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

Sommaire

Concept

Le module **MongodbRetention** est un module de rétention qui permet de stocker dans une base **mongoDB** les différents statuts et contextes des éléments de Shinken.

- Après un redémarrage, les éléments récupèreront leurs statuts à partir de ce qui a été stocké par le module.
- Pour plus d'explication sur le principe de la rétention (*voir le chapitre Rappel du fonctionnement de la rétention*).

Activation du module

Les modules de type `"mongodb_retention"` sont des modules qui doivent être activés sur un démon de type `"scheduler"` qu'on appellera le **démon**.

- L'activation du module s'effectue en ajoutant le **nom** du module dans la configuration du **démon**.
  - Pour cela, il faut ouvrir le fichier de configuration du **démon** (*de type `"scheduler"`*), et ajouter dans le paramètre **modules**, le nom du module de type `"mongodb_retention"`.

- Il est possible de faire plusieurs modules de type `"mongodb_retention"`.
  - Cela permet, **par exemple**, d'avoir des configurations différentes en fonction des royaumes.

- Contraintes :
  - Activable uniquement sur un démon de type `"scheduler"` (*voir la page* Le Scheduler).
  - Il ne peut y avoir qu'un seul module de type `"mongodb_retention"` sur un démon de type `"scheduler"`.
  - Si un module de type `"mongodb_retention"` est activé sur un démon de type `"scheduler"`, il ne faut pas d'autre module de type dessus.
    - Avoir un module de rétention d'un autre type n'est pas un soucis (*par exemple `"pickle_retention_file"`*) et peut être utile dans le cas de migration de type de retention (*voir la page La rétention des données des Schedulers*).
  - Tous les démons de type `"scheduler"` d'un royaume (*et de ses sous-royaumes*) doivent avoir un module de type `"mongodb_retention"`, sinon l'Arbiter refusera la configuration et ne démarrera pas.

Pour prendre en compte le changement de configuration, il faut redémarrer l'Arbiter :

```
service-shinken-arbiter restart
```

Exemple d'activation du module nommé "MongodbRetention" sur le démon nommé "scheduler-master" (configuration livrée par défaut par Shinken)

L'exemple suivant

- active le module "MongodbRetention" ,
- sur le démon "scheduler-master", dont la configuration est dans le fichier **/etc/shinken/schedulers/scheduler-master.cfg**.

Modification dans le fichier du module **/etc/shinken/schedulers/scheduler-master.cfg** :

```
define scheduler {
    [...]
    modules                  Module 1, Module 2, Module 3, MongodbRetention
    [...]
}
```

Puis redémarrage de l'Arbiter

```
service-shinken-arbiter restart
```

Créer un nouveau module de type mongodb_retention

Pour pouvoir configurer un module de type "mongodb_retention", il faut faire un nouveau fichier de configuration grâce au fichier d'exemple fourni par défaut.

- Pour commencer, il faut choisir le nom du nouveau module.
  - Pour l'exemple, on l'appelle "Mon-Module-Mongodb-Retention".
  - Remplacer dans l'exemple le mot "Mon-Module-Mongodb-Retention" par le nom qui a été choisi.
- Puis il faut créer le fichier de configuration :
  - Copier le fichier de définition du module d'exemple : **/etc/shinken-user-example/configuration/daemons/schedulers/modules/retention-mongodb/retention-mongodb-example.cfg** dans le répertoire de définition des modules **/etc/shinken/modules/** .
    (*Exemple : /etc/shinken/modules/retention-mongodb__Mon-Module-Mongodb-Retention.cfg*)

    ```
    cp /etc/shinken-user-example/configuration/daemons/schedulers/modules/retention-mongodb
    /retention-mongodb-example.cfg /etc/shinken/modules/retention-mongodb__Mon-Module-Mongodb-
    Retention.cfg
    ```

- Ensuite, il faut modifier le fichier nouvellement créé pour configurer le nouveau module.
  - Il faut vérifier que le fichier appartienne à l'utilisateur shinken et qu'il possède le droit d'édition. Si ce n'est pas le cas, il faut effectuer les commandes suivantes :

    ```
    chown -R shinken:shinken /etc/shinken/modules/retention-mongodb__Mon-Module-Mongodb-Retention.
    cfg
    chmod u+w /etc/shinken/modules/retention-mongodb__Mon-Module-Mongodb-Retention.cfg
    ```

  - On change le nom du module en "Mon-Module-Mongodb-Retention" dans le fichier **/etc/shinken/modules/retention-mongodb__Mon-Module-Mongodb-Retention.cfg**

    ```
    ...
            #  Module name [ Must be unique ]
    [ MANDATORY ]
        #


        module_name                                          Mon-Module-Mongodb-Retention
    ...
    ```

- Ensuite, il faut ajouter le nouveau module dans le démon de type "scheduler" correspondant.
  - Dans notre exemple, on ajoute le module "Mon-Module-Mongodb-Retention" au démon "scheduler-master" définie dans le fichier **/etc/shinken/schedulers/scheduler-master.cfg**

```
define module {
        [...]
        modules                                                 Module 1, Module 2, Module 3,
Mon-Module-Mongodb-Retention
        [...]
}
```

- Puis pour finir, il faut redémarrer l'Arbiter pour que le Broker puisse prendre en compte ce nouveau module.

```
service-shinken-arbiter restart
```

Rappel du 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 qu'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 (*Voir la page* Le Scheduler).

Pour plus d'informations sur le fonctionnement de la rétention dans Shinken voir La rétention des données des Schedulers (*Voir la page* La rétention des données des Schedulers).

Les données sauvegardées

Pour chaque élément activé dans la configuration (*Hôte, Check ou Cluster*), les données suivantes sont sauvegardées entre autres :

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, à 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 commentaire sont également sauvegardés.
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 est également sauvegardé. Aussi, si un élément en erreur a affecté l'élément actuel, l'identifiant unique de l'élément source du problème est sauvegardé.

Configuration

La configuration du module se trouve par défaut dans le fichier **/etc/shinken/modules/retention-mongodb.cfg** .

- Un exemple dans **/etc/shinken-user-example/configuration/daemons/schedulers/modules/retention-mongodb/retention-mongodb-example.cfg**.

Exemple de fichier de configuration

```
# CFG_FORMAT_VERSION 1 ( SHINKEN : DON'T TOUCH THIS LINE )

#===============================================================================
# MongodbRetention
#===============================================================================
# Daemons that can load this module:
#  - Scheduler
# This module save scheduler retention data (element state and scheduling)
# into a mongodb server
#===============================================================================

define module {

    #  #
    #      MODULE IDENTITY      #
    #  #

    #  Module name [ Must be unique ]                                                [ MANDATORY ]
    #
    module_name                                     MongodbRetention

    #  Module type [ Do not edit ]                                                   [ MANDATORY ]
    #
    module_type                                     mongodb_retention

    #  #
    #      DATABASE CONNECTION      #
    #  #

    #   MongoDB parameters   #

    #  MongoDB uri definition . You can find the mongodb uri syntax at
    #  https://docs.mongodb.com/manual/reference/connection-string/
    #
    #           Default : mongodb://localhost/?w=1&fsync=false
    #
    # mongodb_retention__database__uri                    mongodb://localhost/?w=1&fsync=false

    #  Which database contains retention data
    #
    #           Default : shinken
    #
    # mongodb_retention__database__name                   shinken

    #  username/password to authenticate to MongoDB.
    #  Both parameters must be provided for authentication to function correctly.
    #
    # scheduler__module_mongodb_retention__database__username

    #
    # scheduler__module_mongodb_retention__database__password

    #  Allow localhost MongoDB uri.
    #  By security, the localhost URI is banned as it is a configuration error if you have multiple
    #  Schedulers in the same realm ( all Schedulers must save retention in the same database ).
    #  But in case, your retentions are using a MongoDB cluster,
    #  activate this option to bypass the security ( targeting the local mongos )
    #
    #           ...     : Enable  => 1 ( allow localhost mongo uri )
    #           Default : Disable => 0 ( localhost will be forbidden if there are multiple Schedulers )
    #
    # mongodb_retention__database__bypass_banning_localhost_uri 0

    #  SSH tunnel activation to secure your mongodb connection
    #  That will allow all mongodb to be encrypted & authenticated with SSH
    #
    #           ...     : Enable  => 1 ( enable ssh tunnel )
    #           Default : Disable => 0 ( disable ssh tunnel )
```

```
#
# mongodb_retention__database__use_ssh_tunnel         0

#  If the SSH connection goes wrong, then retry use_ssh_retry_failure time before_shinken_inactive
#
#           Default : 1 ( try )
#
# mongodb_retention__database__use_ssh_retry_failure  1

#  SSH user to connect to the mongodb server.
#
#           Default : shinken
#
# mongodb_retention__database__ssh_user               shinken

#  SSH keyfile to connect to the mongodb server.
#
#           Default : ~shinken/.ssh/id_rsa
#
# mongodb_retention__database__ssh_keyfile            ~shinken/.ssh/id_rsa

#  SSH Timeout used to test if the SSH tunnel is viable or not, in seconds.
#
#           Default : 10 ( seconds )
#
# mongodb_retention__database__ssh_tunnel_timeout     10

#   AutoReconnect Management   #

#  When MongoDB require you to reconnect ( For example, It can occur when a new PRIMARY is elected
#  in a MongoDB cluster ), it will raised the MongoDB AutoReconnect exception.
#
#  How many try to reconnect before module go in error
#
#           Default : 5 ( try )
#
# mongodb_retention__database__retry_connection_X_times_before_considering_an_error 5

#  Time between each try
#
#           Default : 5 ( seconds )
#
# mongodb_retention__database__wait_X_seconds_before_reconnect 5

#  NOTE: Change these values only if you have a MongoDB cluster and you change the
#        heartbeatTimeoutSecs of your MongoDB replica set
#        The value of mongodb_retention__database__wait_X_seconds_before_reconnect *
#        mongodb_retention__database__retry_connection_X_times_before_considering_an_error must be
#        higher than heartbeatTimeoutSecs in the rs.conf(); of your MongoDB replica set.

#  #
#      WORKER CONFIGURATION      #
#  #

#  The retention save will spawn process workers (max to 4) and such process can fail (timeout)
#
#  Worker timeout: Global time for your workers to work. If a worker still runs after worker_timeout
#  seconds (and so numerous try)), then it will be killed and the error will be raised
#  into your monitoring
#
#           Default : 120 ( seconds )
#
# worker_timeout                                      120

#  Worker try timeout: Time for a one try data save. If one worker process still runs after
#  worker_one_try_timeout seconds, it will be killed and a new worker process will be spawn
#  to replace it
#  NOTE: it must be lower than the worker_timeout
#
#           Default : 30 ( seconds )
#
```

```
# worker_one_try_timeout                              30

#  Max number of workers: if you want to limit the number of workers launched, you can change
#  this parameter.By default the number of workers will be the number of CPUs
#  but no more than max_number_of_workers
#
#           Default : 4 ( workers )
#
# max_number_of_workers                               4

#  #
#      MEMORY PROTECTION      #
#  #

#  Are the module worker process are waiting for enough memory to be available before being launch.
#
#           ...     : 0 ( Disable )
#           Default : 1 ( Enable )
#
# scheduler__retention_mongo__enable_sub_processes_memory_usage_protection 1

#  The sub process memory usage protection can have a system reserved memory that won't be used by
#  theses sub process when launched
#
#           Default : 0 ( no reserved memory )
#        Example : 10  (means 10% of the total memory is reserved for the system)
#
# scheduler__retention_mongo__sub_process_memory_usage_system_reserved_memory 0

#  If a sub process cannot be started because of the protection, how many seconds it will be retry
#  and wait that the system memory is freed until it fail to start
#
#           Default : 5 ( seconds )
#
# scheduler__retention_mongo__sub_processes_memory_usage_protection_max_retry_time 5

#  #
#      INTERNAL OPTIONS      #
#  #

#        /!\ INTERNAL : DO NOT EDIT FOLLOWING PARAMETER WITHOUT YOUR DEDICATED SUPPORT
#
#  Number of day we conserve retention data, after this time, we will delete data.
#
#           Default : 7 ( days )
#
# nb_of_max_retention_day                             7

#  Maximum number of elements load in one chunk pass.
#
#           Default : 1000
#
# size_chunk_to_load                                  1000

#  Maximum number of elements delete in one chunk pass.
#
#           Default : 1000
#
# size_chunk_to_delete                                1000

#  Timeout for the execution of each chunk retention request.
#
#           Default : 300
#
# scheduler__retention_mongo__load_retention_chunk_timeout 300

#  Elements whose serialized size exceeds this threshold will generate a warning
#
#           Default : 10000 ( Kilobytes )
#
# scheduler__retention_mongo__oversized_element_warning_threshold__size 10000
```

```
    #  Elements whose serialized size exceeds this threshold will generate an error
    #
    #           Default : 16000 ( Kilobytes )
    #
    # scheduler__retention_mongo__oversized_element_error_threshold__size 16000

}
```

Détails des sections composant le fichier de configuration

Identification du module

Il est possible de définir plusieurs instances de module de type "MongodbRetention" dans l'architecture Shinken .

- Chaque instance devra avoir un nom unique.

> Il est conseillé de n'avoir plusieurs modules de rétention que dans le cas d'une migration.
>
> *Exemple : passage d'une base de données **base1** à une **base2**, on aura un module MongodbRetention avec la **base1**, un autre avec la **base2**, on désactivera alors le premier module pour le prochain redémarrage.*

Nom	Type	Unité	Défaut	Description
`module_name`	Texte	---	**MongodbRetention**	**Shinken conseille de choisir un nom en fonction de l'utilisation du module pour que la configuration soit simple à maintenir.** Doit être unique.
`module_type`	Texte	---	**mongodb_retention**	Ne dois pas être modifié.

Accès à la base MongoDB

Cette configuration s'effectue dans le fichier de configuration du module.

Pour se connecter à la base MongoDB utilisée pour le stockage des données, 2 méthodes sont disponibles :

- **Connexion directe** : Par défaut, mais non sécurisée.
- **Tunnel SSH** : le module se connecte à la base MongoDB au travers d'un module SSH pour plus de sécurité

Configuration de l'URI de connexion et de l'authentification par mot de passe

```
    #  #
    #      DATABASE CONNECTION      #
    #  #

    #   MongoDB parameters   #

    #  MongoDB uri definition . You can find the mongodb uri syntax at
    #  https://docs.mongodb.com/manual/reference/connection-string/
    #
    #           Default : mongodb://localhost/?w=1&fsync=false
    #
    # mongodb_retention__database__uri                    mongodb://localhost/?w=1&fsync=false

    #  Which database contains retention data
    #
    #           Default : shinken
    #
    # mongodb_retention__database__name                   shinken

    #  username/password to authenticate to
MongoDB.
    #  Both parameters must be provided for authentication to function correctly.
    #
    # scheduler__module_mongodb_retention__database__username

    #
    # scheduler__module_mongodb_retention__database__password

    #  Allow localhost MongoDB uri.
    #  By security, the localhost URI is banned as it is a configuration error if you have multiple
    #  Schedulers in the same realm ( all Schedulers must save retention in the same database ).
    #  But in case, your retentions are using a MongoDB cluster,
    #  activate this option to bypass the security ( targeting the local mongos )
    #
    #           ...     : Enable  => 1 ( allow localhost mongo uri )
    #           Default : Disable => 0 ( localhost will be forbidden if there are multiple Schedulers )
    #
    # mongodb_retention__database__bypass_banning_localhost_uri 0
```

Nom	Type	Unité	Défaut	Description
`mongodb_retention__database__uri`	Texte	URL	**mongodb://localhost/?w=1&fsync=false**	Trouver la syntaxe de l'URI de MongoDB à l'adresse https://docs.mongodb.com/manual/reference/connection-string/
`mongodb_retention__database__name`	Texte	---	**shinken**	Nom de la base de données où sont stockées les données de rétention.
`scheduler__module_mongodb_retention__database__username`	Texte	---		Utilisateur pour l'authentification avec mot de passe à la base MongoDB. Utile uniquement si l'activation par mot de passe a été activé (*voir la page* MongoDB - activation de l'authentification par mot de passe)

`scheduler__module_mongodb_retention__database__password`	Texte	---		Mot de passe de l'utilisateur utilisé pour l'authentification avec mot de passe à la base MongoDB. Utile uniquement si l'activation par mot de passe a été activé (*voir la page* MongoDB - activation de l'authentification par mot de passe)
`mongodb_retention__database__bypass_banning_localhost_uri`	Booléen	---	**0**	Cette option permet d'autoriser l'URI MongoDB localhost. Par sécurité, l'URI localhost est interdit, car il s'agit d'une erreur de configuration si il y a plusieurs Scheduler dans le même royaume (*tous les Schedulers doivent sauvegarder la rétention dans la même base de données*). Dans le cas où le module de rétentions utilise un cluster MongoDB, il est possible d'activer cette option pour contourner cette sécurité. Valeur possible : • 1 : Autoriser localhost comme URI de Mongo. • 0 : Localhost sera interdit s'il y a plusieurs Schedulers dans le même royaume.

Connexion directe au serveur MongoDB

Par défaut, le module se connecte de manière directe à la base MongoDB pour y lire et écrire les données.

Dans la configuration du module, le paramètre "`use_ssh_tunnel`" fixé à 0 informe que la connexion est directe.

- 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 MongoDB au monde extérieur, et donc expose à des problèmes de sécurité.

La sécurisation de la base MongoDB est bien sûr toujours possible, mais bien plus complexe à mettre en place.

(*Voir la page Sécurisation des connexions aux bases MongoDB*)

Connexion par SSH au serveur Mongo

Par défaut, le module se connecte de manière directe à la base MongoDB pour y lire et écrire les données.

Dans la configuration du module, le paramètre "`use_ssh_tunnel`" fixé à 0 informe que la connexion est directe.

- 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 MongoDB au monde extérieur, et donc expose à des problèmes de sécurité.

La sécurisation de la base MongoDB est bien sûr toujours possible, mais bien plus complexe à mettre en place (*Voir la page Sécurisation des connexions aux bases MongoDB*).

La méthode de connexion par SSH est ainsi préférable pour des raisons pratiques et de sécurité.

```
#  SSH tunnel activation to secure your mongodb connection
#  That will allow all mongodb to be encrypted & authenticated with SSH
#
#           ...     : Enable  => 1 ( enable ssh tunnel )
#           Default : Disable => 0 ( disable ssh tunnel )
#
# mongodb_retention__database__use_ssh_tunnel         0

#  If the SSH connection goes wrong, then retry use_ssh_retry_failure time before_shinken_inactive
#
#           Default : 1 ( try )
#
# mongodb_retention__database__use_ssh_retry_failure  1

#  SSH user to connect to the mongodb server.
#
#           Default : shinken
#
# mongodb_retention__database__ssh_user               shinken

#  SSH keyfile to connect to the mongodb server.
#
#           Default : ~shinken/.ssh/id_rsa
#
# mongodb_retention__database__ssh_keyfile            ~shinken/.ssh/id_rsa

#  SSH Timeout used to test if the SSH tunnel is viable or not, in seconds.
#
#           Default : 10 ( seconds )
#
# mongodb_retention__database__ssh_tunnel_timeout     10
```

Le module peut également se connecter par tunnel SSH à la base MongoDB, pour des raisons de sécurité.

En effet, le paramétrage de MongoDB permet de définir sur quelle interface réseau ce dernier écoute les requêtes.
En autorisant seulement l'interface réseau avec l'adresse 127.0.0.1, cela évite d'ouvrir la base au monde extérieur.

Dans la configuration de la base MongoDB (*/etc/mongod.conf*), le paramètre " *bind_ip* " doit être positionné pour n'écouter que sur l'interface locale :

- `bind_ip= 127.0 . 0.1`

Dans cette configuration, la base MongoDB écoute que sur l'interface réseau local, pour que le module se connecte, il faut passer par un tunnel SSH.
Pour ce faire, activez les options suivantes :

Nom	Type	Unité	Défaut	Description
`mongodb_retention__database__use_ssh_tunnel`	Booléen	---	**0**	• 1 : Connexion par tunnel SSH. • 0 : Connexion directe.
`mongodb_retention__database__use_ssh_retry_failure`	Entier	nombre d'essais	**1**	Spécifie le nombre supplémentaire de tentatives lors de l'établissement du tunnel SSH si ce dernier n'arrive pas à être établi.

`mongodb_retention__database__ssh_user`	Texte	utilisateur unix	**shinken**	L'utilisateur avec lequel le tunnel sera établi.
`mongodb_retention__database__ssh_keyfile`	Texte	chemin de fichier	**~shinken/.ssh/id_rsa**	La clé SSH privée présente sur le serveur Shinken qui sera utilisé pour établir le tunnel.
`mongodb_retention__database__ssh_tunnel_timeout`	Entier	secondes	**10**	Spécifie le timeout en secondes de la vérification du tunnel SSH avant que la connexion vers MongoDB soit effectuée.

Pour configurer les clés SSH à utiliser, voir la page Création automatique et gestion de la clé SSH de l'utilisateur shinken.

Gestion de l'auto-reconnexion

```
#   AutoReconnect Management   #

#  When MongoDB require you to reconnect ( For example, It can occur when a new PRIMARY is elected
#  in a MongoDB cluster ), it will raised the MongoDB AutoReconnect exception.
#
#  How many try to reconnect before module go in error
#
#           Default : 5 ( try )
#
# mongodb_retention__database__retry_connection_X_times_before_considering_an_error 5

#  Time between each try
#
#           Default : 5 ( seconds )
#
# mongodb_retention__database__wait_X_seconds_before_reconnect 5

#  NOTE: Change these values only if you have a MongoDB cluster and you change the
#        heartbeatTimeoutSecs of your MongoDB replica set
#        The value of mongodb_retention__database__wait_X_seconds_before_reconnect *
#        mongodb_retention__database__retry_connection_X_times_before_considering_an_error must be
#        higher than heartbeatTimeoutSecs in the rs.conf(); of your MongoDB replica set.
```

La reconnexion automatique permet au module de se reconnecter à Mongo dans le cas où :

- Il y a une perte de connexion suite à un problème réseau ou à un redémarrage de mongo
- Dans le cas de l'utilisation d'un cluster MongoDB, lorsque le membre Primaire devient inaccessible, une nouvelle élection est déclenchée, ce qui provoque une coupure temporaire de l'accès à la base.

Définitions

Primaire : nom de MongoDB pour désigner un serveur maître, le serveur sur lequel il est possible de faire des requêtes d'écriture dans la base.

Élection : processus de MongoDB pour choisir un nouveau membre Primaire si le membre Primaire devient inaccessible

Voir la page Haute disponibilité de la base MongoDB (mise en place d'un cluster)

Dans le but de ne pas interrompre le service lorsque l'un de ces cas survient, le module "`mongodb`" va se reconnecter automatiquement.
Pour cela, il va faire un nombre d' essais égal au paramètre "mongodb__database__retry_connection_X_times_before_considering_an_error " avec une pause de X secondes entre chaque essai (*correspondant au paramètre* `"mongodb__database__wait_X_seconds_before_reconnect"`) .

Par défaut pour MongoDB, le temps maximum avant qu'un membre Primaire soit considéré comme indisponible et qu'une nouvelle élection ait lieu est de 10 secondes.
Voir : " heartbeatTimeoutSecs" donné par la commande rs . conf (); dans un shell de MongoDB.

Nom	Type	Unité	Défaut	Description
`mongodb_retention__database__retry_connection_X_times_before_considering_an_error`	Entier	Nombres d'essais	5	Nombre d'essais de reconnexion à la base.
`mongodb_retention__database__wait_X_seconds_before_reconnect`	Entier	Secondes	5	Temps entre chaque essai en seconde.

Les valeurs par défauts du fichier laissent 25 secondes, ce qui est amplement suffisant avec la configuration par défaut de MongoDB.

Il est conseillé de ne pas modifier ces valeurs.

Utilisation des workers

```
#  #
#      WORKER CONFIGURATION      #
#  #

#  The retention save will spawn process workers (max to 4) and such process can fail (timeout)
#
#  Worker timeout: Global time for your workers to work. If a worker still runs after worker_timeout
#  seconds (and so numerous try)), then it will be killed and the error will be raised
#  into your monitoring
#
#           Default : 120 ( seconds )
#
# worker_timeout                                  120

#  Worker try timeout: Time for a one try data save. If one worker process still runs after
#  worker_one_try_timeout seconds, it will be killed and a new worker process will be spawn
#  to replace it
#  NOTE: it must be lower than the worker_timeout
#
#           Default : 30 ( seconds )
#
# worker_one_try_timeout                          30

#  Max number of workers: if you want to limit the number of workers launched, you can change
#  this parameter.By default the number of workers will be the number of CPUs
#  but no more than max_number_of_workers
#
#           Default : 4 ( workers )
#
# max_number_of_workers                           4
```

Afin de répartir la tâche de sauvegarde de la rétention sur plusieurs processus, le module utilise des **workers** qui travailleront en parallèle.

Il est possible de les configurer via les paramètres suivants :

Nom	Type	Unité	Défaut	Description
`worker_timeout`	Entier	secondes	**120**	Temps maximum d'exécution d'un worker. Si ce temps est dépassé, le worker est éteint et une erreur est remontée. Voir le commentaire ci-dessous de `worker_one_try_timeout` pour voir quand modifier ce paramètre

`worker_one_try_timeout`	Entier	secondes	**30**	Temps maximum d'exécution d'une sauvegarde au sein d'un worker.
				
				Si ce temps est dépassé, un nouvel essai de sauvegarde de rétention est effectué par le worker.
				
				⚠️ Ne doit pas être plus grand que le `worker_timeout`.
				
				Si le Scheduler remonte des erreurs de sauvegarde de rétention ou si la sauvegarde dure plus longtemps que la durée de ce paramètre. Si c'est le cas, c'est qu'il y a eu plusieurs essais de sauvegarde.
				
				Le timeout peut être atteint pour des raisons :
				• Structurel : augmentation du nombre d'éléments supervisés, migration sur disque plus lent, augmentation de la charge de la base MongoDB…
				  ◦ Alors augmenter le timeout de ce paramètre
				• Conjoncturel : ralentissement du réseau entre le Scheduler et la base, sauvegarde non programmé de la base…
				  ◦ Ainsi augmenter le timeout du paramètre `worker_timeout`
				
				il est possible de monitorer la durée de sauvegarde de la rétention grâce au modèle d'hôte shinken-scheduler (*Voir la page* Modèle shinken-scheduler)

`max_number_of_workers`	Entier	---	**4**	Nombre de workers maximum (*nombre de clones du module*) qui traitent la sauvegarde de la rétention en parallèle. Un worker sera créé pour chaque CPU de la machine, mais jamais plus que la limite définie par ce paramètre. Il est possible de modifier ce paramètre si le pic de consommation mémoire produit par la sauvegarde de la rétention (*1 fois par heure*) pose un problème. Il est aussi possible de modifier ce paramètre si plusieurs CPU sont toujours utilisés. Exemple une machine avec 4 CPU et un load average tout le temps à 2, alors on peut mettre ce paramètre à 2, car il n'y a que 2 CPU disponibles.

Protection de la mémoire

```
#  #
#      MEMORY PROTECTION      #
#  #

#  Are the module worker process are waiting for enough memory to be available before being launch.
#
#           ...     : 0 ( Disable )
#           Default : 1 ( Enable )
#
# scheduler__retention_mongo__enable_sub_processes_memory_usage_protection 1

#  The sub process memory usage protection can have a system reserved memory that won't be used by
#  theses sub process when launched
#
#           Default : 0 ( no reserved memory )
#        Example : 10  (means 10% of the total memory is reserved for the system)
#
# scheduler__retention_mongo__sub_process_memory_usage_system_reserved_memory 0

#  If a sub process cannot be started because of the protection, how many seconds it will be retry
#  and wait that the system memory is freed until it fail to start
#
#           Default : 5 ( seconds )
#
# scheduler__retention_mongo__sub_processes_memory_usage_protection_max_retry_time 5
```

Comme vu précédemment, au démarrage du module, des workers sont créés. Chaque worker consomme une certaine quantité de mémoire RAM. Une protection peut-être configurée pour vérifier si la quantité de mémoire RAM libre est suffisante avant de créer ces workers.

Nom	Type	Unité	Défaut	Description
`scheduler__retention_mongo__enable_sub_processes_memory_usage_protection`	Booléen	---	**1**	• 1 : Le module attend d'avoir assez de mémoire libre pour créer ses workers. • 0 : Le module crée ses workers dès son démarrage sans vérifier la mémoire.
`scheduler__retention_mongo__sub_process_memory_usage_system_reserved_memory`	Entier	pourcentage	**0**	Pourcentage de mémoire réservé au système (*qui ne sera donc pas utilisé par le module et ses workers*). Par défaut, 0, signifie que rien n'est réservé au système.
`scheduler__retention_mongo__sub_processes_memory_usage_protection_max_retry_time`	Entier	secondes	**5**	Si la protection mémoire est active, ce temps définit l'attente maximum du module avant de considérer que son démarrage est en échec s'il n'y a pas assez de mémoire disponible.

Paramètres de gestion de la rétention

```
#  #
#      INTERNAL OPTIONS      #
#  #

#        /!\ INTERNAL : DO NOT EDIT FOLLOWING PARAMETER WITHOUT YOUR DEDICATED SUPPORT
#
#  Number of day we conserve retention data, after this time, we will delete data.
#
#           Default : 7 ( days )
#
# nb_of_max_retention_day                           7

#  Maximum number of elements load in one chunk pass.
#
#           Default : 1000
#
# size_chunk_to_load                                1000

#  Maximum number of elements delete in one chunk pass.
#
#           Default : 1000
#
# size_chunk_to_delete                              1000

#  Timeout for the execution of each chunk retention request.
#
#           Default : 300
#
# scheduler__retention_mongo__load_retention_chunk_timeout 300

#  Elements whose serialized size exceeds this threshold will generate a warning
#
#           Default : 10000 ( Kilobytes )
#
# scheduler__retention_mongo__oversized_element_warning_threshold__size 10000

#  Elements whose serialized size exceeds this threshold will generate an error
#
#           Default : 16000 ( Kilobytes )
#
# scheduler__retention_mongo__oversized_element_error_threshold__size 16000
```

Il est possible de définir certains paramètres de gestion de la rétention, tel que le nombre de jours à garder :

> Cette partie ne doit pas être modifiée sans le support.

Nom	Type	Unité	Défaut	Description
`nb_of_max_retention_day`	Entier	jours	**7**	Nombre de jours avant de supprimer une donnée de rétention.
`size_chunk_to_load`	Entier	---	**1000**	Nombre maximum d'éléments chargés depuis la base en une fois.
`size_chunk_to_delete`	Entier	---	**1000**	Nombre maximum d'éléments qui peuvent être supprimés de la base en une fois.

`scheduler__retention_mongo__load_retention_chunk_timeout`	Entier	secondes	**300**	Définit le temps maximum, en seconde, autorisé pour un batch (*chunk*) de lecture des données de rétention. Passé ce délai, la requête de lecture s'arrête, et le module s'arrête sur une erreur.
`scheduler__retention_mongo__oversized_element_warning_threshold__size`	Entier	kilooctets	**10000**	Délai en millisecondes passées à la sérialisation d'un Brok lors de l'envoi au module (*et ses workers*). Passé ce délai sera affiché dans les logs du Broker en WARNING deux messages contenant : • le temps passé à le sérialiser et la taille de ses données variables. • le temps passé à le sérialiser et le nombre de ses données variables. (*voir la page* Scheduler - Les logs du module MongodbRetention).
`scheduler__retention_mongo__oversized_element_error_threshold__size`	Entier	kilooctets	**16000**	Délai en millisecondes passées à la sérialisation d'un Brok lors de l'envoi au module (*et ses workers*). Passé ce délai sera affiché dans les logs du Broker en ERROR deux messages contenant : • le temps passé à le sérialiser et la taille de ses données variables. • le temps passé à le sérialiser et le nombre de ses données variables. (*voir la page* Scheduler - Les logs du module MongodbRetention). La valeur ne peut pas être strictement inférieur au seuil d'attention.

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 par défaut 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 :