Migration MMapV1 vers Wired Tiger avec un cluster MongoDB

Sommaire

- Objectif
- Préparation
  - Rappel des concepts d'architecture de Cluster MongoDB
  - Identifier les mongod à migrer
  - Quel format de moteur de stockage est utilisé sur chaque mongod?
  - Préparer vous un fenêtre pour surveiller l'état du cluster
- Procédure de migration
  - Migration des mongod SECONDARY
    - Éteindre mongod
    - Supprimer les données dans la base
    - Modifier le type de moteur vers WiredTiger
    - Démarré mongod
    - Attendre la fin de la synchronisation avec le PRIMARY
      - Exemple de résulta de la commande précédente une fois la synchronisation finie.
  - Migration des mongod PRIMARY
    - Eteindre le Synchronizer (Si besoin)
    - Passer le PRIMARY en SECONDARY
    - Éteindre mongod
    - Rallumer le Synchronizer après élection du nouveau PRIMARY(Si besoin)
    - Supprimer les données dans la base
    - Modifier le type de moteur vers WiredTiger
    - Eteindre le Synchronizer (Si besoin)
    - Démarré mongod
    - Rallumer le Synchronizer après élection du nouveau PRIMARY(Si besoin)

Objectif

Le but est de se baser sur la haute disponibilité du Cluster MongoDB pour faire la migration de MMapV1 vers WiredTiger.

De cette manière, les interruptions de Shinken vont se limiter à quelque redémarrage de Synchronizer (*et donc limiter l'impact en terme d'indisponibilité de Shinken*).

Préparation

Prévoyez 2 périodes de maintenance car il faudra stoppé le Synchroniser 2 fois.

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

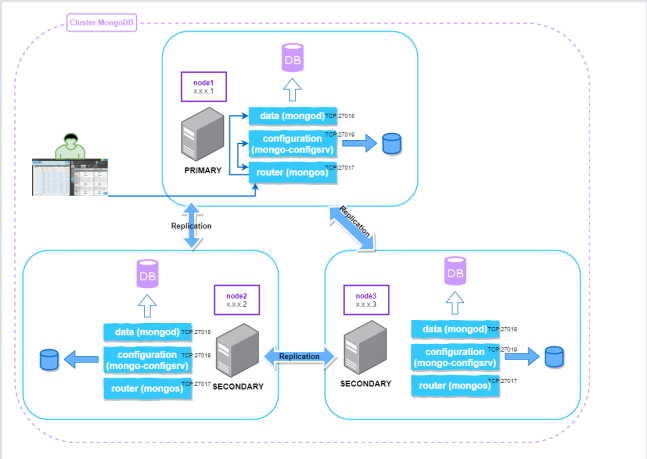
Rappel des concepts d'architecture de Cluster MongoDB

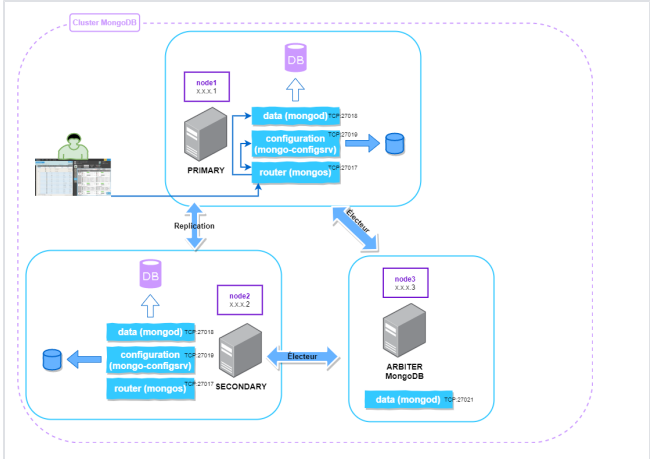
Pour une meilleur compréhension de la procédure, voici 2 schémas d'exemple de type de mise en place de clusters (*3 nœuds ou 2 nœuds et un Arbiter MongoDB*).

Cela permet de rappeler :

- les différents démons utilisé par MongoDB
- ainsi que les ports par défaut connues

> **Important !**
>
> Attention, il se peut que votre installation ait des différences.





Dans les schémas, le port du service mongod est 27018 (par défaut dans l'exemple), mais comme ce port peut être modifié, pour le reste de la procédure, nous utiliserons le terme de **PORT_DE_MONGOD**.

Pour rappel le service mongod permet :

- De se charger du stockage des données
- Il s'assure que les données sont bien répliquées sur les autres nœuds du cluster.

Identifier les mongod à migrer

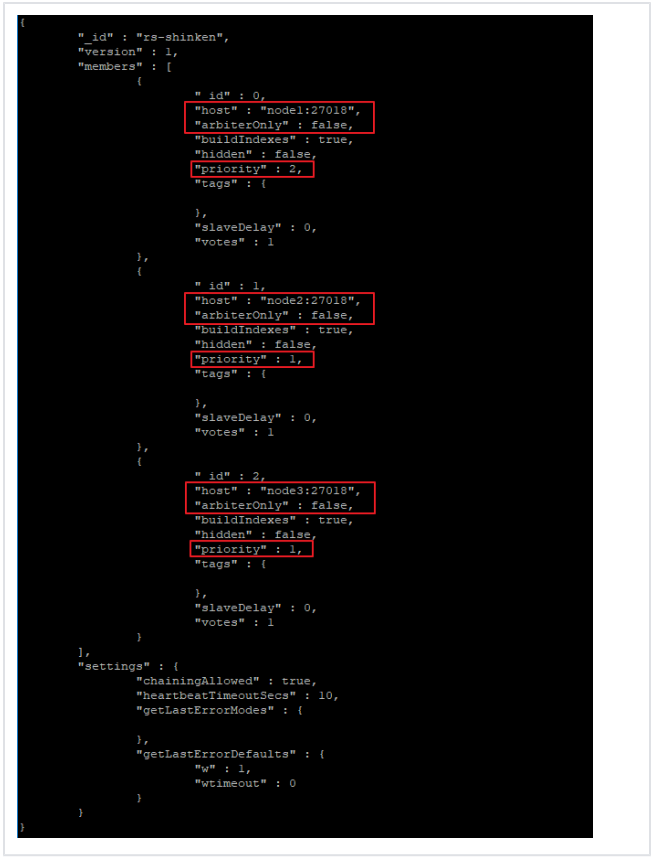
Depuis le document qui décrit votre infrastructure lister les serveurs que vous voulez migrer.

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

Vérifier que les champs "host" contiennent bien les adresses des serveurs que vous voulez migrer.

Vérifier si vous avez un Arbiter MongoDB que celui ci est bien "arbiterOnly" : true.

Vérifier aussi les priorités afin de vérifier si vous avez un PRIMARY par défaut, comme c'est le cas dans l'exemple à droite.



Quel format de moteur de stockage est utilisé sur chaque mongod?

Sur chaque serveur ayant un *mongod* faite 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 mongod

Préparer vous un fenêtre pour surveiller l'état du cluster

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

Utiliser un PORT_DE_MONGOD d'un mongod sur lequel vous **ne faite pas** de migration

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

- Vérifier quand la migration est fini "stateStr" "STARTUP2" "SECONDARY"/" PRIMARY"
- Vérifier lors que l'on va couper le "PRIMARY" qu'un autre membre "SECONDARY" va bien être élu "PRIMARY"

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

Procédure de migration

Migration des mongod SECONDARY

Éteindre mongod

```
service mongod stop
```

Supprimer les données dans la base

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

Modifier le type de moteur vers WiredTiger

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

```
storage:
     engine: wiredTiger
```

Démarré mongod

```
service mongod start
```

Attendre la fin de la synchronisation avec le PRIMARY

Le mongod n'a plus de donné à la suite de la migration. Le système de synchronisation va remettre les donnés dans le membre, comme lors que l'on ajoute un nouveau membre au cluster MongoDB.
Il faut donc attendre la re synchronisation des donnée avant de continuer la migration.

La commande suivant vous donne l'état de la réplication sur le mongod que nous avons migré toute les secondes.

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

Une fois que le mongod est synchroniser ("0 secs (0 hrs) behind the primary") alors vous pouvez passer aux autres SECONDARY ou au PRIMARY

Exemple de résulta de la commande précédente une fois la synchronisation finie.

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

source: bmar-devl: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 des mongod PRIMARY

La migration du mongod PRIMARY va provoquer un nouvelle élection d'un nouveau PRIMARY. Durant le temps de l'élection il ne sera pas possible d'écrire dans la base. Les manipulations suivant vont limité le temps d'indisponibilité de la base à 3 secondes, mais il faudra redémarré le Synchronizer pour prendre en compte le changement de PRIMARY. Les autres démons se reconnecterons au changement de PRIMARY.

Eteindre le Synchronizer (Si besoin)

```
service shinken-synchronizer stop
```

Si aucun Synchronizer est utilisé par ce cluster MongoDB ne faite pas cette étape

Passer le PRIMARY en SECONDARY

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

Important !

La commande vas générer une erreur c'est normal.

La nouvelle élection change le PRIMARY donc le shell mongo se reconnecte avec une erreur.

```
[root@bmar-devl ~]# 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 bous de ce temps une nouvelle élection est proposée.

Il faut donc passer à l'étape suivant en moins de 60 secondes.

Pensé à suivre les élections avec votre fenêtre qui à la commande

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

Éteindre mongod

```
service mongod stop
```

Rallumer le Synchronizer après élection du nouveau PRIMARY(Si besoin)

A faire que si le Synchronizer a été coupé précédemment.

```
service shinken-synchronizer start
```

Pensé à suivre les élections avec votre fenêtre qui à la commande

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

Supprimer les données dans la base

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

Modifier le type de moteur vers WiredTiger

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

```
storage:
     engine: wiredTiger
```

Eteindre le Synchronizer (Si besoin)

```
service shinken-synchronizer stop
```

A faire uniquement si :

- Un Synchronizer utilise ce cluster MongoDB
- **ET** si ce mongod est le PRIMARY par défaut (Voir Identifier les mongod à migrer)

Dans ces conditions le redémarrage de ce mongod va provoqué une nouvelle élection et le PRIMARY va changer, Il faut donc redémarre le Synchronizer.

Démarré mongod

```
service mongod start
```

Rallumer le Synchronizer après élection du nouveau PRIMARY(Si besoin)

A faire que si le Synchronizer a été coupé précédemment.

```
service shinken-synchronizer start
```

Pensé à suivre les élections avec votre fenêtre qui à la commande

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