**Système distribué avec mongoDB**

**Prérequis**

1. Avec la version 4.4 de mongoDB il faut installer les tools <https://docs.mongodb.com/database-tools/installation/installation-windows/>

On obtient donc dans le répertoire C:/Program Files/MongoDB, le dossier Server et Tools.

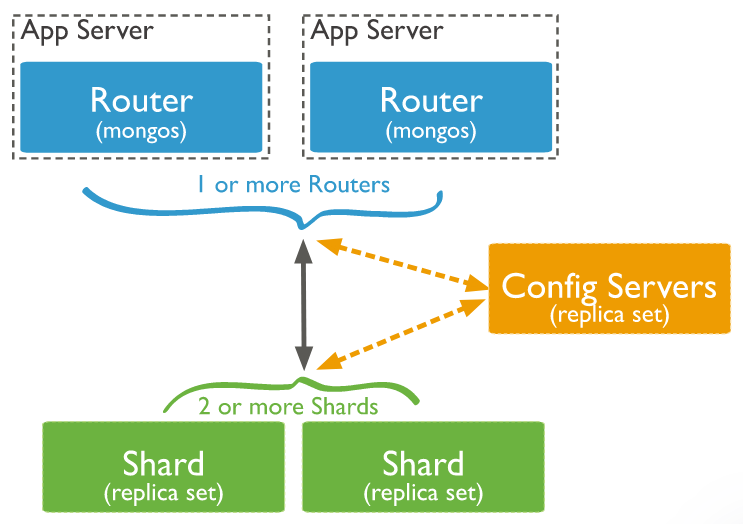
1. Ajouter les chemins

C:\Program Files\MongoDB\Server\4.4\bin

C:\Program Files\MongoDB\Tools\100\bin

à la variable d’environnement Path

**Structure du cluster**

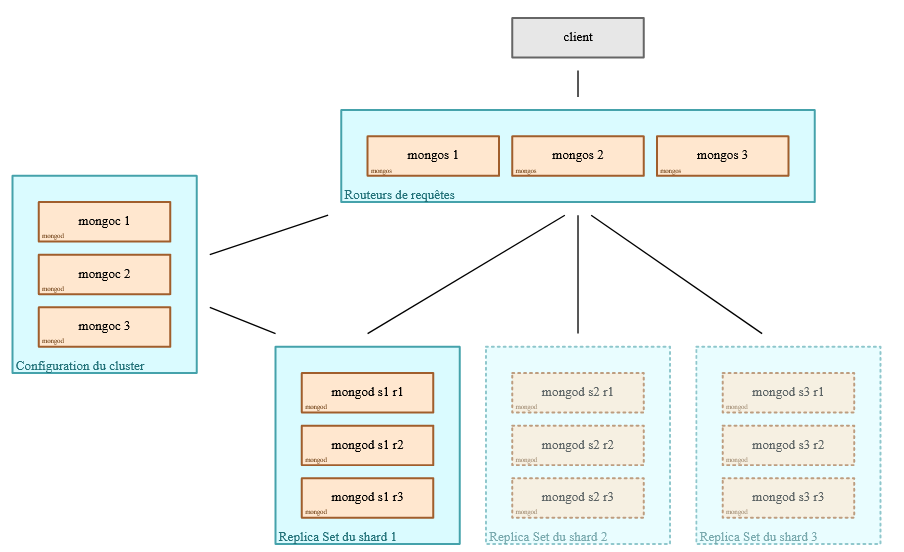


Un cluster est composé au minimum de :

* Un router (mongos) : il s’occupe du routage des requêtes. Pour gérer la tolérance aux pannes, on peut créer 2 routers.
* Un config server : c’est un serveur et il est configuré en ReplicaSet. Il contient la connaissance du réseau (routers et clusters). Ils gèrent les informations de répartitions de charges sur les différents shards et la structure des ReplicaSet.
* Deux shards au minimum : ils sont configurés en ReplicaSet (3 serveurs avec redondance des données). Ils contiennent l’ensemble des données (chunks). Ils peuvent contenir plusieurs chunks.

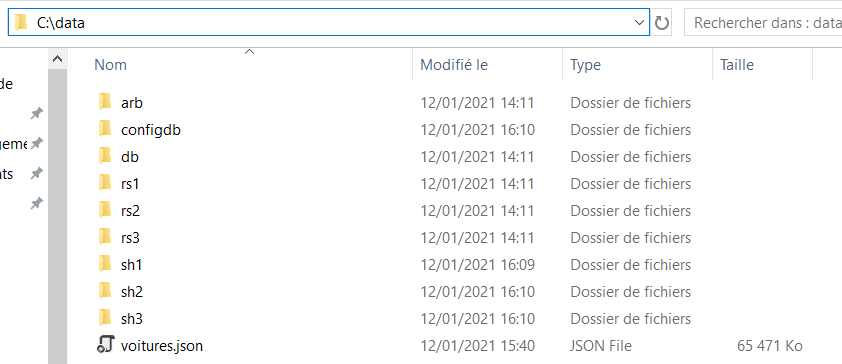
Pour notre démonstration, nous avons la structure suivante :

* Un router ;
* Un configServer ;
* Trois shards.



**Réalisation du cluster**

Il faut tout d’abord créer les dossiers associés à ces 4 éléments (configServer et shards) dans le répertoire C:/data :



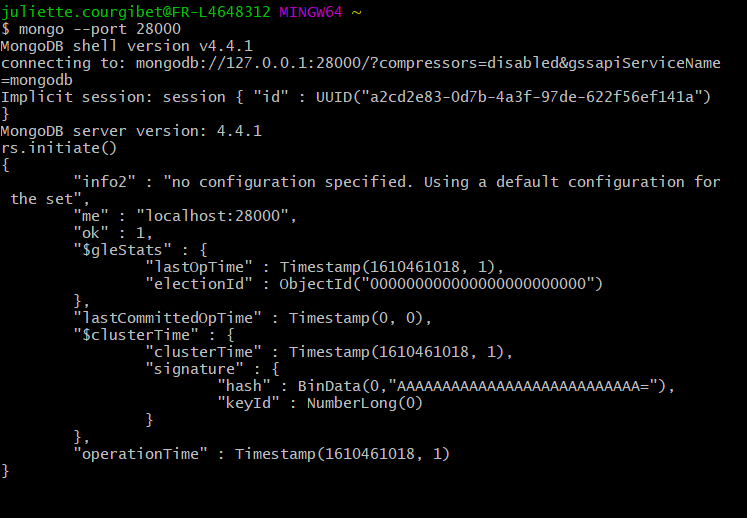
On crée les dossiers sh1, sh2, sh3 et configdb.

1. Lancement du serveur configServer en ReplicaSet.

Dans une fenêtre de commande, on lance :

mongod --configsvr --replSet configReplSet --port 28000 --dbpath C:/data/configdb

Dans une autre fenêtre, on se connecte au port 28000 (donc au configServer) et on l’initie :



Avec la commande : mongo –port 28000

Puis rs.initiate();

1. Lancement des serveurs de données (shards) en ReplicaSet.

Dans 3 fenêtres de commande, on lance :

mongod --shardsvr --replSet sh1 --port 27031 --dbpath C:/data/sh1

mongod --shardsvr --replSet sh2 --port 27032 --dbpath C:/data/sh2

mongod --shardsvr --replSet sh3 --port 27033 --dbpath C:/data/sh3

Puis dans une autre invite de commande, on initie les shards :

mongo --port 27031 --eval "rs.initiate()"

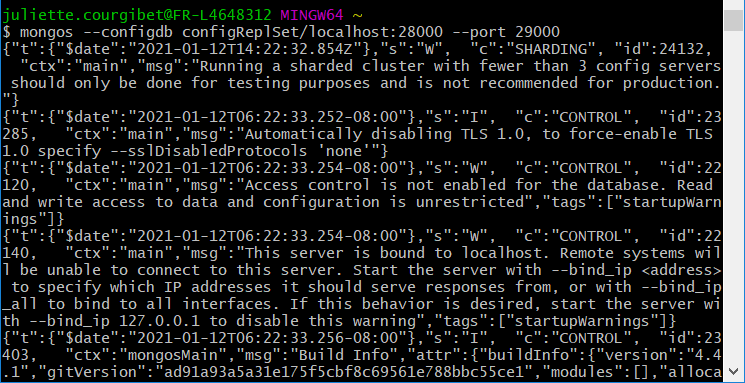
mongo --port 27032 --eval "rs.initiate()"

mongo --port 27033 --eval "rs.initiate()"

1. Lancement du routeur et connexion aux shards

Pour lancer le router, dans une invite de commande, on tape pour le connecter au configServer du port 28000 :

mongos --configdb configReplSet/localhost:28000 --port 29000



On se connecte ensuite au port 29000 (router) pour ajouter les shards :

Dans une autre invite de commande, on tape :

mongo --port 29000

sh.addShard("sh1/localhost:27031");

sh.addShard("sh2/localhost:27032");

sh.addShard("sh3/localhost:27033");

Dans la figure suivante, on voit le résultat obtenu (et attendu) :



**Ingestion de données dans le cluster**

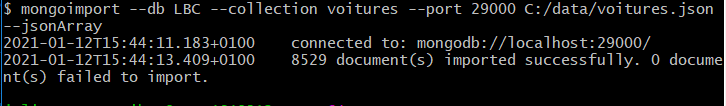
Pour cette étape, on crée une base de données et une collection voitures. Pour cela, on récupère les données issues de notre base (localhost :27017) en enregistrant un fichier au format.json (exportation de données).

Dans la même invite de commande connectée au port 29000, on crée une base de données LBC en autorisant le sharding et en créant une collection voitures :

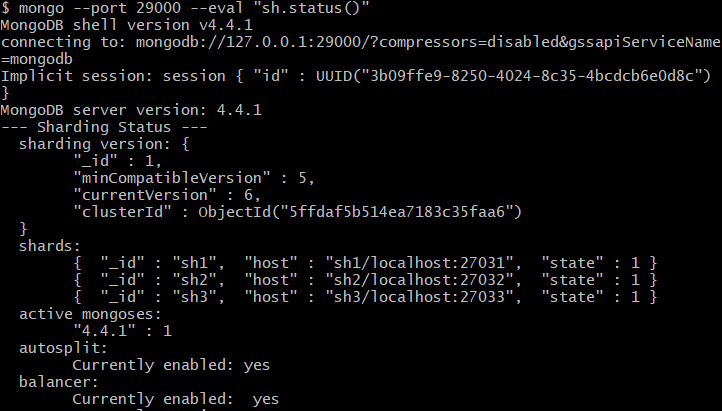


On importe maintenant les données dans une autre invite de commande avec :

mongoimport --db LBC --collection voitures --port 29000 C:/data/voitures.json --jsonArray



On obtient le statut du cluster :



--- Sharding Status ---

sharding version: {

"\_id" : 1,

"minCompatibleVersion" : 5,

"currentVersion" : 6,

"clusterId" : ObjectId("5ffdaf5b514ea7183c35faa6")

}

shards:

{ "\_id" : "sh1", "host" : "sh1/localhost:27031", "state" : 1 }

{ "\_id" : "sh2", "host" : "sh2/localhost:27032", "state" : 1 }

{ "\_id" : "sh3", "host" : "sh3/localhost:27033", "state" : 1 }

active mongoses:

"4.4.1" : 1

autosplit:

Currently enabled: yes

balancer:

Currently enabled: yes

Currently running: no

Failed balancer rounds in last 5 attempts: 0

Migration Results for the last 24 hours:

682 : Success

databases:

{ "\_id" : "LBC", "primary" : "sh3", "partitioned" : true, "version" : { "uuid" : UUID("6e7cb2b0-b620-454e-a33b-fc45a10933de"), "lastMod" : 1 } }

LBC.voitures

shard key: { "\_id" : 1 }

unique: false

balancing: true

chunks:

sh3 1

{ "\_id" : { "$minKey" : 1 } } -->> { "\_id" : { "$maxKey" : 1 } } on : sh3 Timestamp(1, 0)

{ "\_id" : "config", "primary" : "config", "partitioned" : true }

config.system.sessions

shard key: { "\_id" : 1 }

unique: false

balancing: true

chunks:

sh1 342

sh2 341

sh3 341

too many chunks to print, use verbose if you want to force print

{ "\_id" : "voitures", "primary" : "sh3", "partitioned" : true, "version" : { "uuid" : UUID("6fa5223d-126a-4103-9a85-74391a07aad1"), "lastMod" : 1 } }