

TP – Mise en place d'un architecteur avec réplication avec MongoDB

Tous d'abord, nous devons créer des répertoires qui vont accueillir les données qui vont être sauvegarder (un répertoire pour chaque serveur). Cela va permettre de créer plusieurs Serveurs de répliquations.

Création des répertoires de sauvegarde pour chaque serveur :

Pour notre TP nous allons créer trois nœuds (ou serveur) nommé de la façon suivante : r0s1 pour réplikatSet 0 serveur 1, r0s2 et r0s3

Pour cela je vais créer le répertoire a la racine dans /data pour ce faire j'utilise la commande « mkdir /data/r0s1 »

Nous allons par la suite utiliser mongod -replSet rs0 -port 27018 -dbpath /data/r0s1, chaque un de c'est serveur va alors devenir des serveurs de stockage en associant un port d'écoute différent (exemple : 21018, 27019).

Le serveur r0s1 :

```

C:\Windows\system32\cmd.exe - mongod --replSet rs0 --port 27018 --dbpath /data/r0s1

where to place markers for truncation
2020-12-09T15:40:12.064+0100 I STORAGE [conn1] WiredTiger record store oplog pr
ocessing took 1ms
2020-12-09T15:40:12.078+0100 I REPL [conn1] *****
2020-12-09T15:40:12.078+0100 I STORAGE [conn1] createCollection: local.system.r
eplset with generated UUID: d708b5a7-d7c9-4c5c-8f99-156a0ad8fdb5
2020-12-09T15:40:12.094+0100 I STORAGE [conn1] createCollection: admin.system.v
ersion with provided UUID: 5ffde4be-a744-4180-86f7-f0e3bd0fe111
2020-12-09T15:40:12.106+0100 I COMMAND [conn1] setting featureCompatibilityVers
ion to 4.0
2020-12-09T15:40:12.106+0100 I NETWORK [conn1] Skip closing connection for conn
ection # 1
2020-12-09T15:40:12.109+0100 I REPL [conn1] New replica set config in use: {
  _id: "rs0", version: 1, protocolVersion: 1, writeConcernMajorityJournalDefault:
true, members: [ { _id: 0, host: "localhost:27018", arbiterOnly: false, buildIn
dexes: true, hidden: false, priority: 1.0, tags: {}, slaveDelay: 0, votes: 1 } ],
settings: { chainingAllowed: true, heartbeatIntervalMillis: 2000, heartbeatTim
eoutSecs: 10, electionTimeoutMillis: 10000, catchUpTimeoutMillis: -1, catchUpTak
eoverDelayMillis: 30000, getLastErrorModes: {}, getLastErrorDefaults: { w: 1, wt
imeout: 0 }, replicaSetId: ObjectId('5fd0e1cc80f8435adb807bba') } }
2020-12-09T15:40:12.110+0100 I REPL [conn1] This node is localhost:27018 in
the config
2020-12-09T15:40:12.111+0100 I REPL [conn1] transition to STARTUP2 from STAR
TUP
2020-12-09T15:40:12.112+0100 I REPL [conn1] Starting replication storage thr
  
```

Le serveur r0s2 :

```
C:\Windows\system32\cmd.exe - mongod --replSet rs0 --port 27019 --dbpath /data/r0s2

n admin.system.keys as collection version: <unsharded>
2020-12-09T15:42:40.485+0100 I REPL [repl writer worker 3] CollectionCloner
ns:admin.system.keys finished cloning with status: OK
2020-12-09T15:42:40.492+0100 I REPL [replication-0] CollectionCloner::start
called, on ns:config.system.sessions
2020-12-09T15:42:40.493+0100 I STORAGE [repl writer worker 4] createCollection:
config.system.sessions with provided UUID: 6be4a29a-32e0-4154-a4ac-642d977ad2da
2020-12-09T15:42:40.505+0100 I INDEX [repl writer worker 4] build index on: c
onfig.system.sessions properties: { v: 2, key: { lastUse: 1 }, name: "lsidTTLInd
ex", ns: "config.system.sessions", expireAfterSeconds: 1800 }
2020-12-09T15:42:40.505+0100 I INDEX [repl writer worker 4] building index
using bulk method; build may temporarily use up to 500 megabytes of RAM
2020-12-09T15:42:40.512+0100 I INDEX [repl writer worker 4] build index on: c
onfig.system.sessions properties: { v: 2, key: { _id: 1 }, name: "_id_", ns: "co
nfig.system.sessions" }
2020-12-09T15:42:40.512+0100 I INDEX [repl writer worker 4] building index
using bulk method; build may temporarily use up to 500 megabytes of RAM
2020-12-09T15:42:40.514+0100 I REPL [repl writer worker 5] CollectionCloner
ns:config.system.sessions finished cloning with status: OK
2020-12-09T15:42:40.524+0100 I REPL [repl writer worker 5] CollectionCloner:
:start called, on ns:config.transactions
2020-12-09T15:42:40.525+0100 I STORAGE [repl writer worker 6] createCollection:
config.transactions with provided UUID: 9a223266-625f-418a-99f2-fb81925dabb7
2020-12-09T15:42:40.538+0100 I INDEX [repl writer worker 6] build index on: c
```

Le serveur r0s3 :

```
C:\Windows\system32\cmd.exe - mongod --replSet rs0 --port 27020 --dbpath /data/r0s3

2020-12-09T16:18:18.650+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:23:18.649+0100 I NETWORK [LogicalSessionCacheReap] Starting new r
eplica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:23:18.654+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:23:18.655+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:23:18.655+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:23:18.656+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:28:18.649+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:28:18.650+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:28:18.651+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:33:18.649+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:33:18.651+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
2020-12-09T16:33:18.652+0100 I NETWORK [LogicalSessionCacheRefresh] Starting ne
w replica set monitor for rs0/localhost:27018,localhost:27019,localhost:27020
```

Initialisation replicaSet

On va ensuite connecter les serveurs ensemble et les initialiser

Pour cela je me suis connecté sur le serveur principal à l'aide de la commande `mongo -port <N° port>` après cela j'initialise le replicaSet à l'aide de la commande `rs.initiate`

Ajout des différents serveurs au replicaSet

La ligne `Rs0 :Primary>` apparaît

Nous allons maintenant ajouter les serveurs r0s2 et r0s3 avec la commande :

```
rs.add("localhost :27019 ")
```

```
rs.add("localhost :27019 ")
```

Vérification de la configuration du replicaSet :

La commande `rs.conf` permet de vérifier l'ajout des serveurs de réplica

```
'members' : [
  {
    "_id" : 0,
    "host" : "localhost:27018",
    "arbiterOnly" : false,
    "buildIndexes" : true,
    "hidden" : false,
    "priority" : 1,
    "tags" : {
    },
    "slaveDelay" : NumberLong(0),
    "votes" : 1
  },
  {
    "_id" : 1,
    "host" : "localhost:27019",
    "arbiterOnly" : false,
    "buildIndexes" : true,
    "hidden" : false,
    "priority" : 1,
    "tags" : {
    },
    "slaveDelay" : NumberLong(0),
    "votes" : 1
  },
  {
    "_id" : 2,
    "host" : "localhost:27020",
    "arbiterOnly" : false,
    "buildIndexes" : true,
    "hidden" : false,
    "priority" : 1,
    "tags" : {
    }
  }
]
```

Dans l'onglet host nous allons retrouver où se situe les serveurs définis et le port qu'il utilise.

Vérification du statut du replicaSet :

Pour connaître le statuts du serveur (primaire ou secondaire) grâce à la commande `rs.status()`

```
{
  "_id" : 1,
  "name" : "localhost:27019",
  "health" : 1,
  "state" : 2,
  "stateStr" : "SECONDARY",
  "uptime" : 288,
  "optime" : {
    "ts" : Timestamp(1607525242, 1),
    "t" : NumberLong(1)
  },
  "optimeDurable" : {
    "ts" : Timestamp(1607525242, 1),
    "t" : NumberLong(1)
  },
  "optimeDate" : ISODate("2020-12-09T14:47:22Z"),
  "optimeDurableDate" : ISODate("2020-12-09T14:47:22Z"),
  "lastHeartbeat" : ISODate("2020-12-09T14:47:29.236Z"),
  "lastHeartbeatRecv" : ISODate("2020-12-09T14:47:27.767Z"),
  "pingMs" : NumberLong(0),
  "lastHeartbeatMessage" : "",
  "syncingTo" : "localhost:27018",
  "syncSourceHost" : "localhost:27018",
  "syncSourceId" : 0,
  "infoMessage" : "",
  "configVersion" : 3
}
```

Les rôles qui ont été définis sont utiles lors d'un arrêt du serveur primaire de basculer sur un deux serveur restant actif. Pour notre exemple, si le serveur primaire s'arrête, les autres serveurs vont devoir se mettre d'accord sur le serveur qui va prendre les relais, on parle alors d'élection ce qui peut provoquer un temps de rétablissement plus élevé que si nous utilisons un entier arbitre.

Créations d'un arbitre

De la même façon que pour créer un serveur nous allons créer un répertoire dans /data/arb

Puis nous allons exécuter la commande dans le réplicaSet : `mongod -port 30000 -dbpath /data/arb -replSet rs0`

Mettre en place l'arbitre replicaSet : `mongod -port 30000 -dbpath /data/arb -replSet rs0`

Ajouter l'arbitre : `Rs.addArb("localhost :30000") ;`

S'assurer de l'ajout de l'arbitre à l'aide de la commande : `rs.status()`



```
> rs.status()
{
  "configVersion" : 4,
  "term" : 1,
  "primary" : "localhost:27020",
  "secondary" : [
    {
      "_id" : 3,
      "name" : "localhost:30000",
      "health" : 1,
      "state" : 7,
      "stateStr" : "ARBITER",
      "uptime" : 2,
      "lastHeartbeat" : ISODate("2020-12-09T14:56:30.196Z"),
      "lastHeartbeatRecv" : ISODate("2020-12-09T14:56:30.306Z"),
      "pingMs" : NumberLong(0),
      "lastHeartbeatMessage" : "",
      "syncingTo" : "",
      "syncSourceHost" : "",
      "syncSourceId" : -1,
      "infoMessage" : "",
      "configVersion" : 4
    }
  ],
  "ok" : true
}
```