'#########################################################'
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set wsh = CreateObject("WScript.Shell")
'#########################################################'

If Wscript.Arguments.Count = 1 Then
strfilepath = Wscript.Arguments(0)

If (objFSO.FileExists(strfilepath)) Then
        Perf_Data = "|'path_file_existance'=1"
        Wscript.Echo "OK: the file with path " & strfilepath & " exists" & Perf_Data
        Wscript.Quit(0)
else
        Perf_Data = "|'path_file_existance'=0"
        Wscript.Echo "CRITICAL: the file with path " & strfilepath & " does NOT exist" & Perf_Data
        Wscript.Quit(2)
End If

else
        Wscript.Echo "UNKNOWN"
        Wscript.Quit(3)
End If
```

Ici pour l'utilisation de cette sonde via un Poller Windows, la ligne de commande devra être définie comme tel :

```
C:\WINDOWS\SysWOW64\cscript.exe //nologo $WINPLUGINSDIR$\check_file_exists.vbs "$_HOSTFILEPATH$"
```

Récupérer des sondes

Vous pouvez dupliquer des sondes que vous trouverez dans les répertoires **$NAGIOSPLUGINS$** et **$PLUGINSDIR$** et les placer dans votre répertoire **$USERPLUGINSDIR$** pour les personnaliser.

Mais vous pouvez en trouver beaucoup d'autres dans la communauté Nagios, et plus particulièrement dans les espaces suivants:

- Shinken Monitoring Plugins: http://www.shinken.io/ - Des informations pour l'installation de pack Shinken IO ici
- Monitoring Plugins Project: https://www.monitoring-plugins.org
- Nagios Plugins Exchange: https://exchange.nagios.org/

Mise en place d'une sonde dans Shinken Enterprise

Une fois avoir créée votre propre sonde ou une fois l'avoir téléchargée, il faut maintenant la mettre en place dans Shinken afin de pouvoir facilement l'utiliser via des checks appliqués à des modèles de d'hôtes. Nous vous conseillons fortement de passer par des modèles d'hôtes et des checks dédiés à ces modèles pour pouvoir réutiliser ce check le plus simplement possible.

Mise en place de votre sonde

- Créer ou télécharger votre sonde via une commande wget (par exemple) depuis vos serveurs Shinken Poller.
- Placer la sonde dans **$USERPLUGINSDIR$** (/var/lib/shinken-user/libexec/) de votre ou vos daemons Shinken Poller.

Droits d'exécution

- Donner les droits d'exécution : chmod +x monscript.py

Création de la commande Shinken

- Ouvrir l'interface utilisateur (UI) de configuration et créer une commande "cmd_monscript" (par exemple) avec la ligne de commande suivante : **$USERPLUGINSDIR$\monscript.py**
- Si votre sonde prend des arguments, ajustez votre ligne de commande. Vous pouvez vous inspirer d'autres commandes.

Création d'un modèle d'hôte

- Créer un modèle d'hôte, par exemple : "monmodele"

Création d'un check dédié

- Pour appliquer systèmatiquement le script lors de l'application de "monmodele" à un hôte, créer un "check dédié à modèle d'hôte", par exemple check_monscript (cf par exemple le check dédié "http").
- Dans ce check dédié, mettre "generic-service" dans la propriété "Modèle de Check hérité" et mettre le modèle "monmodèle" dans la propriété "Attaché sur les modèles d'hôte"
- Faire pointer la commande de vérification vers la commande cmd_monscript

Application du modèle et évaluation

- Appliquer le modèle "monmodèle" à des hôtes, le check doit bien être appliqué, vous pouvez essayer ce check depuis l'onglet "Checks"