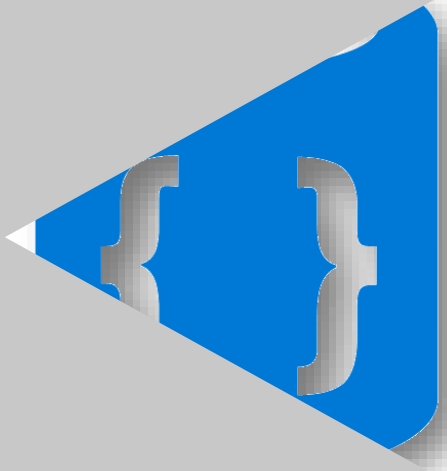
2021 - 2022



Rapport de TP

NoSQL

Enseignants:

- **Mlle BOUA Marie**
- M. ZEGBOLOU Dexter

Membres du groupe :

- **NEBOUT Abraham**
- **GUINDO Hussen**
- **DEGBUN Josue**

Sommaire

Introduction

Depuis une vingtaine d'années, les données générées n'ont fait que s'accroître. Actuellement, nous produisons annuellement une masse de données très importante estimée à près de 3 trillions (3.10^{18}) d'octets de données. On estime ainsi qu'en 2016, 90% des données dans le monde ont été créées au cours des deux années précédentes. Face à cette croissance exponentielle des données, les géants du web au premier rang desquels Yahoo (mais aussi Google et Facebook), ont été les premiers à proposer une alternative aux solutions traditionnelles de bases de données et d'analyse. Les solutions sont nombreuses et permettent d'optimiser les temps de traitement sur des données à grande échelle. Parmi celles–ci, on en trouve les infrastructures de serveurs pour distribuer les traitements sur des dizaines, centaines, voire milliers de nœuds, le stockage des données en mémoire et les bases de données NoSQL**.**

Les bases de données NoSQL conviennent pour plusieurs cas d'usage. Elles sont adaptées pour stocker et retrouver de larges volumes de données. Savoir les manipuler est une compétence très recherchée par les entreprises, car celles–ci croulent aujourd'hui sous le poids des données.

C'est dans cette veine d'acquisition de compétences, que nous avons eu à réaliser un TP sur les bases de données Nosql ayant pour thème : Réalisation d'un cluster MongoDB simple pour le sharding, la réplication, la sauvegarde et la restauration des données à l'aide de Docker.

Tout au long de ce document, nous présenterons les éléments utilisés et les étapes suivies pour la réalisation du travail demandé.

Partie I : Présentation des outils pour la réalisation du TP

I- Présentation des bases de données NoSQL

Les bases de données NoSQL sont désignées indifféremment comme « non relationnelles » ou « non SQL » pour souligner le fait qu'elles peuvent gérer d'importants volumes de données non structurées et évoluant rapidement, et de manière différente par rapport à une base de données relationnelle (SQL) avec lignes et tables.

Les technologies NoSQL ont vu le jour dans les années 1960, sous différentes appellations, mais connaissent un regain de popularité alors que le paysage des données évolue et que les développeurs doivent s'adapter pour gérer la masse de données générées par le cloud, les applications mobiles, les réseaux sociaux et le Big Data.

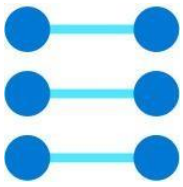
Les bases de données NoSQL aident les professionnels de l'informatique et les développeurs à faire face aux nouveaux défis liés à la diversité toujours croissante des données et des modèles, et sont particulièrement efficaces pour gérer des données imprévisibles, souvent à des vitesses de requête extrêmement rapides. On en distingue quatre principaux types : paire clé–valeur, orientée colonne, orientée graphe, **et** orientée document.

Chacune de ces catégories possède un attribut unique et des limites spécifiques. Toutefois aucun de ces quatre types de bases de données ne permet de résoudre n'importe quel problème. Il est nécessaire de choisir la base de données adéquate en fonction du cas d'usage.

1- Présentation des types de base de données NoSQL

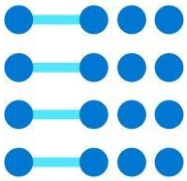
a- Les bases de données de type clé-valeur

Les bases de données de type clé–valeur stockent des paires de clés et de valeurs à l'aide d'une table de hachage. Les types clé–valeur sont particulièrement adaptés lorsqu'une clé est connue et que la valeur associée à la clé est inconnue.



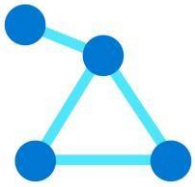
b- Les bases de données de type colonne

Les bases de données en colonnes, en colonnes larges ou en familles de colonnes stockent efficacement les données et interrogent les lignes de données éparses, et offrent la possibilité d'interroger les colonnes spécifiques d'une base de données.



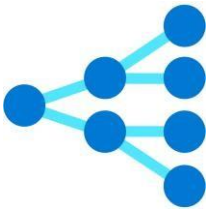
c- Les bases de données de type graphe

Les bases de données graphes utilisent un modèle basé sur les nœuds et les bords pour représenter les données interconnectées (relations entre membres d'un réseau social, par exemple), et offrent un stockage et une navigation facilités en présence de relations complexes.



d- Les bases de données de type document

Les bases de données de documents étendent le concept de base de données clé–valeur en organisant des documents entiers dans des groupes appelés collections. Elles prennent en charge les paires clé–valeur imbriquées et autorisent les requêtes sur tous les attributs d'un document.



Pour la réalisation de ce TP, nous avons choisi MongoDB qui est une base de données de type document.

2- Présentation de MongoDB

MongoDB est une base de données NoSQL orientée documents, utilisée pour le stockage de gros volumes de données. Au lieu d'utiliser des tables et des lignes comme dans les bases de données relationnelles traditionnelles, MongoDB utilise des collections et des documents.

Image de MongoDB

Les documents se composent de paires clé–valeur qui constituent l'unité de base des données dans MongoDB. Les collections contiennent des ensembles de documents et de fonctions qui sont l'équivalent des tables des bases de données relationnelles.

MongoDB présente plusieurs avantages à savoir :

- **La haute disponibilités grâce à la réplication et le basculement intégrés**
- **L'évolutivité horizontale avec sharding natif**
- **La sécurité de bout en bout**
- **La validation et exploration de schémas de documents natives avec Compass (Client MongoDB)**
- **Des outils de gestion disponible pour l'automatisation, la surveillance et la sauvegarde**
- **Une base de données entièrement élastique en tant que service avec de meilleures pratiques intégrées**

Pour mener à bien ce projet, il était question d'utiliser un outil de virtualisation pour la création du cluster. Nous avions le choix entre VirtualBox et Docker. Notre choix s'est porté sur Docker pour plusieurs raisons que nous présenterons dans la section suivante.

II- Présentation de Docker

Docker est une plateforme logicielle open source permettant de créer, de déployer et de gérer des containers d'applications virtualisées sur un système d'exploitation. Les services ou fonctions de l'application et ses différentes bibliothèques, fichiers de configuration, dépendances et autres composants sont regroupés au sein du container. Chaque container exécuté partage les services du système d'exploitation. Avant d'aller plus loin, essayons de savoir ce qu'est un container.

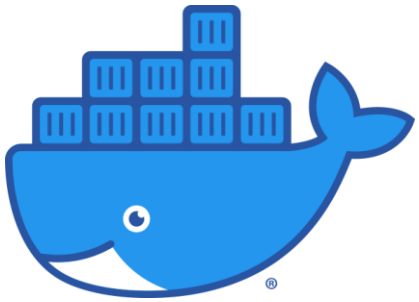
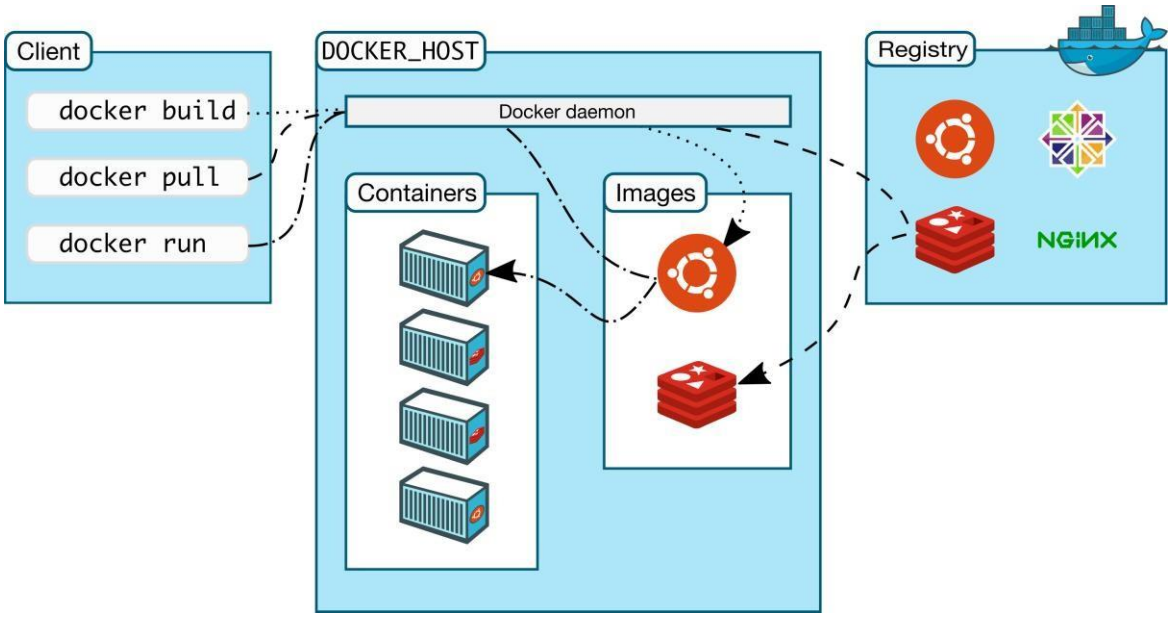


Image de Docker

Un container est un ensemble de processus logiciels léger et indépendant, regroupant tous les fichiers nécessaires à l'exécution des processus : code, runtime, outils système, bibliothèque et paramètres. Ils sont donc proches des machines virtuelles, mais présentent un avantage important. Alors que la virtualisation consiste à exécuter de nombreux systèmes d'exploitation sur un seul et même système, les containers se partagent le même noyau de système d'exploitation et isolent les processus de l'application du reste du système. Ils sont nettement plus efficients qu'une machine virtuelle en termes de consommation des ressources système.

Nous aborderons dans la section suivante, le fonctionnement de Docker raisons qui nous ont poussés à utiliser Docker au lieu de VirtualBox.

1- Fonctionnement de Docker



Architecture de Docker

Docker utilise une architecture client–serveur. Le client Docker communique avec le démon Docker, qui se charge de construire, d'exécuter et de distribuer nos containers Docker. Le client et le démon Docker peuvent fonctionner sur le même système, ou nous pouvons connecter un client Docker à un démon Docker distant. Le client et le démon Docker communiquent à l'aide d'une API REST, via des sockets UNIX ou une interface réseau. Un autre client Docker est Docker Compose, qui nous permet de travailler avec des applications composées d'un ensemble de containers.

- Démon Docker

Le démon Docker écoute les demandes de l'API Docker et gère les objets Docker tels que les images, les conteneurs, les réseaux et les volumes. Un démon peut également communiquer avec d'autres démons pour gérer les services Docker.

- Client Docker

Le client Docker est le principal moyen par lequel de nombreux utilisateurs de Docker interagissent avec Docker. Lorsque vous utilisez des commandes telles que *docker run*, le client envoie ces commandes au démon Docker qui les exécute. La commande docker utilise l'API de Docker. Le client Docker peut communiquer avec plus d'un démon.

- Registres Docker

Un registre Docker stocke les images Docker. Docker Hub est un registre public que tout le monde peut utiliser, et Docker est configuré pour rechercher les images sur Docker Hub par défaut.

- Images

Une image est un modèle en lecture seule contenant des instructions pour créer un conteneur Docker. Souvent, une image est basée sur une autre image, avec quelques personnalisations supplémentaires. Par exemple, nous pouvons construire une image basée sur l'image Ubuntu, mais qui installe le serveur web Apache et notre application, ainsi que les détails de configuration nécessaires au fonctionnement de notre application.

2- Avantages de Docker

Docker présente plusieurs avantages à savoir :

- **Flexibilité : Toutes les applications peuvent être transformées en des containers**
- **Légèreté : Contrairement à la virtualisation classique,** Docker **exploite et partage le kernel du système d'exploitation de l'hôte, ce qui le rend très efficace en termes d'utilisation des ressources du système**

- **Portabilité : Il est possible de créer, déployer et démarrer des containers sur son ordinateur, celui de ses clients ou un serveur distant**
- **Adaptation : L'installation et la désinstallation de containers ne dépend pas des autres containers installés. Ce qui permet de mettre à jour ou remplacer un container sans modifier les autres**
- **Scalabilité : Dupliquer un container est extrêmement simple, ce qui permet de réaliser de la scalabilité horizontale aisément**
- **Sécurité : Par défaut, Docker crée des containers en appliquant des règles de sécurité strictes et isole les processus.**

Docker est certes une technologie très efficace pour la gestion de containers uniques. Toutefois, à mesure qu'augmente le nombre de containers et d'applications conteneurisées, la gestion et l'orchestration se complexifient. Au final, l'on doit prendre du recul et regrouper plusieurs containers pour assurer la distribution des services vers tous nos containers.

Les outils ayant été présentés, nous pouvons entrer de plain–pied dans la réalisation du TP.

Partie II : Réalisation du TP

Notre TP consistait à réaliser un cluster de cinq (5) machines avec MongoDB sur Docker en implémentant les fonctionnalités suivantes : le sharding, la réplication, la sauvegarde et la restauration de données.

De prime abord, il serait idéal de définir ces différentes fonctionnalités.

I- Présentation des fonctionnalités à implémenter

a- Le sharding

Le sharding est une méthode de distribution des données sur plusieurs machines. MongoDB utilise le sharding pour prendre en charge des déploiements avec de très grands ensembles de données et des opérations à haut débit.

Un cluster Sharded MongoDB se compose des éléments suivants :

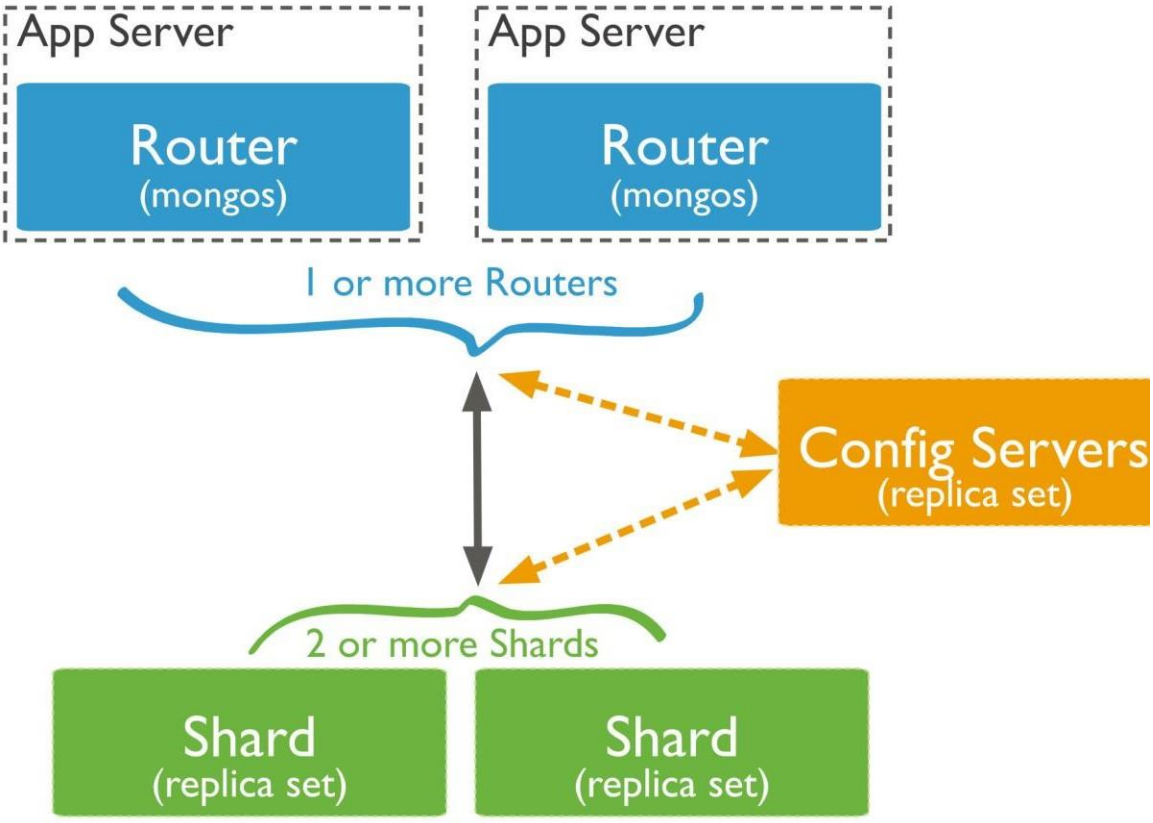
– Shard **: chaque Shard contient un sous-ensemble des données shardées. Chaque Shard peut être déployé comme un ensemble de répliques.**

– Mongos **: Le mongos agit comme un routeur de requêtes, fournissant une interface entre les applications clientes et le cluster Sharded. À partir de MongoDB 4.4, les mongos peuvent prendre en charge les lectures couvertes pour minimiser les latences.**

– Serveurs de configuration **: Les serveurs de configuration stockent les métadonnées et les paramètres de configuration du cluster.**

MongoDB partage les données au niveau de la collection, en distribuant les données de la collection à travers les shards du cluster.

Sur la figure ci–dessous se trouve une illustration du fonctionnement du sharding dans MongoDB.



Configuration d'un Sharding sur MongoDB

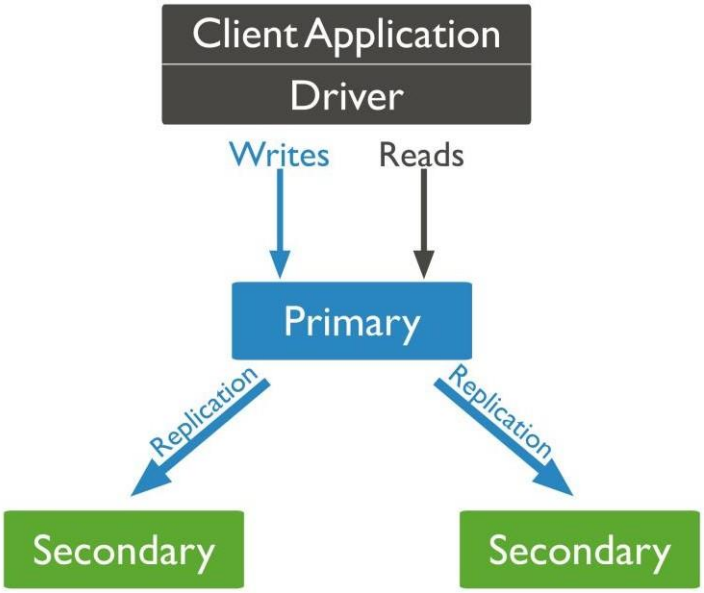
b- La réplication

La réplication assure la redondance et augmente la disponibilité des données. Grâce à des copies multiples des données sur différents serveurs de base de données, la réplication offre un niveau de tolérance aux pannes en cas de perte d'un seul serveur de base de données.

Dans certains cas, la réplication peut fournir une capacité de lecture accrue car les clients peuvent envoyer des opérations de lecture à différents serveurs. Le maintien de copies de données dans différents centres de données peut augmenter la localité et la disponibilité des données pour les applications distribuées. Vous pouvez également conserver des copies supplémentaires à des fins spécifiques, telles que la reprise après sinistre, la création de rapports ou la sauvegarde.

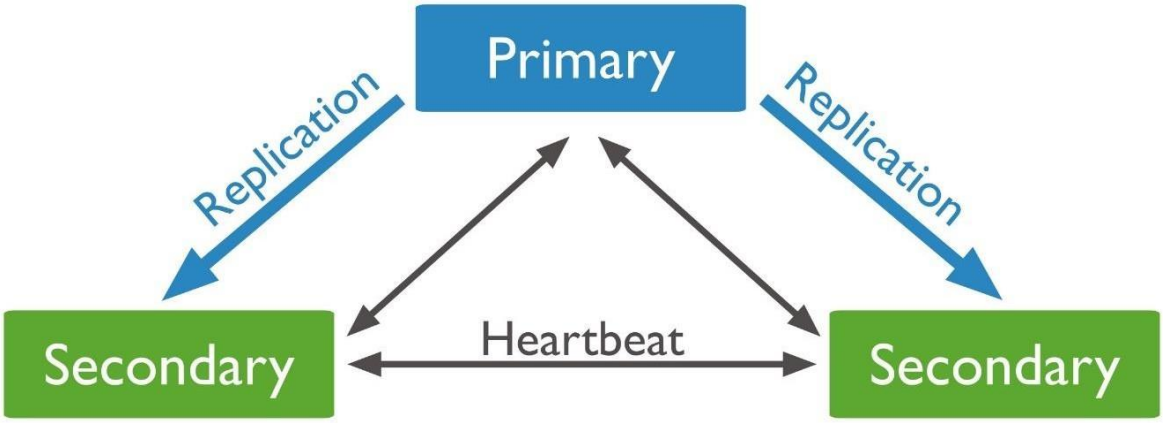
Un groupe de *replica set* **dans MongoDB est un groupe d'instances de mongod qui maintiennent le même ensemble de données. Un ensemble de** *replica set* **contient plusieurs nœuds porteurs de données et éventuellement un nœud arbitre. Parmi les nœuds porteurs de données, un et un seul membre est considéré comme le nœud principal, tandis que les autres nœuds sont considérés comme des nœuds secondaires.**

Le nœud primaire reçoit toutes les opérations d'écriture. Un ensemble de répliques ne peut avoir qu'un seul primaire capable de confirmer les écritures avec la préoccupation d'écriture *{w : "majority"}* **; bien que dans certaines circonstances, une autre instance de mongod puisse transitoirement se croire également primaire. Le primaire enregistre toutes les modifications apportées à ses ensembles de données dans son journal des opérations, c'est–à–dire Oplog. Pour plus d'informations sur le fonctionnement du nœud primaire, voir l'image ci–dessous.**



Configuration d'un replicaset sur MongoDB

Les secondaires reproduisent l'Oplog du primaire et appliquent les opérations à leurs ensembles de données de façon à ce que les ensembles de données des secondaires reflètent l'ensemble de données du primaire de manière asynchrone. Si le primaire est indisponible, un secondaire éligible organise une élection pour élire lui–même le nouveau primaire. Pour plus d'informations sur les membres secondaires, voir l'image ci–dessous.



Communication entre les nœuds d'un replicaset sur MongoDB

c- La sauvegarde et la restauration des données

La sauvegarde et la restauration des données est le processus de sauvegarde de nos données en cas de perte et de mise en place de systèmes sécurisés qui nous permettent de restaurer nos données en conséquence. La sauvegarde des données nécessite la copie et l'archivage des données pour les rendre accessibles en cas de corruption ou de suppression de données. Nous ne pouvons restaurer des données d'une date antérieure que si nous les avons sauvegardées.

Abordons maintenant la dernière partie de ce document qui concerne les étapes à la réalisation de ce TP.

II- Différentes étapes pour la réalisation du TP

Nous avons utilisé Windows comme système d'exploitation.

- Installation de Docker

- vérifier la compatible de notre système avec Docker en consultant la liste de compatibilité sur le site web de Docker https://www.docker.com/.

-Téléchargez l'installateur Docker pour Windows depuis le site web de Docker.

-Double-cliquez sur le fichier téléchargé pour lancer l'installation.

-Suivez les instructions à l'écran pour installer Docker sur votre système.

Une fois l'installation terminée, ouvrez un terminal et exécutez la commande suivante dockter – version.

En cas d'installation correcte, cette commande donne la version de docker installée.

Docker Compose est un outil qui permet de décrire (dans un fichier YAML) et gérer (en ligne de commande) plusieurs containers comme un ensemble de services interconnectés.

C'est à l'intérieur de ce fichier que nous avons eu à configurer nos containers et lecluster.

Trois ficher docker-compose.yml on été nécessaire dans nos configurations ; le premier pour les config, le second pour le sharding et le dernier pour le Mongos.

-Le mongos : représente l'intermédiaire entre le client et le système, il est le nœud sur lequel l'utilisateur effectuera ses requêtes.

```yaml
version: '3'

services:

  mongos:
    container_name: mongos
    image: mongo
    command: mongos --configdb cfgrs/192.168.110.143:40001,192.168.110.143:40002,192.168.110.143:40003 --bind_ip 0.0.0.0 --port 27017
    ports:
      - 60000:27017
```

- Les config permettent la répartition des données sur les différent sharding présentent dans le système.

```yaml
version: '3'

services:

  cfgsvr1:
    container_name: cfgsvr1
    image: mongo
    command: mongod --configsvr --replSet cfgrs --port 27017 --dbpath /data/db
    ports:
      - 40001:27017
    volumes:
      - cfgsvr1:/data/db

  cfgsvr2:
    container_name: cfgsvr2
    image: mongo
    command: mongod --configsvr --replSet cfgrs --port 27017 --dbpath /data/db
    ports:
      - 40002:27017
    volumes:
      - cfgsvr2:/data/db
```

Aperçu du fichier de config

-Jeu de replication du config : permet d'effectuer la réplication

-

```
mongosh mongodb://192.168.110.143:40001
```
```
rs.initiate(
  {
    _id: "cfgrs",
    configsvr: true,
    members: [
      { _id : 0, host : "192.168.110.143:40001" },
      { _id : 1, host : "192.168.110.143:40002" },
      { _id : 2, host : "192.168.110.143:40003" }
    ]
  }
)

rs.status()
```

-configuration des shards : nous avons configurer deux shards, chacun comportant trois nœuds. Les données sont ainsi répliquées sur les nœuds des différents shards.

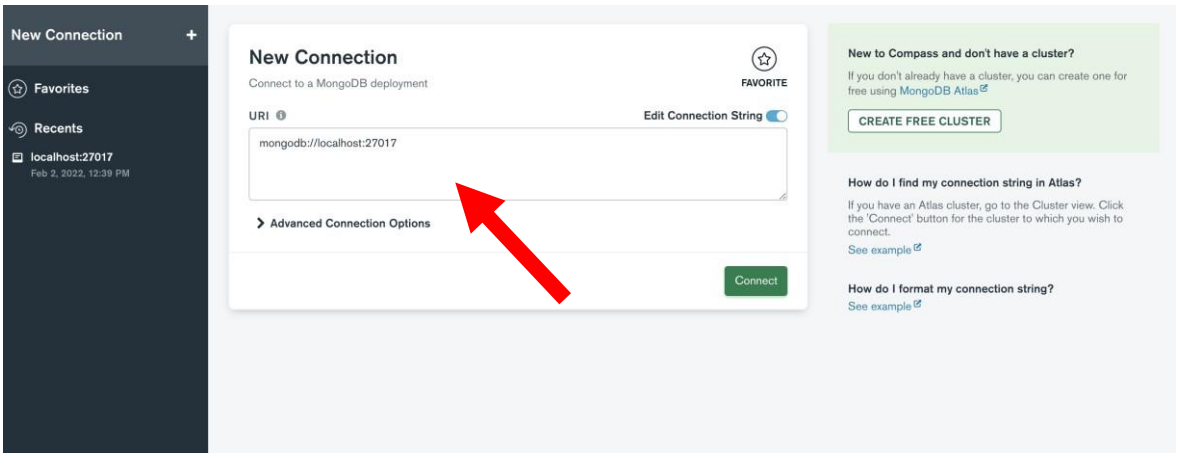
```
version: '3'

services:

  shard1svr1:
    container_name: shard1svr1
    image: mongo
    command: mongod --shardsvr --replSet shard1rs --port 27017 --dbpath /data/db
    ports:
      - 50001:27017
    volumes:
      - shard1svr1:/data/db

  shard1svr2:
    container_name: shard1svr2
    image: mongo
    command: mongod --shardsvr --replSet shard1rs --port 27017 --dbpath /data/db
    ports:
      - 50002:27017
    volumes:
      - shard1svr2:/data/db
```

Un aperçu du fichier de shard

- **Connection au cluster dans le client Compass pour MongoDB**

Exécutons la commande à l'endroit indiqué sur la figure :

```
mongodb://nomUtilisateur:motDePasse@nomMaitre:27017,nomExclave1:27017,nomExclave2:27017,nomExclave3:27017,nomExclave4:27017/nomBaseDeDonnee?replicaSet=nomDuReplicaSet
```



Client de MongoDB Compass

- **Sauvegarde**

Pour effectuer de la sauvegarde dans MongoDB, il faut se connecter au nœud maitre et exécuter la commande ci–dessous :

```
$ docker exec <mongodb container> sh -c 'mongodump --authenticationDatabase admin -u <nomUtilisateur> -p <motDePasse > --db <nomBD> --archive' > db.dump
```

- **Restauration**

Pour effectuer de la restauration dans MongoDB, il faut se connecter au nœud maitre et exécuter la commande ci–dessous :

```
$ docker exec <mongodb container> sh -c 'mongorestore --authenticationDatabase admin -u <nomUtilisateur> -p <motDePasse > --db <nomBD> --archive' < db.dump
```

Webographie

– Base de données NoSQL - Qu'est-ce que NoSQL ? : https://azure.microsoft.com/fr-fr/overview/nosql-database/

– NoSQL : Tout comprendre sur les bases de données non relationnelles : https://datascientest.com/nosql

– Qu'est-ce que MongoDB ? : https://www.mongodb.com/fr-fr/what-is-mongodb

– Docker : tout savoir sur la plateforme de containérisation : https://www.lebigdata.fr/docker-definition

– Docker : Fonctionnement, avantages, et limitations : https://stacktraceback.com/docker-sont-fonctionnement-les-avantages-et-les-limitations/

– Docker overview: https://docs.docker.com/get-started/overview/

– Sharding: https://www.mongodb.com/docs/manual/sharding/#:~:text=Sharding%20is%20a%20method%20for,capacity%20of%20a%20single%20server.

– Réplication : https://www.mongodb.com/docs/manual/replication/

– Comment installer et utiliser Docker sur Ubuntu 20.04 : https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04-fr

Conclusion

En conclusion, nous pouvons retenir que les bases de données NoSQL offrent plusieurs fonctionnalités que les bases de données relationnelles n'offrent pas.

Ce TP nous a été bénéfique et nous a permis de comprendre les définitions que nous avons vu au cours.

Nous avons certes rencontré des difficultés mais nous avons pu les surmonter.