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

Sommaire

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 Le Scheduler qui est chargé d'ordonnancer la vérification des éléments et de récupérer et analyser les résultats de ces vérifications.

Type de 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.

Configurer le module de rétention de type "mongodb_retention"

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 La rétention des données des Schedulers .

Activer le module

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.

- la définition des Schedulers se trouve **/etc/shinken/schedulers/** .
- Il faut modifier la propriété **modules** :
    - en ajoutant le nom du module que vous voulez utiliser.
    - vous pouvez éventuellement, si la situation le nécessite mettre le nom de plusieurs modules (généralement utilisé pour la migration d'une rétention à une autre)

/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                 MyMongodbRetention
...
...
...
}
```

Une fois la rétention Mongo activée sur les Schedulers concernés, il faut modifier potentiellement la configuration du module (comme l'URI de la base Mongo pour que tous les modules de rétention pointent vers l'adresse de la base de données qui hébergera les données de rétention).

- A noter que seuls les serveurs Mongo livrés par Shinken sont supportés. Il faut donc utiliser un serveur Shinken comme serveur pour la rétention.
- Un serveur Shinken sur une machine différente que les Schedulers peuvent également être utilisés pour sauvegarder la rétention.

Les paramètres

Le fichier de configuration livré à l'installation se trouve **/etc/shinken/modules/retention-mongodb.cfg**

- En fonction de votre architecture de supervision, vous aurez certainement besoin de faire plusieurs fichiers définissant des modules de rétention avec des paramètres différents.
- Pour avoir plusieurs modules de rétention mongo, copier le fichier, et à l'intérieur changer juste le nom du module, pour en avoir 2 distincts

Liste des paramètres

La liste des paramètres de ce module est :

Property	Default	Description
module_name	N/A	Nom du module
module_type	N/A	Doit être égal à **mongodb_retention** afin de définir un module de rétention Mongo.
uri	N/A	URI de connexion vers le serveur mongo. De la forme mongodb://adresse_du_serveur/?safe= false
use_ssh_tunnel	0	(0 / 1) =>Permet d'activer l'utilisation d'un tunnel SSH pour la connexion vers le serveur Mongodb
use_ssh_retry_failure	1	Nombre de re-tentative d'ouverture du tunnel SSH en cas de problème
ssh_user	shinken	Nom de l'utilisateur utilisé pour l'authentification SSH

ssh_keyfile	~shinken/.ssh/id_rsa	Clé SSH (privée) utilisée pour l'authentification SSH (note: la configuration par mot de passe n'est pas gérée).
ssh_tunnel_timeout	2	Défini le temps, en seconde, pour que le tunnel réponde et soit considéré viable avant de lancer la connexion MongoDb à travers
mongo_timeout	10	Défini le temps maximum, en seconde, pour que l'établissement de la connexion, ainsi que pour les requêtes de lectures ou d'écritures.
mongo_max_connections_retry	3	Défini le nombre maximum de tentatives de connexions à la base MogoDB avant d'abandonner
mongo_wait_before_retry	1	Défini le temps, en seconde, entre les tentatives de connexions infructueuses
database	shinken	Nom de la base de données utilisée pour sauvegarder les données de rétention dans MongoDB
replica_set	N/A	Utilisé uniquement dans le cas d'un cluster MongoDB qui possède plusieurs replicat_sets (espaces de stockage).
max_number_of_workers	4	Nombre maximum de Worker de sauvegarde utilisée par le module. Le nombre de Workers sera égal au minimum entre le nombre de processeurs et ce paramètre.
worker_timeout	120	Défini le temps maximum autorisé pour que les workers d'écritures puissent travailler, y compris leurs temps de tentatives. Passé ce temps, la sauvegarde des données de rétention est considérée comme échouée.
worker_one_try_timeout	30	Défini le temps maximum, en seconde, autorisé pour qu'un worker écrive ses données de rétention dans la base MongoDB.
scheduler__retention_mongo__enable_sub_processes_memory_usage_protection	1	Cette variable permet d'activer le mécanisme de protection de surconsommation mémoire du processus: si activé, le module va attendre que la RAM (hors swap) soit disponible pour lancer un processus Worker, égal à la taille actuelle du démon Scheduler.
scheduler__retention_mongo__sub_process_memory_usage_system_reserved_memory	0	Utilisé dans le mécanisme de protection de la surconsommation mémoire, permet de définir une réserve de mémoire, en pourcentage, que le démon laisse au système.
scheduler__retention_mongo__sub_processes_memory_usage_protection_max_retry_time	5	Utilisé dans le mécanisme de protection de la surconsommation mémoire, permet de définir le temps maximum attendu par le module pour avoir la RAM disponible pour lancer un processus Worker.
nb_of_max_retention_day	7	Le module de rétention supprime automatiquement de la base les données de rétention plus âgées que ce nombre de jours. Les données âgées arrivent quand un élément (hôte, check, cluster) est supprimé ou désactivé de la supervision.
size_chunk_to_load	1000	(Nombre d'éléments) => Taille d'un batch de récupération de données de rétention quand on charge la rétention
size_chunk_to_delete	1000	(Nombre d'éléments) => Taille d'un batch de suppression des données de rétention âgées.
`scheduler__retention_mongo__load_retention_chunk_timeout`	300	Définit le temps maximum, en seconde, autorisé pour un batch (chunk) de lecture des données de retention. Passé ce délai, la requête de lecture s'arrête, et le module s'arrête sur une erreur.

/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://addresse_du_serveur/?safe=false
    use_ssh_tunnel  0
    ssh_user        shinken
    ssh_keyfile     ~/.ssh/id_rsa

    # database: which mongodb database to use
    database        shinken

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

}
```

Types de connexions vers la base de données

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 tunnel 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 rétention, 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 sûr 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 démon (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_tunnel_timeout" doit être positionné à 2, pour que le test du tunnel SSH ait un timeout à 2 secondes
- 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 {


    ...
    #======== 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)
    ssh_tunnel_timeout          2
    ...

}
```

- 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 faut 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 La rétention des données des Schedulers .

Gestion de la reconnexions

Dans certains cas, il se peut que la connexion à mongo ait besoin de plusieurs tentatives (Ex. : coupures réseau, 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'essais 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 p récisées , 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 par défaut) quelle que soit la valeur des options précédemment définies.

Gestion du nombre de sous processus de sauvegarde

Afin d'accélérer la sauvegarde de la rétention, cette dernière est faite par des processus Workers qui permettent d'utiliser plusieurs CPU pour sérialiser les données de rétention.

Le lancement des processus worker se fait en deux étapes:

- création d'un premier processus fils au Scheduler, qui va nettoyer sa mémoire au maximum afin que ses propres fils (les workers) prennent le moins de mémoire possible.
- lancement de N processus workers:
  - N = minimum entre le nombre de CPU du serveur, et le paramètre max_number_of_workers
  - chaque worker ne va gérer que 1/N des données de rétention à sauvegarder

- Le paramètre " *worker_one_try_timeout* " permet de définir le temps d'attente pour un seul sous-processus.
  - S'il dépasse ce temps, le sous-processus est tué et relancé.
  - La valeur par défaut est de 30 (en secondes).
- Le paramètre *"worker_timeout"* permet de déterminer le temps maximum laissé à tous les sous-sous-processus pour fonctionner (comprenant leurs différentes relances).
  - Par défaut, cette valeur est de 120 (en secondes).

/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

        ...
    # 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
    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
    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 (by
default 4)
    max_number_of_workers   4

...

}
```

Important

Si ces options ne sont pas indiquées, 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) quelle que 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 :