Migration MMapV1 vers Wired Tiger avec un cluster MongoDB

Sommaire

Concept

Cette procédure permet la migration de MMapV1 vers WiredTiger sans interruption de service de Shinken (le seul impact observable pourra éventuellement être un peu de lenteur sur les accès à MongoDB) en se basant sur la haute disponibilité du Cluster MongoDB.

Préparation

> **Important !**
>
> Avant toute opération, faire un shinken-backup complet (*ou avec les options qui permettent de sauvegarder les données*) du *serveur de production* impacté.

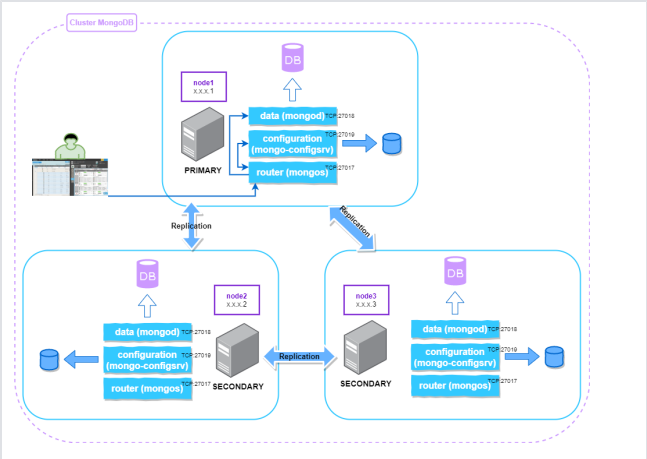
Rappel des concepts d'architecture de Cluster MongoDB

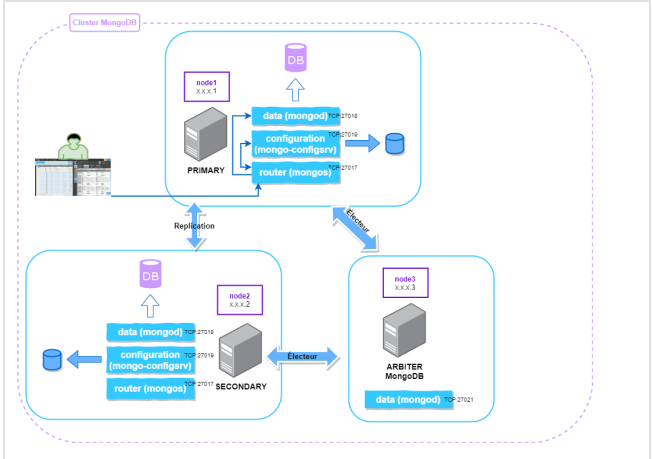
Pour une meilleure compréhension de la procédure, voici deux schémas représentant des architectures de clusters MongoDB (*Primaire - Secondaire - Secondaire ou Primaire - Secondaire - Arbiter*).

Cela permet de rappeler :

- les différents démons utilisés par le cluster MongoDB.
- ainsi que les ports par défaut utilisés.

> **Important !**
>
> Attention, l'infrastructure à migrer peut comporter des différences.





Dans les schémas, le port du serveur mongod est 27018, mais comme ce port peut être modifié dans la configuration du démon, pour le reste de la procédure, le terme de **PORT_DE_MONGOD** sera utilisé.

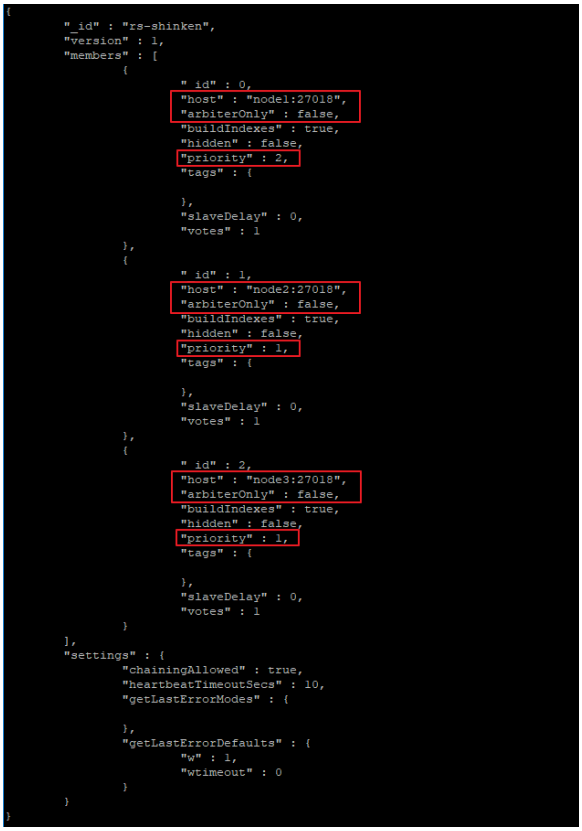
Pour rappel, le serveur mongod permet :

- de stocker des données.
- de s'assurer que les données sont bien répliquées sur les autres nœuds du cluster.

Identifier les mongod à migrer

Avant de démarrer la procédure, faire un inventaire des serveurs à migrer avec la commande suivante :

```
mongo shinken --port  PORT_DE_MONGOD --quiet --eval "printjson(rs.conf())"
```



Vérifier que les champs "host" contiennent bien les adresses des serveurs qui doivent être migrés.

Vérifier s'il y a un Arbiter MongoDB, s'assurer que celui-ci est bien "arbiterOnly" : true.

Vérifier qu'un des nœuds a un champ "priority" plus élevé que les autres (*il sera prioritaire pour devenir PRIMARY*), comme c'est le cas dans l'exemple ci-contre.

Moteur de stockage utilisé sur chaque mongod

Sur chaque serveur ayant un *mongod* exécuter la commande suivante :

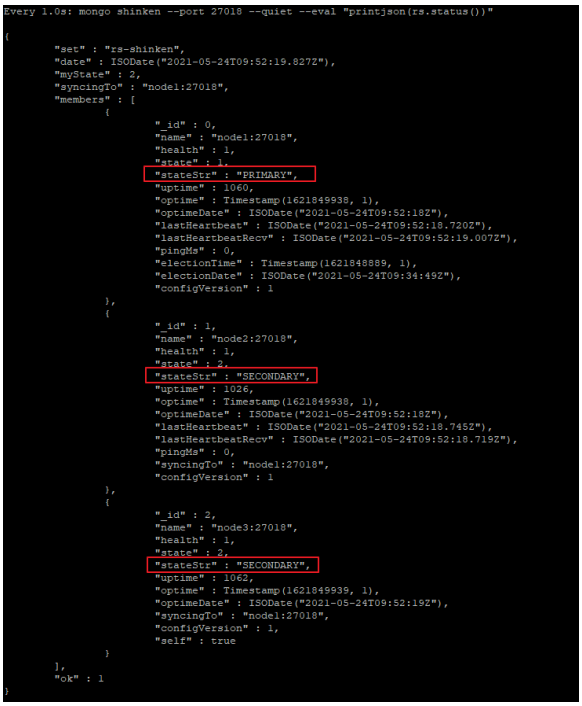
```
mongo shinken --port PORT_DE_MONGOD --quiet --eval "print(db.serverStatus().storageEngine.name)"
```

Si le format est WiredTiger, il ne serra pas nécessaire de migrer ce nœud mongod

Terminal pour surveiller l'état du cluster

Dans un terminal, qui servira à surveiller l'état du cluster, lancer la commande suivante :

```
watch -n 1 'mongo shinken --port PORT_DE_MONGOD --quiet --eval "printjson(rs.status())"'
```



Lancer la commande sur un nœud qui n'est pas en train d'effectuer la migration.

Le mieux est d'utiliser cette commande dans un terminal à part toujours visible afin de :

- Suivre l'évolution de la migration, la variable "stateStr" va passer dans les états suivants : "STARTUP2" "SECONDARY"/"PRIMARY".
- Vérifier lorsque l'on va couper le "PRIMARY" qu'un autre membre "SECONDARY" est bien élu "PRIMARY"

Description des états du Cluster MongoDB : https://docs.mongodb.com/v3.0/reference/replica-states/

Procédure de migration

Migration des mongod SECONDARY

Les mongod secondaires doivent être migrés **l'un après l'autre**.

Éteindre mongod

```
service mongod stop
```

Supprimer les données dans la base

Debian 13

```
rm -fr /var/lib/mongodb/*
```

RHEL / CentOS 7 ou RHEL / Alma / Rocky 8 ou RHEL / Alma / Rocky 9

```
rm -fr /var/lib/mongo/*
```

Modifier le type de moteur vers WiredTiger

Mettre dans le fichier */etc/mongod.conf* :

/etc/mongod.conf

```
storage:
     engine: wiredTiger
```

Démarrer mongod

```
service mongod start
```

Attendre la fin de la synchronisation avec le PRIMARY

Le mongod n'a plus de données.

Le système de synchronisation du cluster va remettre les données en place dans le nœud, comme lorsqu'on ajoute un nouveau membre au cluster MongoDB.
Il est impératif d'attendre la resynchronisation des données avant de continuer la suite de la migration.

La commande suivante donne l'état de la réplication sur le mongod en cours de migration :

```
watch -n 1 'mongo shinken --port PORT_DE_MONGOD --quiet --eval "printjson(db.printSlaveReplicationInfo())"'
```

Une fois que le mongod est synchronisé ("0 secs (0 hrs) behind the primary"), il est possible de passer à un autre nœud (*SECONDARY ou PRIMARY*)

Exemple de résultat de la commande de surveillance, une fois la synchronisation finie.

```
Every 1.0s: mongo shinken --port 27018 --quiet --eval "printjson(db.printSlaveReplicatio...

source: bmar-dev1:37018
        syncedTo: Mon May 24 2021 12:13:54 GMT+0200 (CEST)
        0 secs (0 hrs) behind the primary
source: bmar-dev2:27018
        syncedTo: Mon May 24 2021 12:13:54 GMT+0200 (CEST)
        0 secs (0 hrs) behind the primary
undefined
```

Migration du mongod PRIMARY

La migration du mongod PRIMARY va provoquer l'élection d'un nouveau PRIMARY.

Durant le temps de l'élection, il ne sera pas possible d'écrire dans la base, les manipulations suivantes vont limiter le temps d'indisponibilité de la base à 3 secondes.

Les démons de Shinken se reconnecteront au changement de PRIMARY.

Passer le PRIMARY en SECONDARY

Sur le nœud primaire, exécuter la commande suivante :

```
mongo shinken --port PORT_DE_MONGOD --quiet --eval "rs.stepDown()"
```

Important !

La commande va générer une erreur, c'est normal.

La nouvelle élection change le PRIMARY donc le shell mongo est déconnecté avec une erreur.

```
[root@kmar-dev1 ~]# mongo --port 27018 --quiet --eval "rs.stepDown()"
2021-05-24T14:32:42.475+0200 I NETWORK  DBClientCursor::init call() failed
2021-05-24T14:32:42.478+0200 E QUERY    Error: error doing query: failed
    at DBQuery._exec (src/mongo/shell/query.js:83:36)
    at DBQuery.hasNext (src/mongo/shell/query.js:240:10)
    at DBCollection.findOne (src/mongo/shell/collection.js:187:19)
    at DB.runCommand (src/mongo/shell/db.js:58:41)
    at DB.adminCommand (src/mongo/shell/db.js:66:41)
    at Function.rs.stepDown (src/mongo/shell/utils.js:1006:15)
    at (shell eval):1:4 at src/mongo/shell/query.js:83
```

Important !

La commande empêche l'élection de ce mongod en PRIMARY pour 60 secondes.

Au bout de ce temps, une nouvelle élection est proposée.

Il faut donc enchaîner avec l'étape suivante en moins de 60 secondes.

Le terminal, sur lequel la commande suivante a été précédemment lancée, permet de suivre l'élection du nouveau PRIMARY.

```
watch -n 1 'mongo shinken --port PORT_DE_MONGOD --quiet --eval "printjson(rs.status())"'
```

Éteindre mongod

```
service mongod stop
```

Supprimer les données dans la base

Debian 13

```
rm -fr /var/lib/mongodb/*
```

RHEL / CentOS 7 ou RHEL / Alma / Rocky 8 ou RHEL / Alma / Rocky 9

```
rm -fr /var/lib/mongo/*
```

Modifier le type de moteur vers WiredTiger

Mettre dans le fichier */etc/mongod.conf* :

/etc/mongod.conf

```
storage:
     engine: wiredTiger
```

Démarrer mongod

```
service mongod start
```

Attendre la fin de la synchronisation

Attendre la fin de la synchronisation comme indiqué dans la section Attendre la fin de la synchronisation avec le PRIMARY ci-dessus.