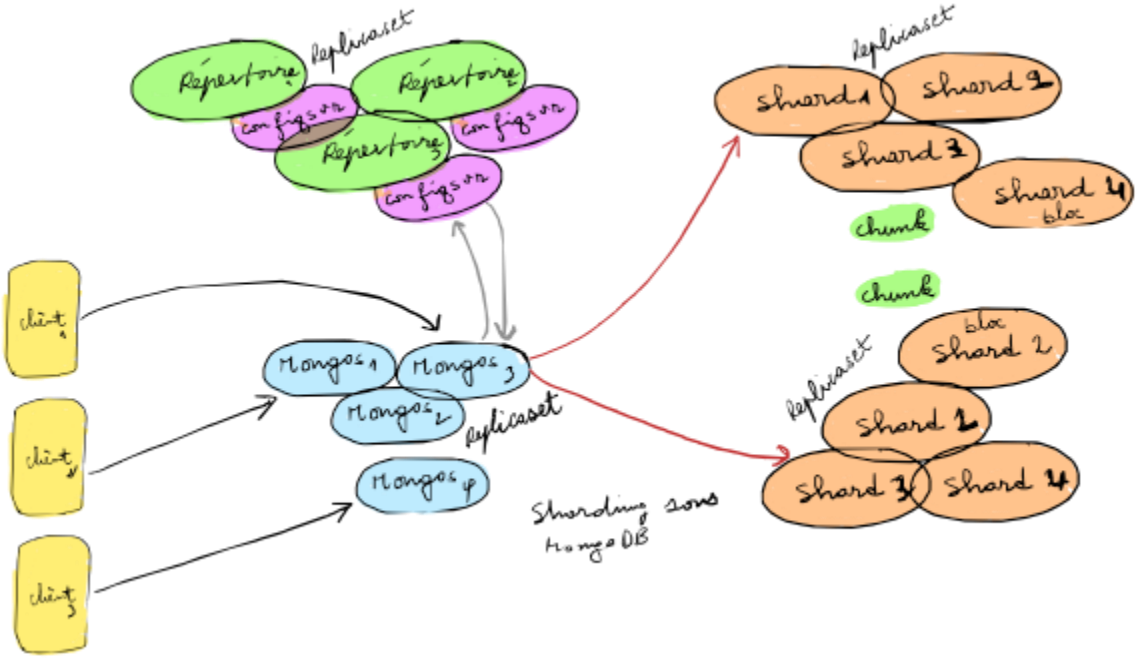
Le partionnement sous MongoDB

Dans ce TP, vous allez mettre en œuvre un système de sharding sous MongoDB.

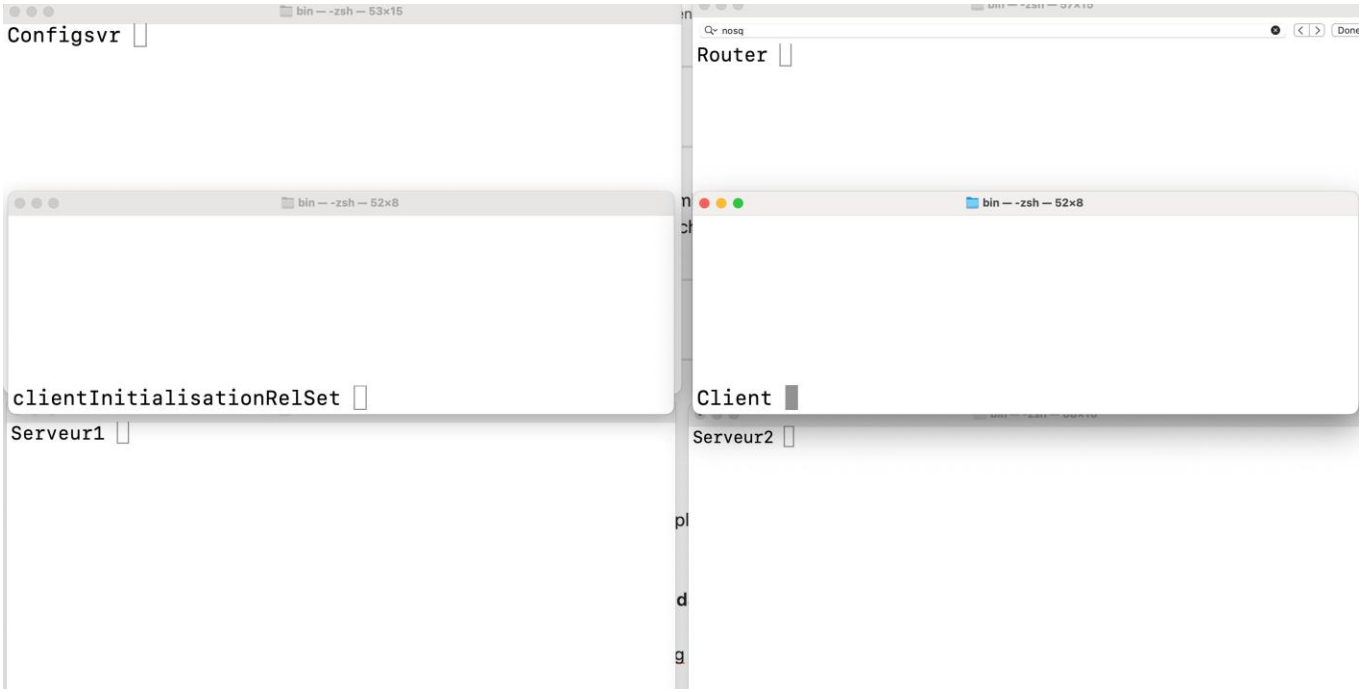
Je vais vous décrire pas à pas comment mettre en place un cluster MongoDB composé de trois éléments essentiels : **un serveur de configuration (config server)**, **un routeur (mongos)**, **et**

- **deux shards (serveur1 et serveur2)**.
- Je vous rappelle qu'un **replica set** constitue l'unité de base d'une architecture tolérante aux pannes et capable de passer à l'échelle. Dans ce TP, nous allons principalement nous concentrer sur le **partitionnement des données** (sharding). Toutefois, il est important de noter que le **config server doit impérativement être répliqué**, même si nous ne travaillons pas directement sur ce point ici.

En effet, si le serveur de configuration tombe en panne, l'ensemble du cluster devient inopérant. C'est ce serveur qui conserve la **métadonnée de partitionnement** : il maintient les informations permettant de savoir **dans quel fragment (chunk)** se trouvent les données. C'est donc le point central de la gestion du sharding. L'architecture globale du système que nous allons mettre en place est illustrée dans la figure ci-dessous.



Commencez par ouvrir six terminaux et donnez-leur un nom afin de vous y retrouver facilement. Vous devriez obtenir quelque chose de similaire à l'exemple illustré dans la figure suivante.



Commencez par créer trois répertoires de stockage : un pour le répertoire et deux pour les shards. Démarrez un serveur de configuration en exécutant la commande suivante (N'oubliez pas d'initialiser le réplica set, même si vous ne disposez que d'un seul répertoire (cf. le TP sur la réplication) :

```
mongod --configsvr --replSet replicaconfig --dbpath configsvrdb --port 27019
```

- **mongod** : Lance le serveur MongoDB.
- **--configsvr** : Indique que ce serveur joue le rôle de *config server* dans un cluster sharded.
- **--replSet replicaconfig** : Spécifie le nom du réplica set auquel appartient ce config server (ici *replicaconfig*).
- **--dbpath configsvrdb** : Chemin du répertoire où seront stockés les fichiers de données du config server.
- **--port 27019** : Définit le port sur lequel le config server écoute.

Lancez un routeur (*mongos*) en exécutant la commande suivante :

```
mongos --configdb replicaconfig/localhost:27019
```

- **mongos** : Démarre le processus *mongos*, qui agit comme routeur dans un cluster sharded.
- **--configdb replicaconfig/localhost:27019** : Indique l'adresse et le réplica set des *config servers* que le *mongos* utilisera pour obtenir la configuration du cluster.

Je vous invite à consulter la documentation officielle de MongoDB pour vérifier si le routeur (*mongos*) doit être répliqué ou non. Cela dépend de la version que vous utilisez. Dans les toutes premières versions, aucun réplica set n'était requis. Dans les versions plus récentes, un

réplica set est obligatoire pour les shards et parfois pour les *mongos*, et il est possible que, dans le futur, cela s'applique également au routeur.

Démarrez les deux shards et initialisez leurs réplica sets en exécutant les commandes suivantes :

```
mongod --replSet replicashard1 --dbpath serv1/ --shardsvr --port 20004
mongod --replSet replicashard2 --dbpath serv2/ --shardsvr --port 20005
```

- **--replSet** : Nom du réplica set pour chaque shard.
- **--dbpath** : Répertoire de stockage des données du shard.
- **--shardsvr** : Indique que le serveur joue le rôle de shard dans un cluster.
- **--port** : Port d'écoute du shard.

Connectez-vous au routeur (*mongos*) et ajoutez les deux shards avec les commandes suivantes :

```
sh.addShard("replicashard1/localhost:20004")
sh.addShard("replicashard2/localhost:20005")
```

```
sh.enableSharding("mabasefilms")
```

Les bases de données gérées par MongoDB ne sont pas partitionnées (*shardées*) par défaut. Il est donc nécessaire de l'activer explicitement en exécutant la commande suivante :

Ici, `mabasefilms` correspond au nom de la base de données, qui doit être précisé dans le programme Python utilisé pour insérer les films.

Les collections dans les bases de données ne sont pas non plus partitionnées (*shardées*) par défaut. Il est donc nécessaire de l'activer explicitement en exécutant la commande suivante :

```
sh.shardCollection("mabasefilms.films", { "titre": 1 })
```

- `sh.shardCollection` : Permet de shard(er) une collection spécifique.
- `"mabasefilms.films"` : Nom de la collection à shard(er), ici la collection `films` dans la base `mabasefilms` .
- `{"titre": 1}` : Spécifie la clé de partitionnement (*shard key*) utilisée pour distribuer les documents entre les shards.

Une fois le cluster mis en place, exécutez le programme disponible sur le site du cours. Celui-ci permet d'insérer des films générés aléatoirement. Par défaut, un million de films sont insérés, un par un. En environnement de production, si vous devez insérer un grand nombre de documents, privilégiez une insertion en masse (batch) en ajustant de préférence la taille des *chunks* pour optimiser les performances.

Étudiez les implications du partitionnement (*sharding*) dans MongoDB sur la gestion des données. Surveillez, en temps réel, la migration des *chunks* d'un serveur à un autre afin d'équilibrer la charge, puis répondez aux questions suivantes.

1. Qu'est-ce que le sharding dans MongoDB et pourquoi est-il utilisé ?

Le **sharding** est un mécanisme qui permet de **répartir les données sur plusieurs serveurs** appelés *shards*.
Il est utilisé pour :

- Gérer de très grands volumes de données,
- Améliorer les performances,
- Permettre le passage à l'échelle horizontale.

2. Quelle est la différence entre le sharding et la réplication dans MongoDB ?

- **Réplication** : copie des mêmes données sur plusieurs serveurs (tolérance aux pannes).
- **Sharding** : découpage des données entre plusieurs serveurs (répartition de la charge).

3. Quels sont les composants d'une architecture shardée (mongos, config servers, shards) ?

Une architecture shardée comprend trois éléments principaux :

- Les **shards**, qui stockent les données,
- Les **config servers**, qui stockent les informations sur la répartition des données,
- Le **mongos**, qui sert d'intermédiaire entre le client et les shards.

4. Quelles sont les responsabilités des **config servers (CSRS)** dans un cluster shardé ?

Les config servers conservent toutes les **métadonnées du cluster shardé**.
Ils savent quels chunks sont stockés sur quels shards et permettent à MongoDB de localiser les données.
➔Sans eux, le cluster shardé ne peut pas fonctionner.

5. Quel est le rôle du **mongos router** ?

- Le mongos est un **routeur de requêtes**.
- Les applications se connectent au mongos, et non directement aux shards.
- Le mongos analyse chaque requête et l'envoie automatiquement vers le ou les shards concernés.

6. Comment MongoDB décide-t-il sur quel shard stocker un document ?

MongoDB utilise la **clé de sharding** définie pour la collection.
La valeur de cette clé permet de déterminer dans quel chunk se trouve le document, puis sur quel shard ce chunk est stocké.

7. Qu'est-ce qu'une **clé de sharding** et pourquoi est-elle essentielle ?

La clé de sharding est un champ du document choisi pour **répartir les données entre les shards**. Elle est essentielle car une mauvaise clé peut entraîner un déséquilibre du cluster et de mauvaises performances.

8. Quels sont les critères de choix d'une bonne clé de sharding ?

Une bonne clé de sharding doit :

- Répartir les données de manière équilibrée,
- Éviter les valeurs répétitives,
- Être utilisée fréquemment dans les requêtes,
- Ne pas être monotone.

9. Qu'est-ce qu'un **chunk** dans MongoDB ?

Un chunk est une **portion de données** appartenant à une collection shardée.
Chaque chunk contient un ensemble de documents et est stocké sur un shard précis.

10. Comment fonctionne le **splitting** des chunks ?

Lorsque la taille d'un chunk dépasse une certaine limite, MongoDB le **divise automatiquement** en deux chunks plus petits.

➔ Cela permet de conserver des chunks de taille raisonnable.

11. Que fait le **balancer** dans un cluster shardé ?

Le balancer est un processus automatique chargé de **rééquilibrer les chunks** entre les shards. Il permet d'éviter qu'un shard contienne beaucoup plus de données que les autres.

12. Quand et comment le balancer déplace-t-il des chunks ?

Le balancer fonctionne en arrière-plan.
Lorsqu'il détecte un déséquilibre, il déplace certains chunks d'un shard vers un autre, sans interrompre le service.

13. Qu'est-ce qu'un **hot shard** et comment l'éviter ?

Un hot shard est un shard qui reçoit trop de requêtes ou trop d'écritures. On peut l'éviter en choisissant une clé de sharding bien distribuée ou en utilisant une clé hashée.

14. Quels problèmes une clé de sharding monotone peut-elle engendrer ?

Une clé monotone (comme une date croissante) peut provoquer l'envoi de toutes les nouvelles données vers un seul shard. Cela entraîne un hot shard et dégrade les performances globales.

15. Comment activer le sharding sur une base de données et sur une collection ?

Pour activer le sharding, il faut d'abord l'activer sur la base, puis sur la collection concernée.

sh.enableSharding("maBase")

sh.shardCollection("maBase.maCollection", { champ: 1 })

16. Comment ajouter un nouveau shard à un cluster MongoDB ?

Pour ajouter un shard au cluster MongoDB, on utilise la commande suivante :

sh.addShard("replicaSet/host:port")

17. Comment vérifier l'état du cluster shardé (commandes usuelles) ?

L'état du cluster peut être vérifié avec des commandes comme :

sh.status()

db.stats()

18. Dans quels cas faut-il envisager d'utiliser un **hashed sharding key** ?

- Une clé hashée est utile lorsque les valeurs de la clé sont monotones.
- Le hachage permet de répartir les données de manière plus uniforme

19. Dans quels cas faut-il privilégier un **ranged sharding key** ?

Une clé ranged est préférable lorsque les requêtes portent souvent sur des intervalles, par exemple des dates ou des plages de valeurs.

20. Qu'est-ce que le **zone sharding** et quel est son intérêt ?

Le zone sharding permet d'affecter certains types de données à des shards spécifiques, par exemple selon une zone géographique.

➔ Cela permet un meilleur contrôle de la localisation des données.

21. Comment MongoDB gère-t-il les requêtes multi-shards ?

Si une requête concerne plusieurs shards, MongoDB envoie la requête à chacun d'eux et regroupe les résultats avant de les retourner au client.

22. Comment optimiser les performances de requêtes dans un environnement shardé ?

Les performances peuvent être améliorées en :

- Choisissant une bonne clé de sharding,
- Indexant cette clé,
- Limitant les requêtes qui touchent plusieurs shards.

23. Que se passe-t-il lorsqu'un shard devient indisponible ?

Si un shard est indisponible mais répliqué, un nœud secondaire prend le relais.
Sinon, certaines données deviennent temporairement inaccessibles.

24. Comment migrer une collection existante vers un schéma shardé ?

Pour shard(er) une collection existante, il faut :

- Activer le sharding sur la base,
- Définir une clé de sharding,
- Activer le sharding sur la collection.

25. Quels outils ou métriques utiliser pour diagnostiquer les problèmes de sharding ?

Pour diagnostiquer les problèmes de sharding, on peut utiliser :

- La commande sh.status(),
- Les logs MongoDB,
- MongoDB Compass,
- Les outils de monitoring MongoDB.