

Université Cheikh Anta Diop



Département Génie Informatique

Année Universitaire : 2023-2024 Licence Génie Logiciel et Système d'Information (LGSi)

Examen Base de Données

Classe: GLSIB

Membres: Sokhna Aicha Amar

Mareme Ba Loum

Déployer un cluster MongoDB sous Docker

MongoDB est une base de données à usage général conçue pour le Web. Il offre entre autres une haute disponibilité lorsqu'il est utilisé dans des clusters, également appelés répliques. Un groupe de répliques est un groupe de serveurs MongoDB, appelés nœuds, contenant une copie identique des données. Si l'un des serveurs tombe en panne, les deux autres prendront la charge pendant que celui qui a planté redémarre.

Les étapes pour créer un cluster Docker sont les suivantes.

1. Installer l'image Mongo de Docker.
2. Créez un réseau Docker.
3. Démarrez trois instances de MongoDB.
4. Lancez l'ensemble de répliques.

Une fois que vous avez un cluster MongoDB opérationnel, vous pourrez l'expérimenter.

Pour ce TP, nous aurons besoin d'un environnement d'exécution pour les conteneurs.

Installation de l'image Mongo de Docker

mongo:5 est l'image qui sera utilisée par Docker. Cette image est la version 5 du serveur MongoDB Community (maintenu par Docker). Vous pouvez également utiliser une image personnalisée MongoDB Enterprise. Pour télécharger l'image veuillez suivre ce lien : https://hub.docker.com/_/mongo

Créer un réseau Docker

La deuxième étape consiste à créer un réseau Docker. Ce réseau permettra à chacun de vos conteneurs exécutés sur ce réseau de se voir. Pour créer un réseau, exécutez la commande **docker network create**.

Le paramètre mongoCluster ici est le nom du réseau ; nous pouvons en choisir un qui convient à votre configuration. Après avoir exécuté la commande, nous devrions voir l'identifiant du réseau que nous venons de créer.

Notez que cette commande ne doit être exécutée qu'une seule fois. Si nous démarrons nos conteneurs par la suite, nous n'aurons pas besoin de recréer ce réseau.

Démarrer les instances MongoDB

Nous sommes maintenant prêt à démarrer notre premier conteneur avec MongoDB. Pour démarrer le conteneur, utilisons la commande docker run.

```
C:\Windows\System32>docker run -d --rm --name mongo1 --network mongoCluster -p 27017:27017 mongo:5 mongod --replSet myReplicaSet --bind_ip_all
b73f41375b176e0f0ebce520b93f831d66aad0b0416c300580c1e15e4192f7
docker: Error response from daemon: driver failed programming external connectivity on endpoint mongo1 (68dfcacd5e4cc05fad70121cc62f0df9d9bf927e9b0412f275c081ec6e08292d
): Bind for 0.0.0.0:27017 failed: port is already allocated.

C:\Windows\System32>docker run -d --rm --name mongo1 --network mongoCluster -p 27018:27017 mongo:5 mongod --replSet myReplicaSet --bind_ip_all
e3253f222f1e7558f0513d2d52b135cb93efac69d44e419b8f8be0ce540c1f98

C:\Windows\System32>docker run -d --rm --name mongo2 --network mongoCluster -p 27019:27017 mongo:5 mongod --replSet myReplicaSet --bind_ip_all
ef219e7a96e3a959f1c7702b078571913ae457efac896a954a4d6e2351a07618
```

Ici, nous disons à docker de démarrer un conteneur avec les paramètres suivants :

-d indique que ce conteneur doit s'exécuter en mode détaché (en arrière-plan).

-p indique le mappage de port. Toute demande entrante sur le port 27017 de votre machine sera redirigée vers le port 27017 du conteneur.

--name indique le nom du conteneur. Cela deviendra le nom d'hôte de cette machine.

--network indique quel réseau Docker utiliser. Tous les conteneurs d'un même réseau peuvent se voir.

mongo:5 est l'image qui sera utilisée par Docker.

La suite de l'instruction est la commande qui sera exécutée une fois le conteneur démarré. Cette commande crée une nouvelle instance mongod prête pour la réplication. Si la commande a été exécutée avec succès, vous devriez voir une longue chaîne hexadécimale représentant l'**ID du conteneur**. Démarrons deux autres conteneurs. Nous devrons utiliser un nom différent et un port différent pour ces deux.

Nous avons maintenant trois conteneurs exécutant MongoDB. Nous pouvons utiliser docker ps pour valider qu'ils sont en cours d'exécution.

```
C:\Windows\System32>docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ef219e7a96e3	mongo:5	"docker-entrypoint.s..."	20 seconds ago	Up 19 seconds	0.0.0.0:27019->27017/tcp	mongo2
a3253f222f1e	mongo:5	"docker-entrypoint.s..."	56 seconds ago	Up 54 seconds	0.0.0.0:27019->27017/tcp	mongo1
a9db2f834e1e	mongo:latest	"docker-entrypoint.s..."	2 days ago	Up 4 minutes	0.0.0.0:27017->27017/tcp	practical_hypatia

Lancer la réplication

L'étape suivante consiste à créer le jeu de répliques réelles avec les trois nœuds. Pour ce faire, nous devrons utiliser le shell MongoDB. Cet outil CLI (interface de ligne de commande) est disponible avec l'installation MongoDB par défaut ou installé indépendamment. Cependant, si vous n'avez pas l'outil installé sur votre ordinateur portable, il est possible d'utiliser mongosh disponible à l'intérieur des conteneurs avec la commande **docker exec**.

```
C:\Windows\System32>docker exec -it mongo1 mongosh --eval "rs.initiate({_id: \"myReplicaSet\", members: [{_id: 0, host: \"mongo1\"}, {_id: 1, host: \"mongo2\"}]})"
```

```
{ ok: 1 }
```

Cette commande indique à Docker d'exécuter l'outil mongosh dans le conteneur nommé mongo1. mongosh essaiera alors d'évaluer la commande rs.initiate() pour lancer le jeu de répliques.

Dans le cadre de l'objet de configuration transmis à rs.initiate(), vous devrez spécifier le nom du jeu de répliques (myReplicaSet, dans ce cas), ainsi que la liste des membres qui feront partie du jeu de répliques. Les noms d'hôte des conteneurs sont les noms des conteneurs spécifiés par le paramètre **--name** dans la commande docker run. Vous pouvez en savoir plus sur les options possibles pour configurer un jeu de répliques à partir de la documentation.

Si la commande a été exécutée avec succès, vous devriez voir un message de la CLI mongosh indiquant les numéros de version de MongoDB et Mongosh, suivi d'un message indiquant :

```
{ ok : 1 }
```

Tester et vérifier la réplication

Vous devriez maintenant avoir un jeu de répliques en cours d'exécution. Si vous souhaitez vérifier que tout a été configuré correctement, vous pouvez utiliser l'outil CLI mongosh pour évaluer l'instruction rs.status(). Cela vous fournira l'état de votre jeu de répliques, y compris la liste des membres.

```
C:\Windows\System32>docker exec -it mongo1 mongosh -eval "rs.status()"
{
  set: 'myReplicaSet',
  date: ISODate('2024-06-13T19:01:39.253Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 2,
  writableVotingMembersCount: 2,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1718305294, i: 7 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2024-06-13T19:01:34.161Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1718305294, i: 7 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1718305294, i: 7 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1718305294, i: 7 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2024-06-13T19:01:34.161Z'),
    lastDurableWallTime: ISODate('2024-06-13T19:01:34.161Z')
  },
}
```

Vous pouvez également vous connecter à votre cluster à l'aide de MongoDB Compass pour créer une base de données et ajouter des documents. Notez que les données sont créées à l'intérieur du stockage de conteneurs et seront détruits lorsque les conteneurs seront supprimés du système hôte. Pour vérifier que votre jeu de répliques fonctionne, vous pouvez essayer d'arrêter l'un des conteneurs avec `docker stop` et réessayer de lire à partir de votre base de données.

Les données seront toujours là. Vous pouvez voir que le cluster est toujours en cours d'exécution en utilisant `rs.status()` sur le conteneur `mongo2`.

Vous verrez toujours le jeu de réplicas, mais notez que le premier membre est maintenant arrêté et que l'un des deux autres membres a été élu comme nœud principal. Si vous redémarrez votre conteneur `mongo1`, vous pourrez le revoir dans le jeu de réplicas, mais en tant que membre secondaire.

Manipulation de base

Il s'agit de créer une base, une collection, d'y insérer des données et de l'interroger. Les documents fournis correspondent à un extrait d'une base de publications scientifiques, **The DBLP Computer Science Bibliography**.
GESTION DE COLLECTION

• Se connecter au serveur primaire `mongo1` : `docker exec -it mongo1 mongosh`

```
C:\Windows\System32> docker exec -it mongo1 mongosh
Current Mongosh Log ID: 666b423e10ab12581fa26a12
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.6
Using MongoDB:      5.0.27
Using Mongosh:      2.2.6

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2024-06-13T18:59:28.577+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2024-06-13T18:59:29.314+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2024-06-13T18:59:29.314+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never' in this binary version
-----
```

• Créer la nouvelle base de : `use DBLP;`

```
myReplicaSet [direct: primary] test> use DBLP
switched to db DBLP
myReplicaSet [direct: primary] DBLP> db.createCollection('publis')
{ ok: 1 }
myReplicaSet [direct: primary] DBLP> exit
```

Importer les données du TP dans MongoDB :

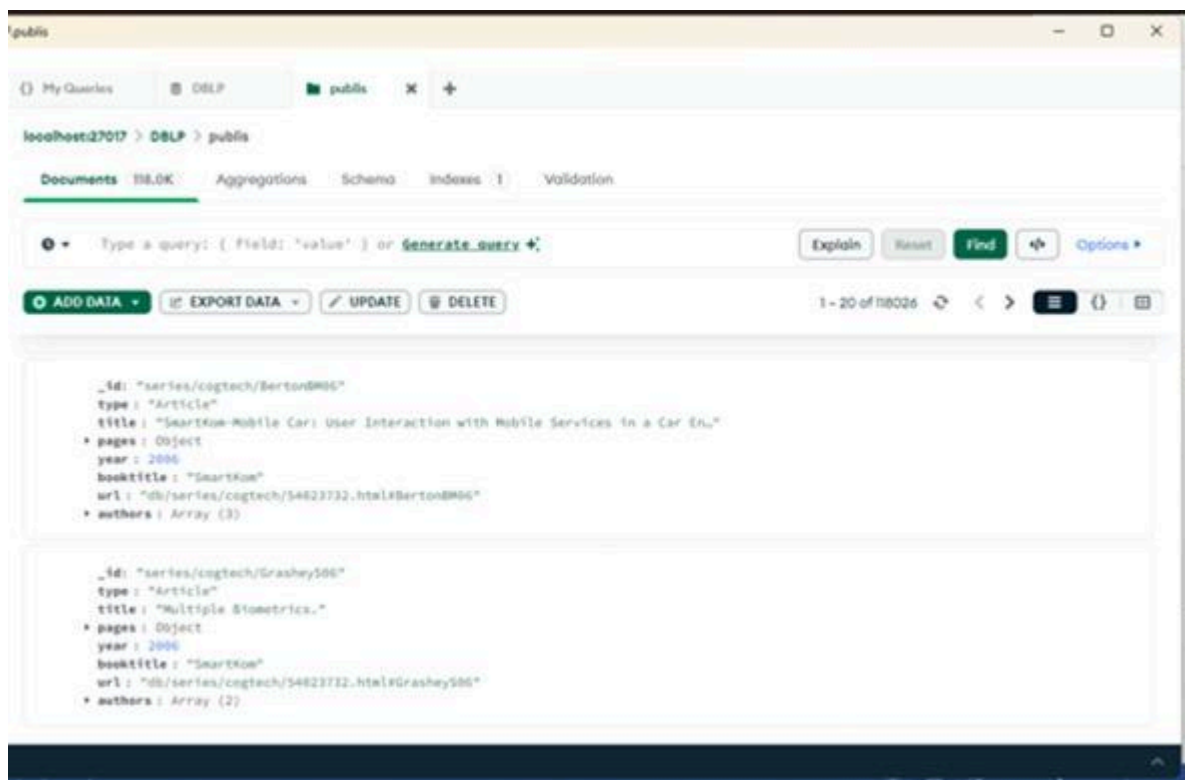
- Télécharger le fichier contenant les données : DBLP.json.zip
- Décompresser le fichier dblp.json.zip
- Copier le fichier dblp.json dans la racine du conteneur mongo1 : `docker cp dblp.json container_id:/`

```
C:\Windows\System32>docker cp C:\Users\HP\Downloads\dblp.json mongo1:/dblp.json
Successfully copied 37.9MB to mongo1:/dblp.json
```

1. Liste de tous les livres (type "Book") ;

2. Liste des publications ;

3. Dbp visualiser au niveau de mongoddb



4. server.js se charge de la connexion à la base de données

```

JS server.js > ...
10 app.use(bodyParser.json());
11 app.use(bodyParser.urlencoded({ extended: true }));
12
13 // Connexion à la base de données MongoDB
14 mongoose
15 .connect(db.url, {
16   useNewUrlParser: true,
17   useUnifiedTopology: true,
18 })
19 .then(() => {
20   console.log(`Connecté à la base de données '${db.url}' !`);
21 })
22 .catch(err => {
23   console.error(`Impossible de se connecter à la base de données '${db.url}'`);
24   process.exit();
25 });
26
27 // Middleware de logging

```

4.db.config.js se charge de récupérer les données au niveau de DBLP

```

config > JS db.config.js > ...
1 module.exports = {
2   url: "mongodb://localhost:27017/DBLP",
3 };

```

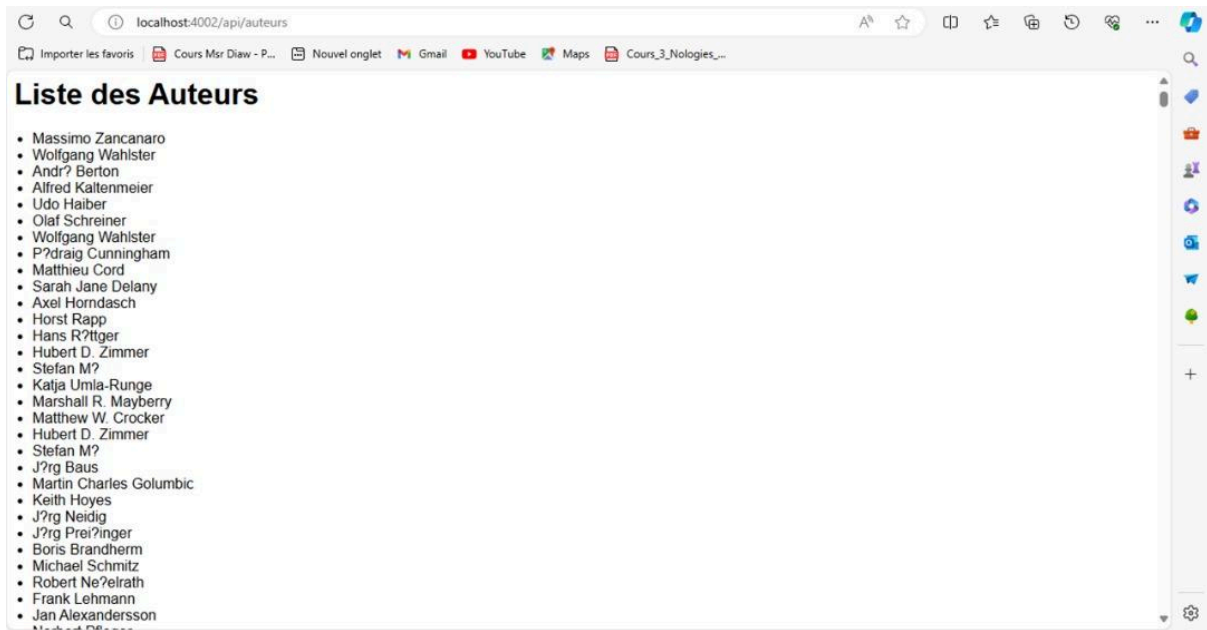
5.Voici le lien qu'il faut mettre sur le navigateur pour visualiser l'application

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
(node:12564) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next version.
Le serveur Express écoute sur le port 4002.
Connecté à la base de données 'mongodb://localhost:27017/DBLP' !
GET /api/auteurs 200 54204.191 ms - 7276220
GET /favicon.ico 404 164.472 ms - 150

```

6.Liste des auteurs



Ici nous parvenons à voir la liste des auteurs