

Shinken-healthcheck: Vérifier le bon fonctionnement de Shinken Entreprise

Qu'est-ce que le Health Check ?

Le Health Check est une commande présente dans toute installation Shinken Entreprise qui permet de vérifier le bon fonctionnement de Shinken Entreprise.

Cet outil est utilisé pour vérifier :

- L'état de l'installation de Shinken Entreprise (version des démons).
- L'état des principales options de configuration réseau (ports, adresses).
- L'état des modules et sous-modules activés sur les démons.
- L'état des connexions réseau et la synchronisation d'horloge entre les démons.

Le Health Check est donc un outil de diagnostic général qui peut détecter les problèmes les plus importants. Cependant, il ne fournit pas autant d'informations et de détail que les checks fournis par Shinken pour son propre monitoring, par exemple des indicateurs de performance.

Sommaire

Qu'est-ce que le Health Check ?

Informations de version
Vérification de l'architecture
Vérification de la licence
Vérification des bibliothèques externes
Vérification des interfaces de Configuration et Visualisation
Vérification des espaces de stockage
Vérification des add-ons
Historique des installations et de la configuration
Comment interpréter les informations et erreurs courantes du Healthcheck

Usage

```
shinken-healthcheck
```

Principales options

Option	Option longue	Description
-h	--help	Affiche le message d'aide
-v	--version	Affiche la version de Shinken Entreprise installée
-l	--local	Effectue une vérification des démons locaux seulement
-g	--global	Effectue une vérification complète des démons (doit être lancé depuis la machine comportant l'Arbiter et le Synchronizer). Par défaut sur une machine avec un Arbiter et un Synchronizer, un Health Check global est effectué sauf si un Health Check local est explicitement demandé.
	--debug	Active l'affichage des données de debug dans la sortie de la commande. Utile seulement dans le cas d'un envoi de ces données aux équipes de support de Shinken Solutions.

-f	--file	Ecrit la sortie de la commande dans un fichier. La sortie de la commande est également affichée.
	--output-directory	Dossier dans lequel sera placé le fichier de sortie. Par défaut, le dossier courant est utilisé.
	--output-name	Fichier dans lequel sera placé le fichier de sortie. Valeur par défaut: shinken-healthcheck_\$(DATE).txt
	--timeout	Temps en secondes a partir duquel un démon sera considéré comme injoignable. Par défaut: 3 secondes
	--modules-warning-expire	Temps en minutes pendant lequel un redémarrage de module génère une alerte. Par défaut 120 (2 heures), valeur maximale 1440 (24 heures)
	--show-history	Affiche l'historique des installations et donnés de Shinken Entreprise sur ce serveur

La commande shinken-healthcheck sépare sa vérification en plusieurs parties qui sont décrites dans les sections suivantes.

Informations de version

```

=====
This tool is used to check the state of your Shinken Enterprise (02.04.01) installation and configuration
Note: Global check launch as launch from a arbiter server
=====
Healthcheck report 13/09/2017 16:19:31
-----
shinken-healthcheck versions:
  Original installed version : 2.03.03-40
  Updated version           : 02.04.01-002.fr
=====
[.....] 100%
Architecture

```

La première section visible dans le Healthcheck est l'affichage de la version installée, ainsi que la première version installée sur ce serveur.

Vérification de l'architecture

Etat des démons

Le Healthcheck affiche ensuite pour tous les démons activés dans la configuration, différentes informations indiquant le bon fonctionnement du démon:

- Le type et le nom du démon. Si celui est un Spare, une mention "SPARE" est présente à la suite du nom du démon.
- La configuration est-elle valide ?
- Le démon est-il joignable sur le port trouvé dans la configuration ?
- La version actuelle du démon. Si il y a une différence de version entre l'Arbiter et le démon, un message d'erreur indique cette différence.
- Connexion avec l'Arbiter, ainsi que le décalage de temps entre le démon et l'Arbiter.
- Si plusieurs Arbiters envoient une configuration au démon, un message d'erreur indique que 2 Arbiters sont en conflit, en précisant l'URL de chacun de ces arbiters.
- Liste des autres démons à contacter pour pouvoir fonctionner correctement (section "Talk to").

```

[broker: broker-master-2]
OK: Configuration seems valid
OK: Connection to daemon is OK at port 7772
OK: Daemon version is: 02.04.01-002.fr
OK: Correct connection from arbiter "arbiter-master" ( and no time shift )
Module WebUI:
OK: Name: WebUI Type: webui
Submodules:
OK: Name: Mongodb Type: mongodb
OK: Name: Cfg_password Type: cfg_password_webui
OK: Name: trending Type: trending
OK: Name: webui-enterprise Type: webui_enterprise
OK: Name: sla Type: sla
Modules:
OK: Name: Graphite-Perfdata Type: graphite_perfdata
OK: Name: Livestatus Type: livestatus
OK: Name: sla Type: sla
Talk to:
OK: Reachable poller satellite (poller-master2) at https://192.168.0.2:7771
OK: Reachable scheduler satellite (scheduler-master2) at https://master2:7768

```

- Etat des modules, et le cas échéant des sous-modules. Si un module a redémarré récemment, un avertissement sera affiché en indiquant la liste des derniers redémarrages du module ou sous-modules.
- Champs spécifiques au démon
 - Liste des tags pour le Poller et Reactionner, ainsi que les éventuelles erreurs sur les workers.

Dans l'affichage de l'état des démons, ainsi que dans les sections suivantes, plusieurs états peuvent être retournés:

- **OK**: Tout va bien
- **AT RISK**: Problème pouvant potentiellement nuire au fonctionnement du système.
- **ERROR**: Une erreur bloquante a été détectée.

Royaumes et sous-royaumes

Dans une configuration de Shinken Entreprise, les démons peuvent être répartis sur plusieurs machines.

Dans le Healthcheck, les démons sont regroupés en fonction de la machine sur laquelle ils sont installés.

On voit dans l'exemple ci-contre qu'un Poller est installé sur la machine d'adresse 192.168.1.35, et qu'un Arbitrer et un Broker sont installés et activés sur la machine vm3 (172.16.0.3).

```
- 192.168.1.35 (192.168.1.35):
*****
[poller: poller-spare] - SPARE
  ERROR Cannot contact daemon 192.168.1.35:7771 ( timed out )
  OK Configuration seems valid

- vm3 (172.16.0.3):
*****
[arbitrer: arbitrer-vm3]
  OK Configuration seems valid
  OK Connection to daemon is OK at port 7770
  OK Daemon version is: 02.04.01-003_BUIL001.fr
  OK This element is defined as the master arbitrer
Modules:
  OK Name: synchronizer-import Type: synchronizer-import
Talk to:
  ERROR Unreachable poller satellite (poller-spare) at http://192.168.1.35:7771 (timed out)
  OK Reachable scheduler satellite (scheduler-vm3) at http://vm3:7768
  OK Reachable reactionner satellite (reactionner-vm3) at http://vm3:7769
  OK Reachable poller satellite (poller-vm3) at http://vm3:7771
  OK Reachable broker satellite (broker-vm3) at http://vm3:7772
  OK Reachable receiver satellite (receiver-vm3) at http://vm3:7773
[broker: broker-vm3]
  OK Configuration seems valid
```

Si plusieurs royaumes sont définis, la sortie de shinken-healthcheck organise les machines par royaume et sous-royaumes, puis les démons sont répartis par machine d'installation.

Dans l'exemple ci-contre, on voit que la configuration comporte 4 royaumes, agencés comme suivant:

- Un royaume principal: **France**
 - Un Sous Royaume: **Corse**
 - Un Sous Royaume: **Sud Ouest**
 - Un Sous Royaume: **Bordeaux**

On voit aussi, pour chaque royaume, les démons activés ainsi que la machine sur laquelle ils sont installés. Dans l'exemple de healthcheck, on peut faire le récapitulatif suivant:

- Royaume **France**: 8 démons répartis sur 2 machines
 - Machine d'adresse a.a.a.a: Poller
 - Machine master1 (b.b.b.b): Arbitrer, Broker, Poller, Reactionner, Synchronizer, Receiver, Scheduler
- Royaume **France/Corse**: 3 démons installés sur une seule machine
 - Machine master2 (d.d.d.d): Broker, Poller, Scheduler
- Royaume **France/Sud Ouest**: Un démon installé sur une machine
 - Machine master2: Broker
- Royaume **France/Sud Ouest/Bordeaux**: 2 démons installés
 - Machine master2: Poller, Scheduler

Exemple d'architecture

```
-----
| Realm /France |
-----
```

```
-----
| In France/ |
-----
```

```
- a.a.a.a (a.a.a.a):
*****
[poller: poller-windows1]
...

- master1 (b.b.b.b):
*****
[arbitrer: arbitrer-master]
...
[broker: broker-master]
...
[poller: poller-master1]
...
[reactionner: reactionner-master]
...
[synchronizer: synchronizer-master]
...
[receiver: receiver-1]
...
[scheduler: scheduler-master]
...
```

```
-----
| Realm /France/Corse |
-----
```

```
-----
| In Corse/ |
-----
```

```
- master3 (d.d.d.d):
*****
```

```

[broker: broker-master-3]
...
[poller: poller-master3]
...
[scheduler: scheduler-master3]
...

-----
| Realm /France/Sud Ouest |
-----

-----
| In Sud Ouest/ |
-----

- master2 (c.c.c.c):
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[broker: broker-master-2]
...

-----
| Realm /France/Sud Ouest/Bordeaux |
-----

-----
| In Bordeaux/ |
-----

- master2 (c.c.c.c):
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[poller: poller-master2]
...
[scheduler: scheduler-
master2]
...

```

Vérification de la licence

Cette section du Healthcheck affiche des informations sur la licence en cours.

Elle affiche le propriétaire de la licence, le type de licence, et la date d'expiration de la licence.

Si le nombre de noeuds est dépassé ou que la licence est expirée, une erreur sera affichée dans cette section.

```

License key
[+] The license key is valid
The license key (Customer: Node limit:500) is valid (start: end: => 111 days remaining)

```

Vérification des librairies externes

Shinken Enterprise utilise de nombreuses librairies externes pour fonctionner.

Cette partie du Healthcheck vérifie que toutes les librairies nécessaires au bon fonctionnement de Shinken Enterprise sont installées sur la machine, ainsi que leur version.

En cas d'erreur sur une des librairies, une erreur est affichée indiquant la nature de l'erreur.

```

Local libraries
[+] [event] Library event is available. Version: 0.13.0
[+] [ldap] Library ldap is available. Version: 2.3.10
[+] [pycurl] Library pycurl is available. Version: libcurl/7.52.1 OpenSSL/1.0.1e zlib/1.2.3 c-ares/1.12.0 libssh2/1.8.0 nghttp2/1.6.0
[+] [pyzmq] Library pyzmq is available. Version: 2.2.2

```

Vérification des interfaces de Configuration et Visualisation

La section Modules effectue des vérifications sur le fonctionnement des interfaces de Configuration et Visualisation. Pour chaque interface, le Healthcheck vérifie que la base de données Mongo est accessible, et que les paramètres d'authentification sont bien définis.

```

Interfaces
[Configuration UI]
    OK:      Mongodb server is available at: mongodb://localhost/?safe=false
    OK:      Auth_secret is a custom variable
    OK:      The master_key is a custom variable
[Visualization UI]
    WebUI
        OK:      Auth_secret is a custom variable
        OK:      Mongodb server is available at: mongodb://localhost/?safe=true

```

Vérification du statut du chiffrement des champs protégés

La sous-section Encryption status donne des informations sur le chiffrement des champs protégés :

- Protection activée ou pas (et nom de la clé si elle est activée)
- Si la clé n'a pas été sauvegardée ou si la clé ne peut pas être chargée une erreur est affichée
- Si il y a une incohérence de configuration du chiffrement (voir la page [Mécanismes de sécurisation des données](#)), le message d'erreur correspondant sera affichée.

```

InterFaces
[Configuration UI]
OK: MongoDB server is available at: mongodb://localhost:27020/unsafe=false
OK: Auth secret is a custom variable
[Encryption status]
OK: The key has never been extracted. Run shinken-protected-fields-key
ERRR: Encryption DISABLED with key named : cif
[Visualization UI]
On broker-master : localhost
OK: Auth secret is a custom variable
OK: MongoDB server is available at: mongodb://localhost:27020/unsafe=true
[Integration status]
OK: I can locate my /bin/shinken-protected-fields-key. No fault file or directory you must to restore the key (only, using shinken-protected-fields-key)

```

Vérification des espaces de stockage

Shinken Enterprise sauvegarde les données de métrologie dans un serveur Graphite. Si la configuration de la sauvegarde des données de métrologies est simple pour une configuration basique, elle devient rapidement compliquée lorsque la configuration comporte plusieurs royaumes avec plusieurs Brokers.

Lorsque plusieurs Broker sont dans un même royaume, il faut s'assurer que les Brokers écrivent les données de métrologie au même endroit (sur le ou les mêmes serveurs Graphite). Il faut également s'assurer que toutes les interfaces de Visualisation d'un royaumes lisent les données de métrologie sur le même serveur, afin d'assurer une cohérence des données. Aussi, il faut s'assurer de ne pas avoir oublié de configurer au moins un broker d'un royaume pour écrire les données de métrologie dans la base Graphite, sans quoi aucune donnée de métrologie ne sera sauvegardée.

The Healthcheck possède donc une section dédiée à la vérification des espaces de stockage des données de métrologie, qui vérifie que les données de métrologie sont bien sauvegardées pour chaque royaume, que toutes les interfaces de visualisation lisent bien les données de métrologie au bon endroit, et que les serveurs de métrologie sont effectivement joignables et aptes à sauvegarder des données.

La première section de la vérification des espaces de stockage Graphite s'occupe de vérifier que pour chaque royaume, les données de métrologie sont bien sauvegardées dans au moins un serveur Graphite.

Sur l'exemple ci-contre, on voit que tous les royaumes ont bien au moins un Broker qui permet la sauvegarde des données de météorologie:

- Le royaume **Corse** sauvegarde ses données de métrologie sur le serveur **192.168.1.47**
- Le royaume **Metropole** sauvegarde ses données de métrologie sur le serveur **172.16.0.3**
- Le royaume **Reunion** sauvegarde ses données de métrologie sur le serveur **192.168.1.35**

Si un royaume ne possède aucun Broker sauvegardant les données de météorologie, une erreur sera affichée.

On note que si il n'y a aucun hôte à superviser dans le royaume, le Broker n'a pas besoin de sauvegarder de données de métrologie, et donc ne sera pas indiqué en erreur si il ne sauvegarde pas de données dans Graphite.

La vérification des espaces de stockage vérifie ensuite si toutes les interfaces de Visualisation d'un royaume.

Pour chaque royaume défini dans la configuration, le healthcheck va vérifier qu'il existe bien au moins un Broker de ce royaume permettant d'afficher les données de métrologie.

Si le royaume comporte plusieurs Brokers, le healthcheck vérifie que tous les Brokers ont la même configuration en ce qui concerne le serveur Graphite à utiliser pour récupérer les données de métrologie.

[illegible][illegible]

Enfin, lors de la vérification des espaces de stockage, le healthcheck cherche à déterminer si tous les serveurs utilisés dans les différents royaumes pour stocker des données de métrologie sont joignables.

Le healthcheck vérifie également que tous ces serveurs permettent bien de lire et écrire des données.

Ainsi, pour chaque serveur, le healthcheck demande au Broker d'essayer d'effectuer une écriture et une lecture de données, et affiche le résultat.

On voit ici dans l'exemple que tous les serveurs utilisés pour le stockage des données de métrologie sont bien joignables et permettent effectivement d'écrire et de lire des données de métrologie.

```
storage (graphite)
  Write connection status: OK
  Read connection status: OK
  Note: Global check launch as Launch from a arbitrer server
  Healthcheck report 17/04/2018 16:35:14

Server(s)
192.168.8.3:
  Write connection status: OK
  Read connection status: 5 hosts (with metrics) can be requested.
  Note: Global check launch as Launch from a arbitrer server
192.168.1.35:
  Write connection status: OK
  Read connection status: 3 hosts (with metrics) can be requested.
  Note: Global check launch as Launch from a arbitrer server
192.168.1.47:
  Write connection status: OK
  Read connection status: 1 hosts (with metrics) can be requested.
  Note: Global check launch as Launch from a arbitrer server
```

Vérification des addons

La dernière section du Healthcheck affiche l'état des différents addons actifs sur la machine.

La liste des addons actuellement activés est affichés, avec pour chacun, leur statut ainsi que l'état de différentes vérifications.

Puisque les addons peuvent avoir chacun un fonctionnement différent, les vérifications effectuées diffèrent selon l'addon.

```
local addons
[nagvis-shinken-architecture]
  OK: Module configuration file found (/etc/shinken/modules/architecture-export.cfg)
  OK: NagVis installation found (/etc/shinken/external/nagvis)
  OK: 'nagvis-shinken-architecture' addon is running correctly at http://localhost/shinken-core-map

[nagvis]
  OK: NagVis installation found (/opt/nagvis)
  OK: 'nagvis' addon is running correctly at http://localhost/shinken-map
```

Historique des installations et de la configuration

Le Healthcheck affiche par défaut la version initiale d'installation ainsi que la version actuellement installée.

Il est possible d'obtenir des informations supplémentaires sur l'historique de l'installation en utilisant l'option "--show-history" de la commande "shinken-healthcheck":

```
shinken-healthcheck --show-history
```

Ce paramètre affiche des informations sur l'état de l'installation depuis son installation initiale.

L'utilisation de cette option est surtout pratique lorsque vous communiquez avec le support Shinken, qui utilisera les informations remontées pour plus facilement remonter à l'origine de votre problème.

Historique des mises à jour

L'option --show-history affiche d'abord une liste triée chronologiquement des différentes versions de Shinken installées sur la machine courante.

On peut donc voir facilement, pour une date donnée, quelle était la version de Shinken installée.

Cette option est également utile pour communiquer avec le support Shinken, pour détecter des erreurs de configuration liées à des anciennes versions de Shinken.

```
$ shinken-healthcheck --show-history
=====
This tool is used to check the state of your Shinken Enterprise (02.05.00) installation and configuration
Note: Global check launch as Launch from a arbitrer server
=====
Healthcheck report 17/04/2018 16:35:14

shinken-healthcheck versions:
Original installed version : 02.05.00-001_BUILD47.fr
Updated version           : 02.05.00-006_BUILD01.fr

INSTALLATION HISTORY:
- INSTALLATION : version=02.05.00-001_BUILD47.fr date=2018-03-12
- UPDATE       : to version=02.05.00-002_BUILD01.fr date=2018-03-15
- UPDATE       : to version=02.05.00-004_BUILD01.fr date=2018-03-21
- UPDATE       : to version=02.05.00-004_BUILD04.fr date=2018-03-23
- UPDATE       : to version=02.05.00-004_BUILD08.fr date=2018-03-28
- UPDATE       : to version=02.05.00-005_BUILD03.fr date=2018-03-29
- UPDATE       : to version=02.05.00-005_BUILD04.fr date=2018-03-29
- UPDATE       : to version=02.05.00-005_BUILD05.fr date=2018-03-30
- UPDATE       : to version=02.05.00-005_BUILD06.fr date=2018-03-30
- UPDATE       : to version=02.05.00-005_BUILD07.fr date=2018-03-30
- UPDATE       : to version=02.05.00-006_BUILD02.fr date=2018-04-13
- UPDATE       : to version=02.05.00-006_BUILD01.fr date=2018-04-17
```

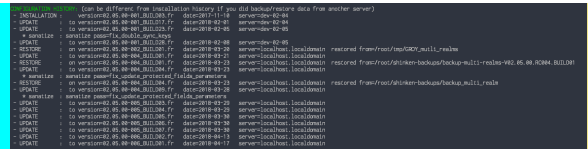
Historique des données

Après l'historique des installations vient l'historique des données présentes dans Shinken. Cet historique de données est séparé en plusieurs sections correspondant chacune à un type de donnée différente:

- Configuration (configuration de Shinken, élément de supervision)
- SLA (historique des états des éléments supervisés et données de SLA)
- Métrologie (métriques)
- Données utilisateurs (données sauvegardées dans l'interface de visualisation)

On dispose encore d'une liste chronologique présentant les différentes opérations effectuées sur les données:

- Mise à jour
- Restauration d'une sauvegarde
- Exécution du script de nettoyage des données, qui retravaille la configuration automatiquement lors d'une mise à jour pour la rendre compatible avec les dernières versions de Shinken Enterprise



Historique des paramètres de chiffrement des données

Le dernier type d'information remonté par l'option `--show-history` est l'historique des paramètres de chiffrement des données sensibles.

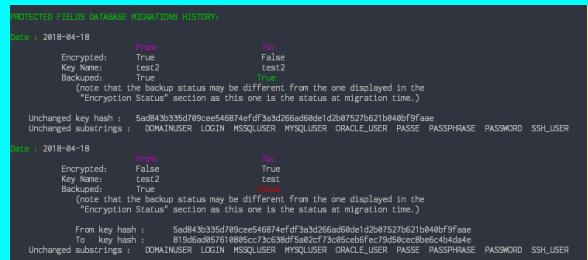
A chaque fois que les paramètres de chiffrement sont modifiés et déclenchent un changement dans le chiffrement des données (activation, désactivation, changement des champs chiffrés), une entrée sera ajoutée dans la liste des modifications.

L'état d'activation, le nom de la clé utilisée, l'état de sauvegarde de la clé ainsi que la liste de définition des données à chiffrer sont affichés pour chaque entrée de la liste.

Notez que le statut de sauvegarde de la clé peut être différent du statut affiché dans la section **[Encryption Status]** :

- dans la section **[Encryption Status]**, shinken-healthcheck affiche l'état actuel.
- dans l'historique, shinken-healthcheck affiche l'état au moment de la modification de la configuration.

Seuls les cinq derniers changements sont conservés.

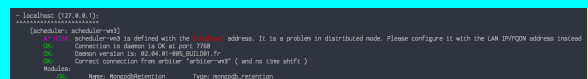


Comment interpréter les informations et erreurs courantes du Healthcheck

Le healthcheck peut afficher de nombreux cas d'erreur différents. Aussi, certaines informations importantes sont présentées régulièrement. Pour faciliter l'utilisation du Healthcheck, cette section présente ces différentes informations et erreurs courantes, accompagnées de leur interprétation.

Le démon est configuré avec l'adresse "localhost"

Puisque les démons constituant une installation de Shinken Enterprise ont pour vocation à être utilisés dans une architecture distribuée, il ne faut pas définir les démons avec des adresses locales. Les démons vont devoir communiquer entre eux, il faut donc les configurer avec une adresse précise pour éviter tout problème dans une architecture distribuée.



Cette erreur est très fréquemment rencontrée dans le healthcheck puisque dans l'installation par défaut de Shinken Entreprise, les démons utilisent l'adresse localhost. Dès que des démons seront utilisés dans la même configuration sur plusieurs machines différentes, il est impératif d'utiliser des adresses précises.

Le démon est injoignable

Le Scheduler "*scheduler-vm3*" est injoignable.

Dans ce cas, on exécute le healthcheck sur la machine vm3. On peut donc déduire que le démon est éteint.

Sur un démon hébergé sur une machine distante, il faudra d'abord déterminer si la machine est joignable avant de pouvoir affirmer que le démon est éteint. Un healthcheck local sur la machine hébergeant le démon peut confirmer cette hypothèse.

```
[scheduler: scheduler-vm3]
[WARN] Cannot contact daemon vm3:7768 ( [Errno 111] Connection refused )
[OK] Configuration seems valid
```

Le démon et son Arbiter ont des versions différentes

Pour garantir un bon fonctionnement, il faut que tous les démons présents dans la configuration possèdent la même version de Shinken Entreprise.

Dans ce cas, le poller "*poller-domtom*" et l'Arbiter n'ont pas la même version installée, ce qui peut provoquer des dysfonctionnements.

```
[poller: poller-domtom]
[WARN] Daemon version is 02.04.01-003_BUILD04.fr and arbiter version is 02.04.01-003_BUILD05.fr. Please update.
[OK] Configuration seems valid
[OK] Connection to daemon is OK at port 7771
[WARN] Correct connection from arbiter "arbiter-vm3" ( and no time shift )
Tag list:
None [This poller will handle all untagged checks (the "None" tag)]
Talk to:
```

Conflit d'Arbiter sur un démon

Lors d'une mauvaise configuration, il arrive que 2 installations différentes de Shinken Entreprise utilisent les démons du même serveur.

Dans ce cas, plusieurs Arbiters envoient une configuration contradictoire au même démon. Dans ce cas, une erreur est affichée dans le healthcheck indiquant le conflit d'Arbiter.

Pour chaque Arbiter, le nom et l'adresse de l'Arbiter concerné sont également indiqués.

```
[WARN] Daemon version is: 02.04.01-005_BUILD02.fr
Arbiters conflicts:
[WARN] Arbiter: arbiter-master (http://172.16.0.199:7770/) This arbiter will expire in 100 seconds.
[WARN] Arbiter: arbiter-vm3 (http://vm3:7770/) This arbiter will expire in 7 seconds.
Talk to:
```

Décalage de temps entre 2 démons

Puisque Shinken Entreprise repose sur une architecture distribuée composée de plusieurs démons, la synchronisation de l'horloge entre les démons est un point important pour un bon fonctionnement du système.

Dans le healthcheck, si un démon ne possède pas la même heure que l'Arbiter, une erreur est affichée et indique le décalage horaire.

Dans l'exemple, le poller "*poller-domtom*" possède un décalage horaire de 4160 secondes avec l'Arbiter "*arbiter-vm3*".

```
[poller: poller-domtom]
[WARN] Correct connection from arbiter "arbiter-vm3" but a time shift of -4160 seconds
[OK] Configuration seems valid
[OK] Connection to daemon is OK at port 7771
[OK] Daemon version is: 02.04.01-003_BUILD07.fr
Tag list:
None [This poller will handle all untagged checks (the "None" tag)]
Talk to:
```

Un démon est configuré en tant que Spare

Un démon peut être configuré en tant que Spare. Un démon Spare est par défaut inactif, et devient actif lorsqu'un démon principal n'est plus disponible, afin d'assurer une continuité du service.

Pour pouvoir identifier facilement les démons Spare dans l'architecture, une mention "SPARE" est présente a coté du nom du démon.

Si le démon prend la main afin d'assurer la continuité de service, alors une mention additionnelle "RUNNING" est affichée.

```
- 172.16.0.2 (172.16.0.2):
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
[broker: broker-corse-spare] (SPARE)
OK: Configuration seems valid
OK: Connection to daemon is OK at port 7772
OK: Daemon version is: 02.04.01-003_BUILD07.fr
OK: Correct connection from arbiter "arbiter-vm3" ( and no time shift )
Talk to:

- 192.168.1.139 (192.168.1.139):
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
[scheduler: scheduler-spare] (SPARE) (RUNNING)
OK: Configuration seems valid
OK: Connection to daemon is OK at port 7768
OK: Daemon version is: 02.04.01-005_BUILD011.fr
OK: Correct connection from arbiter "arbiter-mas"
Modules:
OK: Name: MongoDBRetention Type: mong
```

Informations et détections dans le cadre de virtualisation Vmware

Il se peut que vous décidiez de virtualiser votre architecture Shinken, ou simplement quelques uns des démons, avec **VMWare**.

Si c'est le cas, une ligne sera ajoutée dans la sortie du "shinken-healthcheck" et précisera que votre satellite fonctionne sur une architecture VMWare.

Une vérification sera faite sur la présence ou non des "VMWare tools" sur votre VM (Machine virtuelle). Si ces derniers ne sont pas installés, alors une information sera affichée.

Nous vous conseillons de toujours avoir vos VM tools installés, et à jour, sur l'ensemble de votre parc de VM.

Enfin, si votre configuration de virtualisation ne permet pas à votre machine virtuelle une utilisation correcte des CPU physiques de son hôte via les VCPU que vous avez associés à celle ci, alors une information sera affichée. Vos VCPU n'arrivent pas à traiter assez rapidement toutes les demandes d'exécution.

La valeur "stolen CPU" en pourcentage est utilisée pour la détection. Si cette valeur dépasse 15%, alors un message de type "AT RISK" apparaît.

Si vous recevez ce message, plusieurs options sont possibles :

- Réduisez le nombre d'allocation de VCPU sur votre VM
- Augmentez vos ressources CPU pour vos VM
- Déplacer vos VM sur un autre serveur physique moins sollicité en terme CPU

```
Local daemons
[broker]
OK: Connection to daemon is OK at port 7772
OK: Daemon version is: 02.04.03-001_BUILD03.fr
OK: This satellite runs on a VMWare VM
OK: Correct connection from arbiter "arbiter-mas"
```

```
[broker: broker-master]
AT RISK: This satellite runs on a VMWare VM but the VMWare tools are not installed ; pl
OK: Configuration seems valid
OK: Connection to daemon is OK at port 7772
OK: Daemon version is: 02.04.00-release.fr
OK: Correct connection from arbiter "arbiter-master" ( and no time shift )
```

```
[scheduler: scheduler-master]
AT RISK: This satellite runs on a VMWare VM but the stolen
VCPU allocation on the VM to lighten supervisor load
OK: Configuration seems valid
OK: Connection to daemon is OK at port 7768
```

Un module a redémarré de manière imprévue

Certains démons de Shinken Entreprise utilisent des modules qui permettent d'étendre leur fonctionnalités. Il arrive que ces modules redémarent pour des raisons diverses et variées.

Lorsqu'un module a redémarré dans les 2 dernières heures, un avertissement est affiché pour le module en question, car il peut s'agir d'un problème récurrent et/ou potentiellement gênant. Un analyse des logs peut permettre d'avoir plus d'informations sur les erreurs du module.

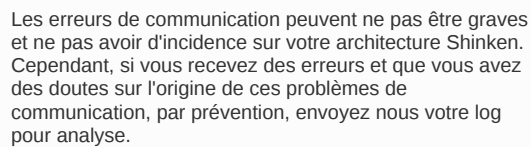
```
[scheduler: scheduler-web]
AT RISK: The scheduler started without loading retention
OK: Configuration seems valid
OK: Connection to daemon is OK at port 7768
OK: Daemon version is: 02.04.01-003_BUILD07.fr
OK: Correct connection from arbiter "arbiter-web" ( and no time shift )
Modules:
Name: MongoDBRetention Type: mongodb_retention
This module has restarted (3 times) in the last 120 minutes. A restart is removed from count after 120 minutes.
Last restart:
- 2017-04-28 16:33:48 : Connex starts a ssh tunnel for a mongodb connection to server 172.16.8.2 because your local ssh key (~/.ssh/id_rsa) is not authorized on the distant server. You can use the command "ssh-copy-id -i ~/.ssh/id_rsa shinkenuser@172.16.8.2" from this local s
> Permission denied (publickey,gssapi-keyex,gssapi-withmic,password).
No connection to MongoDB database.
```

Erreur de communication entre les démons

Les différents démons de Shinken Entreprise communiquent entre eux pour fonctionner. Il arrive que des erreurs de communications soient remarquées et génèrent des erreurs, qui sont alors notifiées dans le healthcheck dans la section du démon concerné.

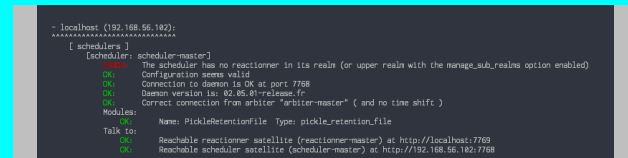
Il est à noter que ces erreurs ne sont pas forcément fatales au bon fonctionnement de Shinken Entreprise, mais qu'elles nécessitent une attention particulière.

```
[poller: poller-web]
Some API calls between daemons failed in the last 24 hours (2 errors). Please look at your daemon logs for more details about these errors.
OK: Configuration seems valid
OK: Connection to daemon is OK at port 7771
OK: Daemon version is: 02.04.01-003_BUILD07.fr
OK: Correct connection from arbiter "arbiter-web" ( and no time shift )
Req list: [this poller will handle all untagged checks (the "None" tag)]
Talk to: None
Reachable scheduler satellite (scheduler-web) at http://web.7768
```

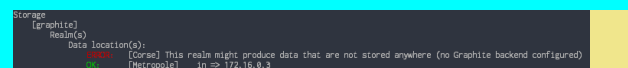


Les schedulers produisent des jobs à effectuer pour les pollers /reactionners, ainsi que des données pour les brokers (par exemple pour l'interface de visualisation). Ils ont donc besoins de ces autres daemons pour fonctionner.

 Si le scheduler est dans un sous royaume et qu'un poller /reactionner/broker est présent dans un royaume supérieur, ceci signifie que ce dernier n'a pas l'option `manage_sub_realms` activée.



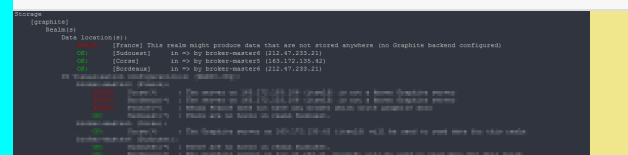
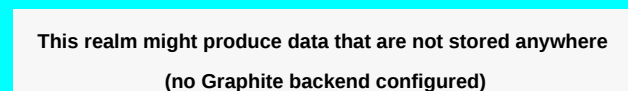
Dans une configuration avec plusieurs royaumes, il est plus difficile de vérifier que les données de météorologie sont bien écrites et peuvent être lues pour chaque royaume.



- Les erreurs d'écriture
- Les erreurs de lecture

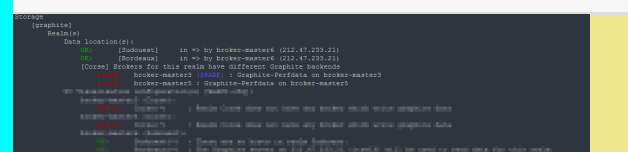
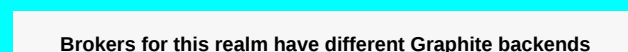
Cas 1 : Les données d'un royaume ne sont pas sauvegardé
This realm might produce data that are not stored anywhere (no Graphite backend configured)

Correction : Ajouter le module Graphite-Perfdata sur le broker



Exemple : Un broker et son spare dans le royaume Corse avec la configuration par défaut du module Graphite-Perfdata

```
define module {
    #===== Module identity =====
    # Module name. Must be unique
    module name Graphite-Perfdata-
```



```
# Module type (to load module code). Do not
edit.
module_type                               graphite_perfdata

##### Graphite address #####
# host:  graphite server address (ip or fqdn)
host                                       broker-master-3

# port:  tcp port of the graphite server
port                                      2003

}
```

```
modules          Simple-log, WebUI, sla,
Livestatus, Graphite-Perfdata-Corse
```

Exemple : Dans le fichier /etc/shinken/module/webi.cfg la configuration précise un royaume qui n'existe pas

```
graphite_backends      France:212.47.233.21, *:
127.0.0.1
```

```
manage_sub_realms      1
```

realm_store_only	Corse, Bordeaux
------------------	-----------------

[illegible]

```

# =====
# [utils]
# =====
def main():
    Data = load_data(100)

    # =====
    # [preprocessing]
    # =====
    X_train, X_test, y_train, y_test = train_test_split(
        Data, test_size=0.2, random_state=42)

    # =====
    # [model]
    # =====
    model = LogisticRegression()
    model.fit(X_train, y_train)

    # =====
    # [evaluation]
    # =====
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    print(f"Accuracy: {accuracy}")

if __name__ == "__main__":
    main()

```

[illegible][illegible]

```
graphite_backends      Corse:212.47.233.21, *:
127.0.0.1
```

Correction : Préciser un serveur qui gère les données de Corse :

```
graphite_backends      Corse:163.172.153.106, *:
127.0.0.1
```

Cas 5 : Le serveur graphite précisé dans la configuration de webui n'est pas un serveur graphite connu
The server on * is not a known Graphite server

Exemple : La configuration de webui précise que les données de Corse sont sur toto qui n'est pas un serveur graphite connu

graphite_backends	Corse:toto, *:127.0.0.1
-------------------	-------------------------

Correction : Préciser un serveur qui gère les données de Corse :

```
graphite_backends      Corse:163.172.153.106, *:
127.0.0.1
```

Enfin, les erreurs de graphites sont maintenant comptés par le module Webui et elle sont retournées dans les erreurs de lecture.

The server on * is not a known Graphite server

```
Server(s)
192.168.1.80:
Write connection status
OK success broker-prod
Read connection status
ERROR failure broker-prod - O
```