




Mise en place d'une architecture tolérante aux pannes MongoDB

Pour mettre en place une architecture avec MongoDB tolérante aux pannes on procède de la façon suivante :

1. Définir un répertoire de sauvegarde pour chacun des serveurs.

Ici, nous partons sur trois serveurs dont les répertoires sont nommés rs1, rs2 et rs3, avec différents ports d'écoute (27018, 27019, 27020).

 RS1	09/12/2020 17:21	Dossier de fichiers
 RS2	09/12/2020 17:05	Dossier de fichiers
 RS3	09/12/2020 17:05	Dossier de fichiers

2. Lancement de 3 serveurs :

```
mongod --replSet rs0 --port 27018 --dbpath /data/RS1
mongod --replSet rs0 --port 27019 --dbpath /data/RS2
mongod --replSet rs0 --port 27020 --dbpath /data/RS3
```

3. Ajout des nœuds au replica set :

Pour commencer, il faut se connecter au 1^{er} serveur via la commande ci-dessous.

```
C:\Program Files\MongoDB\Server\4.0\bin>mongo --port 27018
```

Ensuite, il faut initialiser ce serveur avec la commande ci-dessous. Le serveur comprendra qu'il est le PRIMARY

```
> rs.initiate ();
```

Pour ajouter des nœuds au RS, il faut utiliser les commandes suivantes. En l'occurrence ici, les deux serveurs qu'on a créé au préalable

```
rs.add("local:27019");
```


```
rs.add("local:27020");
```

Suite à cela, nous replica est créé et on peut le vérifier avec rs.statut()

```
rs0:PRIMARY> rs.status()
{
  "set" : "rs0",
  "date" : ISODate("2020-12-09T17:18:18.212Z"),
  "myState" : 1,
  "term" : NumberLong(1),
  "syncingTo" : "",
  "syncSourceHost" : "",
  "syncSourceId" : -1,
  "heartbeatIntervalMillis" : NumberLong(2000),
  "optimes" : {
    "lastCommittedOpTime" : {
      "ts" : Timestamp(1607534292, 1),
      "t" : NumberLong(1)
    },
    "readConcernMajorityOpTime" : {
      "ts" : Timestamp(1607534292, 1),
      "t" : NumberLong(1)
    },
    "appliedOpTime" : {
      "ts" : Timestamp(1607534292, 1),
      "t" : NumberLong(1)
    },
    "durableOpTime" : {
      "ts" : Timestamp(1607534292, 1),
      "t" : NumberLong(1)
    }
  },
  "lastStableCheckpointTimestamp" : Timestamp(1607534285, 1),
  "electionCandidateMetrics" : {
    "lastElectionReason" : "electionTimeout",
    "lastElectionDate" : ISODate("2020-12-09T17:15:07.320Z"),
    "electionTerm" : NumberLong(1),
    "lastCommittedOpTimeAtElection" : {
      "ts" : Timestamp(0, 0),
      "t" : NumberLong(-1)
    },
    "lastSeenOpTimeAtElection" : {
      "ts" : Timestamp(1607534107, 1),
      "t" : NumberLong(-1)
    },
    "numVotesNeeded" : 1,
    "priorityAtElection" : 1,
    "electionTimeoutMillis" : NumberLong(10000),
    "newTermStartDate" : ISODate("2020-12-09T17:15:07.330Z"),
    "wMajorityWriteAvailabilityDate" : ISODate("2020-12-09T17:15:07.359Z")
  },
  "members" : [
    {
      "_id" : 0,
      "name" : "localhost:27018",
```

4. Définition de l'arbitre

Création d'un dossier dans data

 ARB	09/12/2020 17:47	Dossier de fichiers
---	------------------	---------------------

On se connecte à l'arbitre avec la commande :

```
mongod --replSet rs0 --port 30000 --dbpath /data/ARB
```

Ensuite sur le serveur primaire, on doit ajouter l'arbitre de la façon suivante :

```
rs.addArb("localhost:30000")
```

Avec un rs.status(), on constate que l'arbitre a été créé

```
"_id" : 3,  
"name" : "localhost:30000",  
"health" : 1,  
"state" : 7,  
"stateStr" : "ARBITER",  
"uptime" : 57,  
"lastHeartbeat" : ISODate("2020-12-09T17:29:01.811Z"),  
"lastHeartbeatRecv" : ISODate("2020-12-09T17:29:01.907Z"),  
"pingMs" : NumberLong(0),  
"lastHeartbeatMessage" : "",  
"syncingTo" : "",  
"syncSourceHost" : "",  
"syncSourceId" : -1,  
"infoMessage" : "",  
"configVersion" : 4
```