Rétention en base de donnée centralisée par royaume (Module MongodbRetention)

Fonctionnement de la rétention

Dans Shinken Entreprise, lorsque des éléments sont en supervision, des vérifications régulières sont effectuées sur les hôtes, clusters et checks.

Suite à ces vérifications, un statut (OK, Attention, Critique, Inconnu) ainsi que un ou plusieurs contextes (Flapping, Période de maintenance, Prise en compte) sont attribués à chaque élément.

Sans rétention, lorsque Shinken doit être redémarré (maintenance du serveur de supervision, ou bien mise à jour de Shinken), ces statuts et contextes sont perdus, et les éventuelles notifications déclenchées sur un état non voulu seront envoyées !

Activer la rétention permet de conserver les états des hôtes, clusters et checks entre les redémarrages de Shinken et ainsi bénéficier d'une vision claire de l'état des éléments supervisés à tout moment.

Cette rétention s'effectue au niveau du démon Scheduler qui est chargé d'ordonnancer la vérification des éléments et de récupérer et analyser les résultats de ces vérifications.

Données sauvegardées

Pour chaque élément (hôte, check ou cluster) activé dans la configuration, les données suivantes sont sauvegardées:

Type de donnée	Commentaire
Identifiant unique de l'élément	L'UUID est un champ interne à Shinken permettant d'identifier un élément (hôte, check ou cluster) de manière unique
Données d'ordonnancement	Date de la dernière et de la prochaine vérification
Statut actuel	Statut actuel de l'élément
Dernier changement de statut	Date du dernier changement de statut et statut précédent
Contexte	Indique si l'hôte est en Flapping, a une Prise en compte ou des périodes de maintenance. Dans le cas des Périodes de maintenance et des Prises en compte, l'auteur, date et commentaires sont également sauvegardées.
Résultat et résultat long	Résultat et résultat long de la dernière vérification
Contacts	Ensemble des contacts (identifiés par leur nom) qui ont reçu une notification concernant l'élément
Problèmes sources	Lorsque l'élément possède des liens avec d'autres éléments, lorsque cet élément est en erreur, l'identifiant unique des autres éléments affectés sont également sauvegardés. Aussi, si un élément en erreur a affecté l'élément actuel, l'identifiant unique de l'élément source du problème est sauvegardé

Configurer la rétention Mongodb

Le module MongodbRetention se charge de sauvegarder la rétention dans une base de données Mongo. L'avantage de ce type de rétention est qu'il peut, contrairement à la rétention par fichiers plats, être utilisé dans un environnement distribué avec plusieurs Schedulers. Plus de détails sont disponibles sur les cas d'utilisations de ce type de rétention dans la page Configurer la rétention des données.

Pour l'utiliser, il faut activer ce module sur le Scheduler pour lequel on veut sauvegarder la rétention.

Cette configuration s'effectue dans le fichier de configuration du Scheduler concerné. Dans Shinken Entreprise, la définition des Schedulers se trouve **/etc/shinken/schedulers/**.

/etc/shinken/schedulers/mon_scheduler.cfg

```
define scheduler {
...
...
...
    #======== Modules to enable for this daemon =========
    # Available:
    # - PickleRetentionFile : (if you have only one scheduler into a realm) save retention data (element
state and scheduling) into a file
    # - MongodbRetention    : (if you have more than one scheduler into a realm) save retention data (element
state and scheduling) into a mongodb database
    modules                 MongodbRetention
...
...
...
}
```

Une fois la rétention Mongo activée sur les Schedulers concernés, il faut modifier l'URI de la base Mongo pour pointer vers l'adresse de la base de données qui hébergera les données de rétention.

L'installation de Shinken comporte une installation de Mongo. Il est donc possible d'utiliser un serveur Shinken comme serveur utilisé pour la rétention.

Un serveur externe peut également être utilisé pour sauvegarder la rétention.

Pour se connecter au serveur Mongo utilisé pour la rétention, 2 méthodes sont disponibles:

- **Connexion directe**: Par défaut, mais non sécurisée.
- **Tunnel SSH**: Shinken se connecte au serveur Mongo au travers d'un module SSH pour plus de sécurité

Connexion directe au serveur Mongo

Par défaut, le module de rétention se connecte de manière directe au serveur Mongo pour y lire et écrire les données de rétention.

Dans la configuration du module de retention, on sait que la connexion se fait de manière directe lorsque le paramètre "use_ssh_tunnel" est à 0.

/etc/shinken/modules/retention-mongodb.cfg

```
define module {


    #======== Module identity =========
    # Module name. Must be unique
    module_name     MongodbRetention
    ...
    use_ssh_tunnel  0

...

}
```

Cette méthode de connexion a pour avantage d'être facile à configurer au niveau de Shinken. Par contre, elle oblige à permettre l'accès à la base Mongo au monde extérieur, et donc s'exposer à des problèmes de sécurité.

La sécurisation de la base Mongo est bien sur toujours possible (voir Sécurisation des connexions aux bases MongoDB) mais bien plus complexe à mettre en place. La méthode de connexion par SSH est donc préférable pour des raisons pratiques et de sécurité.

Connexion par SSH au serveur Mongo

Le module de rétention peut également se connecter par tunnel SSH au serveur Mongo, pour des raisons de sécurité.

- Dans la configuration du serveur Mongo (/etc/mongod.conf), assurez-vous que le paramètre "*bind_ip*" est positionné pour n'écouter que sur l'interface locale:

```
bind_ip=127.0.0.1
```

- Depuis le serveur hébergeant le Scheduler, assurez-vous que les clés publiques SSH de l'utilisateur lançant le daemon (par défaut "shinken") sont autorisées sur le serveur hébergeant Mongo :
  - Connectez-vous avec le user lançant le démon sur le serveur Shinken
  - Générez la paire de clés SSH si nécessaire
  - Copiez la clé publique sur le serveur mongo

```
root@serveur_shinken # su - shinken
shinken@serveur_shinken $ ssh-keygen
shinken@serveur_shinken $ ssh-copy-id user_distant@serveur_mongo
[...]
shinken@serveur_shinken $ ssh user_distant@serveur_mongo
user_distant@serveur_mongo $
```

Note

Si vous avez un serveur qui héberge à la fois le démon Scheduler et la base MongoDB, il vous faudra également appliquer ces commandes pour autoriser l'utilisateur shinken à se connecter automatiquement sur lui même en SSH

- Modifiez la configuration du module de rétention Mongo
  - le paramètre "*use_ssh_tunnel*" doit être positionné à 1
  - le paramètre "*use_ssh_retry_failure*" permet de spécifier le nombre supplémentaire de tentatives lors de l'établissement du tunnel SSH si ce dernier n'arrive pas à être établi.
  - le paramètre "*ssh_user*" doit être positionné au user utilisé pour se connecter au serveur mongo (user_distant dans l'exemple précédent)
  - le paramètre "*ssh_keyfile*" doit pointer vers la clé ssh privée sur le serveur Shinken (par défaut ~/.ssh/id_rsa)

/etc/shinken/modules/retention-mongodb.cfg

```
define module {


    #======== Module identity =========
    # Module name. Must be unique
    module_name     MongodbRetention

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


    #======== Mongodb connection =========
    # uri: to connect the mongodb server
    uri             mongodb://ip_du_serveur/?safe=false
    use_ssh_tunnel  1
        use_ssh_retry_failure        1
    ssh_user        user_distant
    ssh_keyfile     /chemin/vers/cle/ssh (par defaut ~/.ssh/id_rsa)

    # database: which mongodb database to use
    database        shinken

    # Advanced option if you are running a cluster mongodb environnement
    # replica_set

}
```

- Vérifiez la configuration
  - Redémarrez l'arbiter
  - Lancez shinken-healthcheck, qui, en cas de problème, affichera des messages d'erreur dans la section "scheduler" détaillant le problème rencontré

Il se peut également que plusieurs royaumes veuillent définir une rétention Mongo sur un serveur différent pour chaque royaume. Dans ce cas, il fau faire plusieurs définitions de module de rétention.

- Le module_type sera identique, tandis que le reste de la configuration du module pourra changer.
- Il faudra ensuite, dans la configuration du Scheduler, spécifier le nom du module approprié.

Important

Même si ce n'est pas obligatoire, nous vous conseillons de faire une fichier séparé par définition de module nommé du nom du module de rétention (dans un but de clarté de votre configuration)

Ne pas utiliser "localhost" ou "127.0.0.1" comme URI de la base Mongo lorsqu'il y a plusieurs Schedulers dans le même royaume. Des explications détaillées sur ce problème sont présentes dans la page Configurer la rétention des données.

Options de connexions

Dans certains cas, il se peut que la connexion à mongo ai besoin de plusieurs tentatives (Ex. : coupures réseaux, sauvegarde externe de la base en cours, ...)

Afin de déterminer les options pour se reconnecter il faut utiliser les options suivantes :

- *Le paramètre "mongo_max_connections_retry"* permet de déterminer le nombre d'essai de connexions maximum qui sera tenté avant d'échouer. Par défaut, cette valeur est de 3
- Le paramètre "*mongo_wait_before_retry*" permet de définir le temps d'attente entre deux tentatives de connexions. La valeur par défaut est de 1 seconde.

/etc/shinken/modules/retention-mongodb.cfg

```
define module {


    #======== Module identity =========
    # Module name. Must be unique
    module_name     MongodbRetention

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

        ...
    # Number of retry if mongo is not acessible
    # Default value : 3
    mongo_max_connections_retry    3
    # Delay before next retry
    # Default value : 1s
    mongo_wait_before_retry 1

...

}
```

Important

Si ces options ne sont pas indiquer, les valeurs par défaut seront utilisées.

La durée de la sauvegarde de la rétention ne peut excéder le délai d'extinction des démons (fixé à 120 secondes) quelque soit la valeur des options précédemment définies.

Détails de l'emplacement des données de rétention

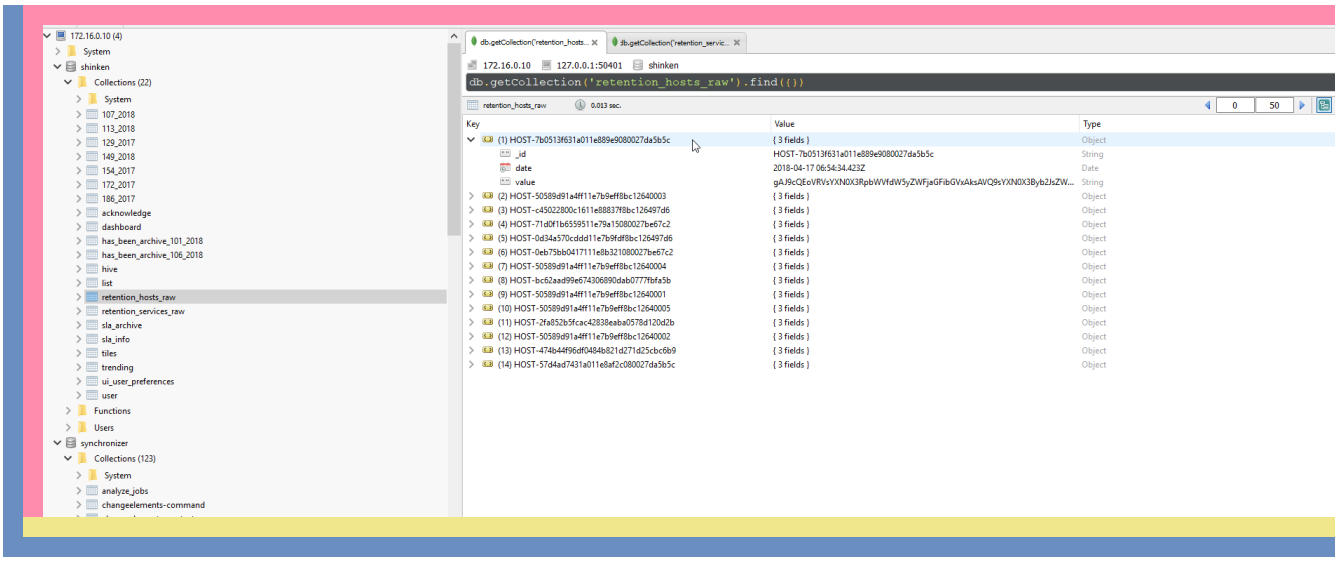
Les données de rétention via le module MongodbRetention sont donc sauvegardées via le moteur de base de données Mongodb.

La base utilisée pour la rétention est la base **shinken**.

Afin de distinguer les hôtes des checks, deux collections sont utilisées :

- pour les hôtes, la collection **retention_hosts_raw**
- pour les checks, la collection **retention_services_raw**

Voici la visualisation des collections via l'utilitaire RoboMongo permettant de se connecter aux bases Mongodb :


172.16.0.10 (4)
System
shinken
Collections (22)
System
107_2018
113_2018
129_2017
149_2018
154_2017
172_2017
186_2017
acknowledge
dashboard
has_been_archive_101_2018
has_been_archive_106_2018
hive
list
retention_hosts_raw
retention_services_raw
sla_archive
sla_info
tiles
trending
ui_user_preferences
user
Functions
Users
synchronizer
Collections (123)
System
analyze_jobs
changeelements-command
db.getCollection('retention_hosts...
db.getCollection('retention_servic...
172.16.0.10
127.0.0.1:50401
shinken
db.getCollection('retention_hosts_raw').find({})
retention_hosts_raw
0.013 sec.
0
50
Key
Value
Type
(1) HOST-7b0513f631a011e889e9080027da5b5c
{ 3 fields }
Object
_id
HOST-7b0513f631a011e889e9080027da5b5c
String
date
2018-04-17 06:54:34.423Z
Date
value
gAJ9cQEoVRVsYXN0X3RpbWVfdW5yZWFjaGFibGVxAksAVQ9sYXN0X3Byb2JsZW...
String
(2) HOST-50589d91a4ff11e7b9eff8bc12640003
{ 3 fields }
Object
(3) HOST-c45022800c1611e88837f8bc126497d6
{ 3 fields }
Object
(4) HOST-71d0f1b6559511e79a15080027be67c2
{ 3 fields }
Object
(5) HOST-0d34a570cddd11e7b9fdf8bc126497d6
{ 3 fields }
Object
(6) HOST-0eb75bb0417111e8b321080027be67c2
{ 3 fields }
Object
(7) HOST-50589d91a4ff11e7b9eff8bc12640004
{ 3 fields }
Object
(8) HOST-bc62aad99e674306890dab0777fbfa5b
{ 3 fields }
Object
(9) HOST-50589d91a4ff11e7b9eff8bc12640001
{ 3 fields }
Object
(10) HOST-50589d91a4ff11e7b9eff8bc12640005
{ 3 fields }
Object
(11) HOST-2fa852b5fcac42838eaba0578d120d2b
{ 3 fields }
Object
(12) HOST-50589d91a4ff11e7b9eff8bc12640002
{ 3 fields }
Object
(13) HOST-474b44f96df0484b821d271d25cbc6b9
{ 3 fields }
Object
(14) HOST-57d4ad7431a011e8af2c080027da5b5c
{ 3 fields }
Object