# Activité : Architecture de Shinken

Vous allez observer le contenu des différents répertoires de Shinken et exécuter les commandes proposées pour découvrir l'architecture de Shinken.

A ce stade, vous avez installé les **modules obligatoires** de shinken et les **modules optionnels** webui et mongodb.

# Les répertoires associés à Shinken

```
root@ctShinken:~# find / -name shinken
/usr/bin/shinken
/usr/local/lib/python2.7/dist-packages/shinken
/home/shinken
/etc/shinken
/etc/init.d/shinken
/etc/default/shinken
/run/shinken
/var/log/shinken
/var/mail/shinken
/var/lib/shinken
/var/lib/shinken/doc/build/html/_modules/shinken

root@ctShinken:~# ls /var/lib/shinken/modules/
auth-cfg-password dummy_arbiter dummy_broker dummy_broker_external
dummy_poller dummy_scheduler __init__.py mod-mongodb webui

root@ctShinken:~# ls /var/lib/shinken/libexec
discovery dump_vmware_hosts.py external_mapping.py
link_libvirt_host_vm.py link_vmware_host_vm.py link_xen_host_vm.py
notify_by_email.py notify_by_xmpp.ini notify_by_xmpp.py
send_nsca.py service_dependency_mapping.py
```

- Répertoire contenant les modules : **/var/lib/shinken/modules**.
- Répertoire contenant les sondes : **/var/lib/shinken/libexec**.
- Fichiers de configuration : **/etc/shinken**.
- Lanceurs : **/etc/init.d/shinken**.
    - Démons shinken : **/usr/bin**.
    - Fichiers de log : **/var/log/shinken**.

Les fichiers/répertoires de configuration sont :

```
root@ctShinken:~# ls /etc/shinken/
arbiters      brokers     certs         commands    contactgroups
contacts      daemons     dependencies  dev.cfg     discovery
escalations   hostgroups  hosts         modules     notificationways
packs         pollers     reactionners  realms      receivers
resource.d    sample      sample.cfg    schedulers  servicegroups
services      shinken.cfg templates     timeperiods
```

Quand on démarre **shinken**, on constate qu'en réalité le script démarre plusieurs démons.

Chacun de ces démons dispose d'un fichier de configuration dans **/etc/shinken**.

```
root@ctShinken:~#  service shinken start
Starting scheduler:
.
Starting poller:
.
Starting reactionner:
.
Starting broker:
.
Starting receiver:
.
Starting arbiter:
```

.

C'est ici la **principale différence** entre **shinken** et son aîné **nagios**. Son architecture est modulaire, elle repose sur le principe Unix : à une tâche un outil.

Shinken utilise **six processus** (ou composants) différents qui travaillent ensemble et permettent d'obtenir une flexibilité bien supérieure au **nagios** originel. Surtout, contrairement à un processus monolithique, cette architecture favorise la montée en charge.

Shinken se décompose ainsi :

- **L'arbitre (Arbiter)** : il est notamment responsable de la validation et du chargement de la configuration, de la découpe en différentes parties, et de l'envoi aux autres éléments.
- **L'ordonnanceur (Scheduler)** : il est chargé d'ordonnancer les « **checks** », d'analyser leurs résultats et de déclencher une action en fonction de ces derniers si c'est nécessaire. Ce n'est pas lui qui lance les checks ou les notifications, il ne fait que rediriger les informations. Il garde juste dans une file d'attente les « **checks** » en attentes (pending) et notification pour les autres éléments (collecteurs ou « **Réactionneurs** »). Il peut y avoir plusieurs ordonnanceurs, c'est d'ailleurs conseillé.
- **Le collecteur (Poller)** : son rôle est de lancer les plugins en fonction des requêtes des ordonnanceurs. Ces plugins, qui peuvent être ceux de Nagios, vont aller interroger le système surveillé et retourner un résultat indiquant l'état. Lorsqu'un plugin renvoie un résultat, il le transmet à l'ordonnanceur. Il peut y avoir plusieurs collecteurs.
- **Le « réactionneur » (Reactionner)** : il est chargé de l'envoi des notifications et de lancer les « **event_handlers** » (action automatique programmable).
- **Le « courtier » (Broker)** : son rôle est de prendre des données sur les schedulers (comme les statuts par exemple) et de les rendre disponibles à l'externe de Shinken.
- **Le « receveur » (Receiver)** : son rôle est de recevoir les données d'acquisition passive et de les acheminer vers le bon scheduler responsable de faire la corrélation et le traitement des statuts. Il peut y avoir plusieurs receveurs.

On peut observer les processus en cours d'exécution grâce à la commande suivante :

```
root@ctShinken:~# ps -AHF | grep shinken
root     32592   733  0  3187  2160   0 21:21 tty1     00:00:00       grep shinken
shinken  23134     1  1 149945 41444  0 20:53 ?        00:00:20   python2.7 /usr/bin/shinken-scheduler -
d -c /etc/shinken/daemons/schedulerd.ini
shinken  23138 23134  0 35323 34136   0 20:53 ?        00:00:00     python2.7 /usr/bin/shinken-scheduler
-d -c /etc/shinken/daemons/schedulerd.ini
shinken  23153     1  0 131465 41624  0 20:53 ?        00:00:12   python2.7 /usr/bin/shinken-poller -d -
c /etc/shinken/daemons/pollerd.ini
shinken  23156 23153  0 72517 32716   0 20:53 ?        00:00:00     python2.7 /usr/bin/shinken-poller -d
-c /etc/shinken/daemons/pollerd.ini
shinken  23334 23153  0 129333 39016  0 20:53 ?        00:00:00     python2.7 /usr/bin/shinken-poller -d
-c /etc/shinken/daemons/pollerd.ini
shinken  23175     1  0 147914 41900  0 20:53 ?        00:00:11   python2.7 /usr/bin/shinken-reactionner
-d -c /etc/shinken/daemons/reactionnerd.ini
shinken  23178 23175  0 72517 32716   0 20:53 ?        00:00:00     python2.7 /usr/bin/shinken-
reactionner -d -c /etc/shinken/daemons/reactionnerd.ini
shinken  23352 23175  0 129335 39212  0 20:53 ?        00:00:00     python2.7 /usr/bin/shinken-
reactionner -d -c /etc/shinken/daemons/reactionnerd.ini
shinken  23199     1  0 150088 42228  0 20:53 ?        00:00:16   python2.7 /usr/bin/shinken-broker -d -
c /etc/shinken/daemons/brokerd.ini
shinken  23202 23199  0 72312 36112   0 20:53 ?        00:00:00     python2.7 /usr/bin/shinken-broker -d
-c /etc/shinken/daemons/brokerd.ini
shinken  23292 23199  0 129989 42552  0 20:53 ?        00:00:01     python2.7 /usr/bin/shinken-broker -d
-c /etc/shinken/daemons/brokerd.ini
shinken  23220     1  0 131343 40252  0 20:53 ?        00:00:10   python2.7 /usr/bin/shinken-receiver -d
-c /etc/shinken/daemons/receiverd.ini
shinken  23223 23220  0 35318 33856   0 20:53 ?        00:00:00     python2.7 /usr/bin/shinken-receiver
-d -c /etc/shinken/daemons/receiverd.ini
root     23237     1  0 111574 42064  0 20:53 ?        00:00:10   python2.7 /usr/bin/shinken-arbiter -d
-c /etc/shinken/shinken.cfg
root     23238 23237  0 37592 39716   0 20:53 ?        00:00:00     python2.7 /usr/bin/shinken-arbiter -
d -c /etc/shinken/shinken.cfg
```

- L'option **–A** affiche tous les processus.
- L'option **–H** affiche l'arborescence des processus.
- L'option **–F** affiche toutes les informations associées au processus.

On s'aperçoit ici que les processus Shinken sont en fait pris en charge par l'interpréteur Python.

> On constate aussi, élément important à noter, que les processus ne sont pas exécutés avec les droits root mais avec les droits d'un utilisateur shinken.

On voit aussi les fichiers de paramètres utilisés par les commandes (**.ini**).

Tous les programmes Python utilisés par Shinken se trouvent dans **/usr/bin** et les fichiers de configuration dans **/etc/shinken**.

On peut repérer les ports TCP écoutés par Shinken en exécutant la commande suivante :

```
root@ctShinken:~# netstat -tlnp | grep "LISTEN .*python"
tcp        0      0 127.0.0.1:39026         0.0.0.0:*               LISTEN      30925/python2.7
tcp        0      0 127.0.0.1:47924         0.0.0.0:*               LISTEN      31145/python2.7
tcp        0      0 0.0.0.0:7767            0.0.0.0:*               LISTEN      31383/python2.7
tcp        0      0 127.0.0.1:45047         0.0.0.0:*               LISTEN      31322/python2.7
tcp        0      0 0.0.0.0:7768            0.0.0.0:*               LISTEN      30847/python2.7
tcp        0      0 0.0.0.0:7769            0.0.0.0:*               LISTEN      30999/python2.7
tcp        0      0 127.0.0.1:7770          0.0.0.0:*               LISTEN      31319/python2.7
tcp        0      0 0.0.0.0:7771            0.0.0.0:*               LISTEN      30922/python2.7
tcp        0      0 0.0.0.0:7772            0.0.0.0:*               LISTEN      31143/python2.7
tcp        0      0 0.0.0.0:7773            0.0.0.0:*               LISTEN      31222/python2.7
tcp        0      0 127.0.0.1:33502         0.0.0.0:*               LISTEN      31224/python2.7
tcp        0      0 127.0.0.1:50599         0.0.0.0:*               LISTEN      31002/python2.7
tcp        0      0 127.0.0.1:55719         0.0.0.0:*               LISTEN      30849/python2.7
```

options de netstat :

- **t** pour tcp,
- **l** pour listenning,
- **n** pour numeric
- et **p** pour program

Les différents ports sont définis dans les fichiers de configuration associés à chaque module. Par exemple pour le module **poller**, on trouve dans **/etc/shinken/pollers/poller-master.cfg** une directive port :

```
#=============================================================================
# POLLER (S1_Poller)
#=============================================================================
# Description: The poller is responsible for:
# - Active data acquisition
# - Local passive data acquisition
# https://shinken.readthedocs.org/en/latest/08_configobjects/poller.html
#=============================================================================
define poller {
    poller_name     poller-master
    address         localhost
    port            7771
```

Chaque modification dans les fichiers de configuration nécessite le redémarrage de **l'Arbiter**.

Le démarrage de l'arbiter se fait :

- soit en relançant l'ensemble des démons avec **service shinken restart**,
- soit en lançant la commande suivante pour ne lancer que **l'arbiter**.

```
root@ctShinken:~# service shinken-arbiter restart
Restarting arbiter
Doing config check
```

### Je reviens au menu Shinken

- Supervision des services avec Shinken