Gloire BAYOUNDOULA
INFOA3

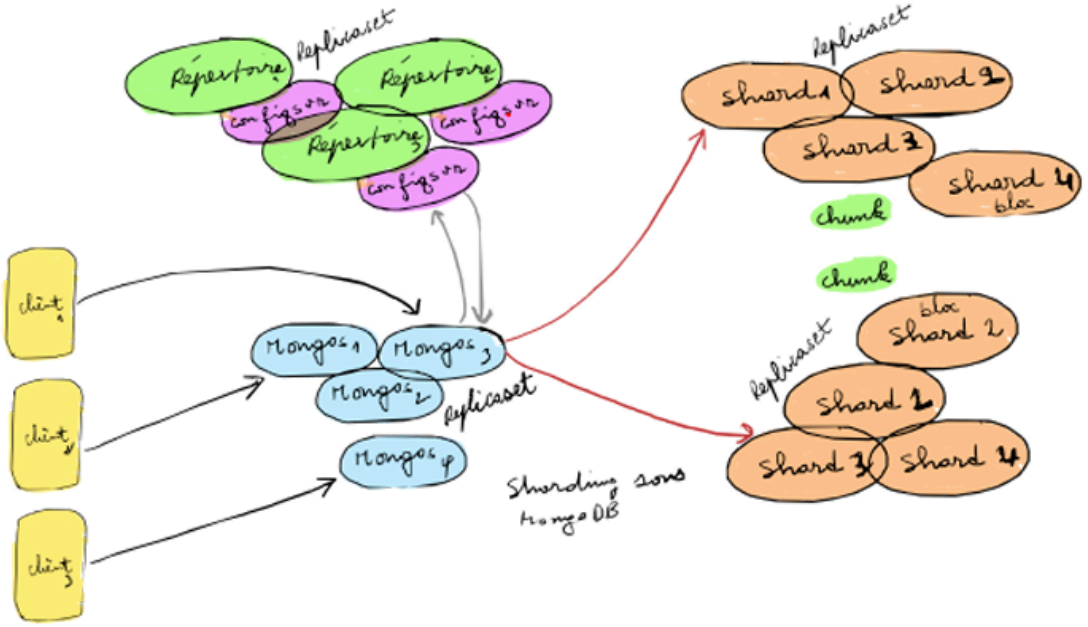
Matière : **Bases de données NoSQL**
Partie TP3: **Partitionnement avec MongoDB et Cassandra**
Enseignant : **Monsieur Samir Youcef**

Rendu TP3 - Partionnement avec MongoDB et Cassandra

Un replica set (comme vu au TP2) constitue le composant fondamental d'une architecture de base de données à la fois tolérante aux pannes et capable de monter en charge.

Dans ce TP, l'accent est mis principalement sur le partitionnement des données (sharding). Il faut néanmoins souligner que le serveur de configuration doit lui aussi être répliqué, même si cet aspect n'est pas traité en détail ici.

En cas de défaillance du serveur de configuration, c'est tout le cluster qui devient inutilisable. Ce serveur stocke en effet les métadonnées de partitionnement, c'est-à-dire les informations indiquant dans quel fragment (chunk) se trouvent les données. Il constitue donc l'élément central de la gestion du sharding. L'architecture globale du système mis en place est représentée dans la figure ci-dessous:



Manipulation (commandes à exécuter)

Pour mettre en place cette architecture, nous allons réaliser ou exécuter les commandes suivantes :

Vous aurez besoin de 6 fenêtres (terminaux ouverts). Dans mon cas, je travaille en environnement Windows (terminal Powershelle) avec MongoDB Compass que j'ai installé.

- Créer les 3 répertoires : 1 pour le repertoire et 2 pour les shards
  mkdir C:\Users\cloud\configsvrdb
  mkdir C:\Users\cloud\serv1
  mkdir C:\Users\cloud\serv2

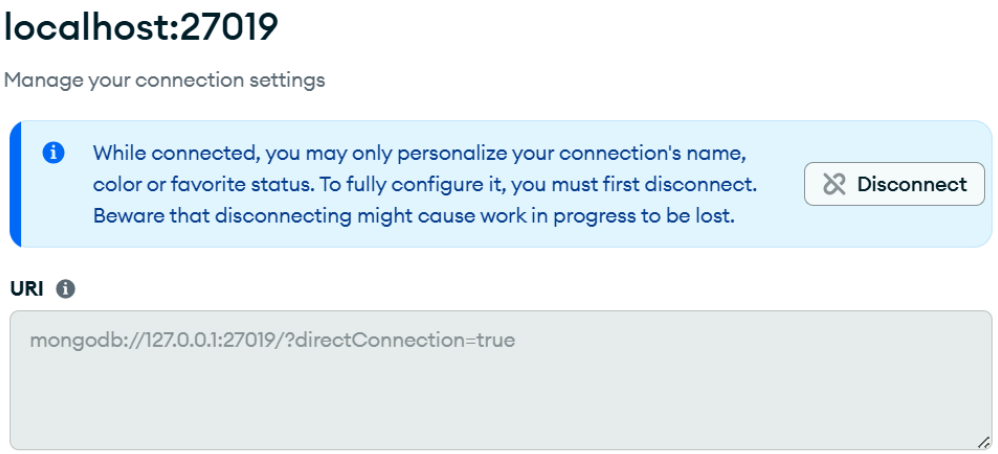
Il faut que l'ordre d'exécution dans les terminaux soit respecté

- Lancer dans le terminal 1, le repertoire :

```
& "C:\Program Files\MongoDB\Server\8.2\bin\mongod.exe" --configsvr --replSet replicaconfig --dbpath .\configsvrdb --port 27019 --bind_ip localhost
```

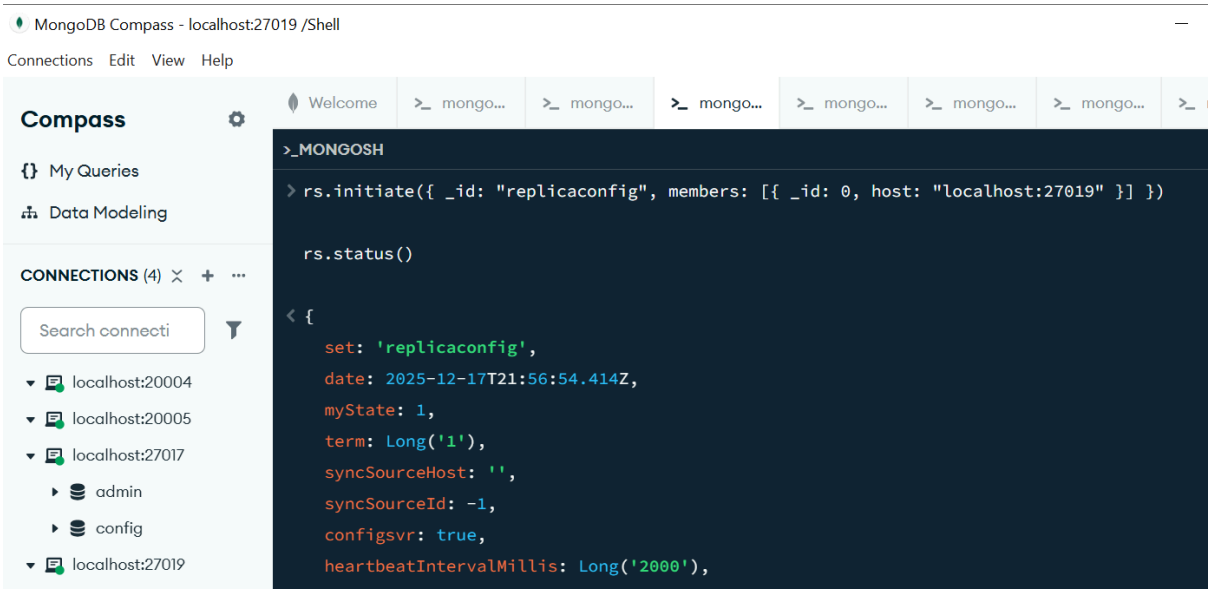
Dans MongoDB Compass, rajouter la connection

```
mongodb://localhost:27019/?directConnection=true
```



Ouvrir l'invite de commande de mongo db et taper ces deux commandes pour l'initialisation

```
rs.initiate({ _id: "replicaconfig", members: [{ _id: 0, host: "localhost:27019" }] })

rs.status()
```



- Lancer dans le terminal 2:

```
& "C:\Program Files\MongoDB\Server\8.2\bin\mongos.exe" --configdb replicaconfig/localhost:27019 --bind_ip localhost
```

Ensuite, attendre que la log affiche l'instruction :

waiting for connections on port 27017

- Terminal 3

Lancer le shard1:

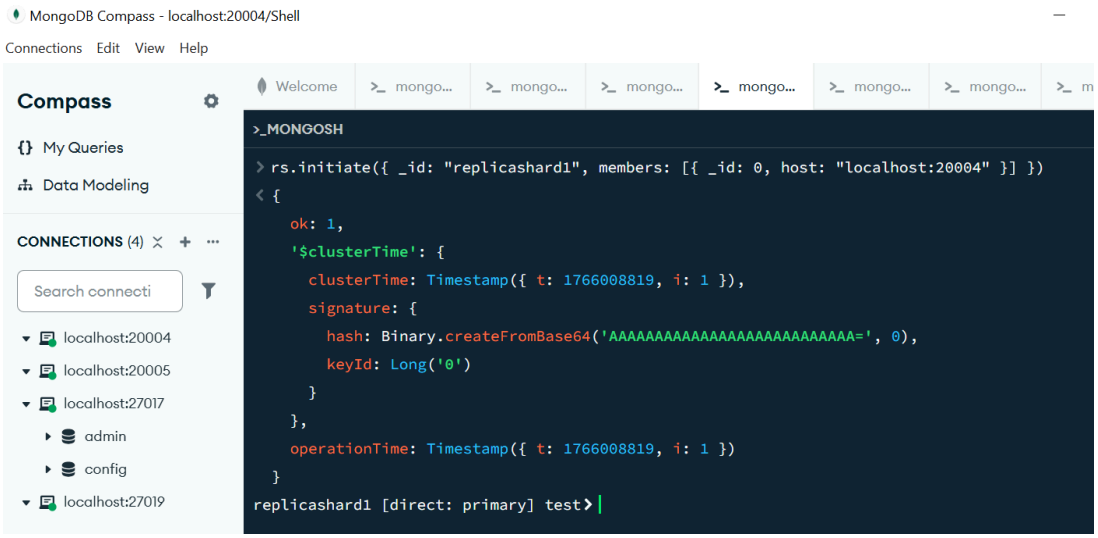
```
& "C:\Program Files\MongoDB\Server\8.2\bin\mongod.exe" --replSet replicashard1 --dbpath .\serv1 --shardsvr --port 20004 --bind_ip localhost
```

Dans MongoDB Compass, rajouter la connection

```
mongodb://localhost:20004/?directConnection=true
```

Ouvrir l'invite de commande de mongodb sur MongoDB compass de la connection `localhost:20004` et taper cette commande pour l'initialisation :

```
rs.initiate({ _id: "replicashard1", members: [{ _id: 0, host: "localhost:20004" }] })
```



Même manipulation pour le shard 2 :

- Terminal 4

Lancer le shard 2:

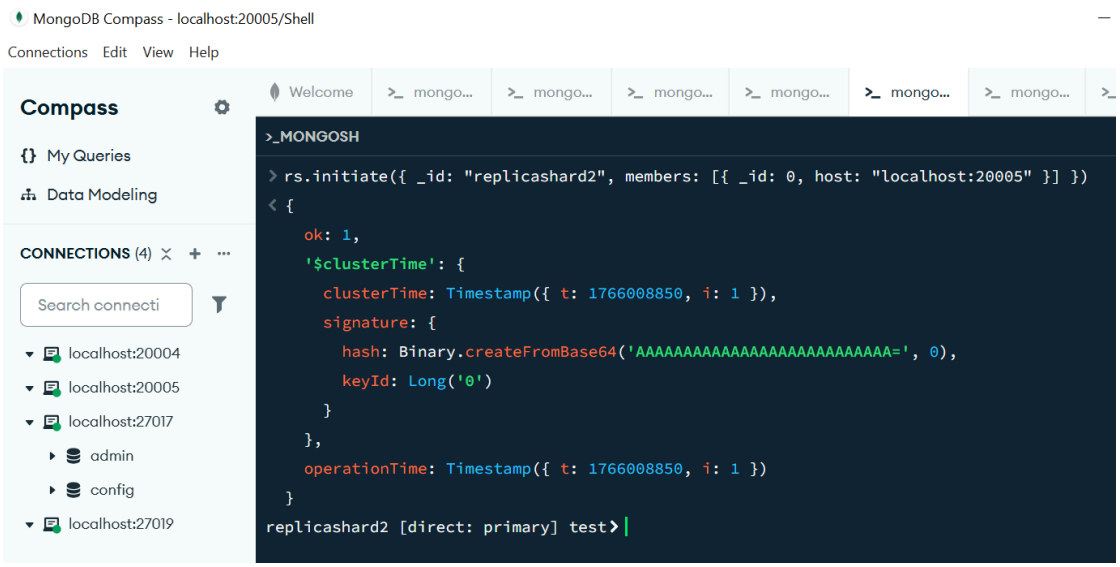
```
&    "C:\Program    Files\MongoDB\Server\8.2\bin\mongod.exe"    --replSet replicashard2    --dbpath    .\serv2    --shardsvr    --port    20005    --bind_ip localhost
```

Dans MongoDB Compass, rajouter la connection

```
mongodb://localhost:20005/?directConnection=true
```

Ouvrir l'invite de commande de mongodb sur MongoDB compass de la connection `localhost:20005` et taper cette commande pour l'initialisation :

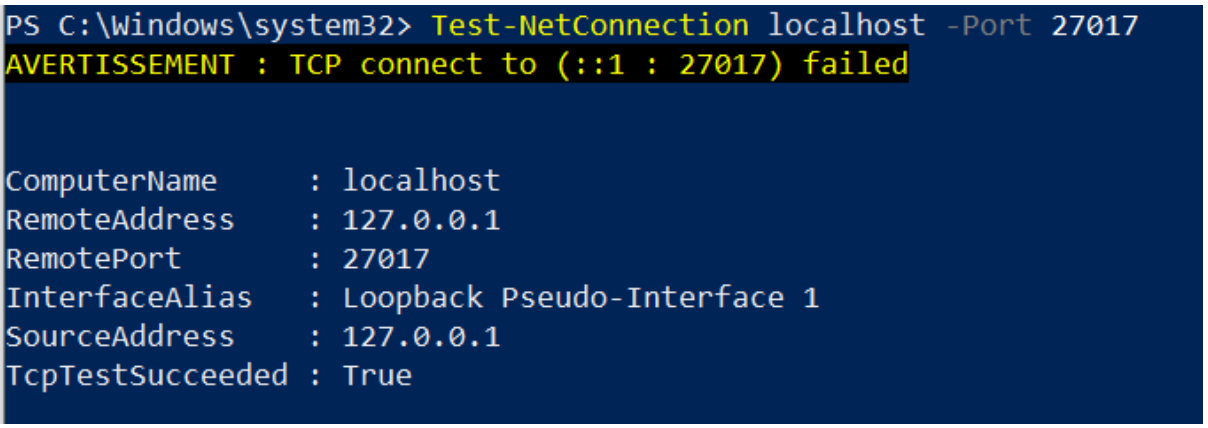
```
rs.initiate({    _id:    "replicashard2",    members:    [{    _id:    0,    host: "localhost:20005" }] })
```



- Dans le terminal 5:

  Premièrement, on teste la connexion du port 27017 dans le terminal 5 comme suit :

Test-NetConnection localhost -Port 27017



Ensuite, sur MongoDB Compass, pour la connexion 27017.

Il faut créer la connection sur MongoDB Compass pour le localhost:27017 comme suit:

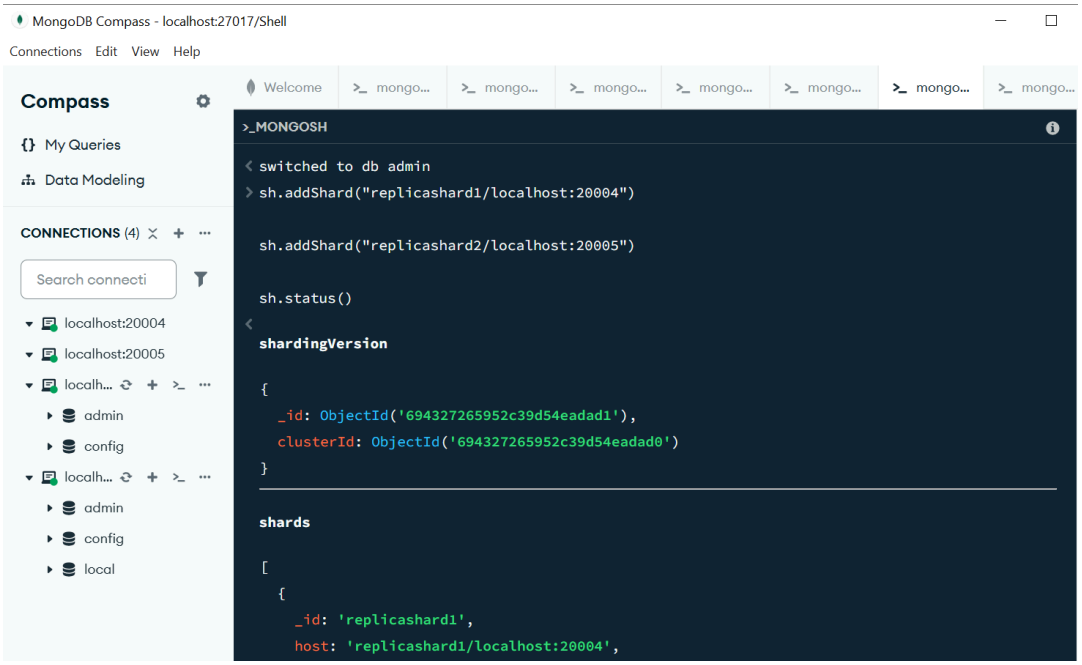
```
mongodb://localhost:27017/?directConnection=true
```

Après cela, dans l'invite de commande de MongoDBCompass taper :

```
use admin
```

Et enfin:

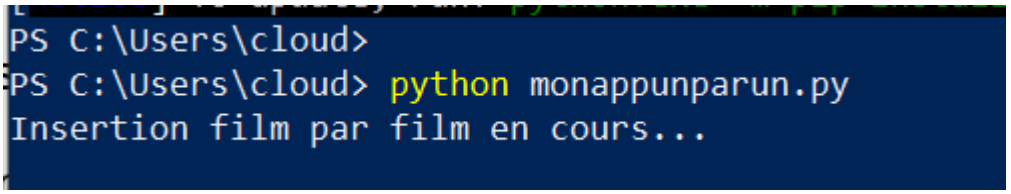
```
sh.addShard("replicashard1/localhost:20004")

sh.addShard("replicashard2/localhost:20005")

sh.status()
```



A ce stade, le cluster est créé car quand tous les 6 terminaux tournent, cela signifie que le cluster sharding est fonctionnel.

- Il faut télécharger le fichier python, et le lancer dans le terminal 6 :

```
python monappunparun.py
```



Résumé des 6 terminaux du TP MongoDB Sharding

Pour résumer :

Terminal 1 : mongod --configsvr --replSet replicaconfig --dbpath .\configsvrdb --port 27019
Rôle : Config Server (métadonnées sharding)
Port : 27019
Statut : Replica set initialisé

Terminal 2 : mongos --configdb replicaconfig/localhost:27019
Rôle : Routeur (point d'accès client)
Port : 27017
Statut : Logs sessions normales

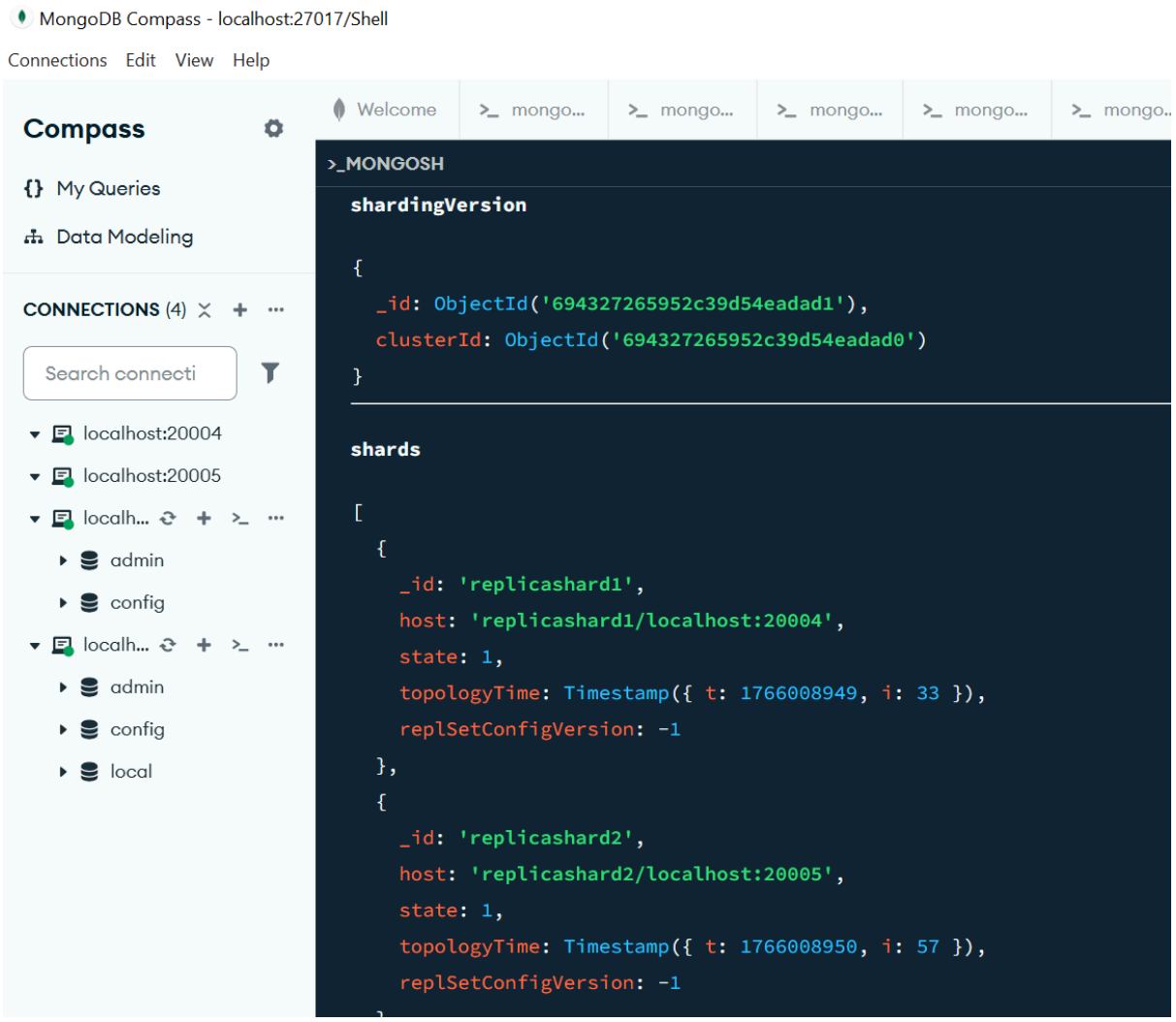
Terminal 3 : mongod --replSet replicashard1 --dbpath .\serv1 --shardsvr --port 20004
Rôle : Shard 1 (données shardées)
Port : 20004
Statut : Replica set initialisé

Terminal 4 : mongod --replSet replicashard2 --dbpath .\serv2 --shardsvr --port 20005
Rôle : Shard 2 (données shardées)
Port : 20005
Statut : Replica set initialisé

Terminal 5 : Test-NetConnection localhost -Port 27017
Rôle : Test connexion (optionnel)
Port : -
Statut : TcpTestSucceeded: True

Terminal 6 : python monappunparun.py
Rôle : Programme TP (1M films)
Port : -
Statut : Insère → sh.status() observe

Pendant que le programme python tourne dans le terminal 6, on observe dans le terminal mongodb compass du localhost 27017 en exécutant la commande sh.status() de temps en temps que le sharding évolue :



```
active mongoses

[
  {
    '8.2.2': 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': 'No recent migrations'
}
```

Réponses aux questions :

1. Qu'est-ce que le sharding dans MongoDB et pourquoi est-il utilisé ?
Le sharding est un mécanisme de partitionnement horizontal des données, où une collection est découpée en sous-ensembles (chunks) répartis sur plusieurs serveurs appelés shards. Il est utilisé pour faire face à de très gros volumes de données et à un grand nombre de requêtes en répartissant la charge, ce qui améliore la scalabilité et les performances au-delà des limites d'un seul serveur.

2. Quelle est la différence entre sharding et réplication dans MongoDB ?
D'une part, la réplication consiste à dupliquer les mêmes données sur plusieurs nœuds d'un replica set afin d'assurer la haute disponibilité et la tolérance aux pannes. Un nœud pouvant prendre le relais si un autre tombe.

Le sharding, d'autre part, ne duplique pas logiquement les données mais les distribue entre plusieurs shards. Chaque shard ne contenant qu'une partie du jeu de données pour augmenter la capacité de stockage et la scalabilité horizontale.

3. Quels sont les composants d'une architecture shardée(mongos, config servers, shards) ?
Une architecture shardée MongoDB se compose de shards qui stockent les données partitionnées et sont généralement eux-mêmes des replica sets pour la haute disponibilité.

Elle comprend aussi un ou plusieurs config servers (CSRS) qui conservent la métadonnée de sharding.

Enfin, il y'a des processus mongos (routeurs) qui reçoivent les requêtes côté client et les redirigent vers les shards appropriés.

4. Quelles sont les responsabilités des config servers (CSRS) dans un cluster shardé?
Les config servers stockent la configuration globale du cluster shardé, notamment la liste des shards, la définition des clés de sharding, la répartition des chunks et les zones éventuelles. Ils servent de source d'autorité pour les mongos et pour le balancer, qui s'appuient sur ces métadonnées pour router les requêtes et équilibrer les chunks entre shards.

5. Quel est le rôle du mongos router ?
Le mongos joue le rôle de routeur d'accès au cluster. En effet, les clients se connectent à lui comme à un serveur MongoDB classique et il se charge de déterminer quels shards sont concernés par chaque opération. Il interroge les config servers pour connaître la répartition des chunks, envoie les requêtes aux bons shards, agrège éventuellement les résultats et renvoie une réponse unifiée au client.

6. Comment MongoDB décide-t-il sur quel shard stocker un document ?
MongoDB utilise la clé de sharding définie pour la collection. Factuellement, la valeur de cette clé dans le document permet de calculer dans quel chunk le document doit aller. Chaque chunk correspond à un intervalle (ranged) ou à un hash de valeurs de clé, et la métadonnée de configuration indique sur quel shard se trouve ce chunk, ce qui permet de déterminer le shard de destination.

7. Qu'est-ce qu'une clé de sharding et pourquoi est-elle essentielle ?
La clé de sharding est un ou plusieurs champs d'un document utilisé pour distribuer les documents d'une collection entre les shards. Elle est essentielle parce qu'elle influence la répartition de la charge, le nombre de requêtes ciblées versus diffusées à tous les shards, et donc directement les performances et l'équilibrage du cluster.

8. Quels sont les critères de choix d'une bonne clé de sharding ?
Une bonne clé de sharding doit présenter une forte cardinalité et une bonne dispersion des valeurs, afin d'éviter la concentration de données et de requêtes sur un seul shard. Elle doit aussi permettre de cibler la majorité des requêtes (présente dans les filtres/index) et ne pas être strictement monotone dans le temps, pour éviter que toutes les nouvelles insertions arrivent sur le même shard.

9. Qu’est-ce qu’un chunk dans MongoDB ?
Un chunk est une plage de valeurs de clé de sharding représentant un sous-ensemble des documents d’une collection shardée. C’est l’unité minimale de déplacement pour l’équilibrage : le balancer déplace des chunks entiers d’un shard à un autre pour répartir correctement les données et la charge.

10. Comment fonctionne le splitting des chunks ?
Lorsqu’un chunk devient trop volumineux en dépassant une certaine taille configurée, MongoDB le découpe en plusieurs chunks plus petits en se basant sur la distribution des valeurs de la clé de sharding.

Ce splitting se fait automatiquement et permet ensuite au balancer de déplacer ces nouveaux chunks vers d’autres shards pour maintenir un équilibre du stockage et de la charge.

11. Que fait le balancer dans un cluster shardé ?
Le balancer est un processus qui surveille la distribution des chunks entre les shards et corrige les déséquilibres lorsqu’un shard contient significativement plus de chunks qu’un autre. Il planifie et exécute les migrations de chunks d’un shard surchargé vers un shard moins chargé afin de garder une répartition homogène des données.

12. Quand et comment le balancer déplace-t-il des chunks ?
Le balancer se déclenche lorsque la différence de nombre de chunks entre shards dépasse un seuil configuré, évalué périodiquement. Il migre un chunk en le copiant d’abord vers le shard cible, en synchronisant les écritures, puis en supprimant la copie sur le shard source une fois la migration confirmée, tout en essayant de limiter l’impact sur la production.

13. Qu’est-ce qu’un hot shard et comment l’éviter ?
Un hot shard est un shard qui reçoit beaucoup plus de trafic ou stocke beaucoup plus de données que les autres. Ce qui crée ainsi un goulot d’étranglement en termes de charge et de performances.

On l’évite principalement en choisissant une clé de sharding bien répartie qui est souvent hachée, en évitant les clés monotones et en surveillant régulièrement la distribution des chunks et des requêtes.

14. Quels problèmes une clé de sharding monotone peut-elle engendrer ?
Une clé monotone, comme un identifiant croissant ou un timestamp de création, conduit généralement à ce que toutes les nouvelles insertions se fassent sur la même extrémité de l’espace des clés. Cela concentre les écritures sur un seul shard (le hot shard), provoque un déséquilibre des chunks et peut dégrader les performances et la capacité de mise à l’échelle.

15. Comment activer le sharding sur une base et une collection ?
Pour cela, on commence par activer le sharding sur la base de données avec la commande sh.enableSharding("nomBase") exécutée depuis un mongos. Ensuite, on shard la collection voulue avec sh.shardCollection("nomBase.nomCollection", { "champCle": 1 }) ou avec une clé hachée { "champCle": "hashed" }, ce qui crée la distribution initiale des chunks.

16. Comment ajouter un nouveau shard à un cluster MongoDB ?
Pour cela, on déploie d'abord un nouveau shard, généralement un replica set configuré avec l'option --shardsvr sur les mongod concernés.

Ensuite, depuis un mongos, on exécute la commande sh.addShard("nomReplicaSet/host1:port1,...") pour déclarer ce shard dans le cluster et permettre au balancer d'y migrer des chunks.

17. Comment vérifier l'état du cluster shardé (commandes usuelles) ?
On peut utiliser sh.status() dans le shell mongo pour obtenir une vue globale du sharding : shards, bases shardées, chunks et zones éventuelles. Des commandes comme db.printShardingStatus(), la consultation des collections dans la base config, ou encore l'Atlas UI / métriques de monitoring permettent aussi de vérifier l'état, la répartition des chunks et l'activité du balancer.

18. Quand utiliser une hashed sharding key ?
Une clé de sharding hashée est adaptée quand on veut répartir de manière uniforme les écritures et le stockage, surtout s'il s'agit d'un champ monotone comme un identifiant ou une date. Le hash casse la monotonie et minimise le risque de hot shard, au prix de requêtes range moins efficaces, car les valeurs proches ne sont plus regroupées physiquement.

19. Quand privilégier une ranged sharding key ?
Une clé de sharding par plage (ranged) est appropriée lorsque les requêtes sont souvent basées sur des intervalles de valeurs (ex. recherches par date, par ID dans une plage). Elle permet de tirer profit de la localisation des données pour les requêtes range, mais impose de choisir une clé suffisamment variée pour éviter les problèmes de déséquilibre, surtout si elle est monotone.

20. Qu'est-ce que le zone sharding et quel est son intérêt ?
Le zone sharding permet d'associer certains intervalles de clé de sharding à des « zones » logiques, elles-mêmes mappées à des shards spécifiques. Cela permet par exemple de placer les données d'une région géographique sur des shards situés physiquement dans cette région, ou de séparer des données « chaudes » et « froides » sur des nœuds différents, pour des raisons de conformité, de latence ou de coûts.

21. Comment MongoDB gère-t-il les requêtes multi-shards ?
Pour une requête dont le filtre inclut la clé de sharding, le mongos peut souvent cibler un seul shard ou un sous-ensemble précis de shards. Si la requête ne permet pas de déduire des shards spécifiques, le mongos envoie la requête à tous les shards (broadcast), puis agrège les résultats, ce qui peut être plus coûteux en termes de latence et de ressources.

22. Comment optimiser les performances de requêtes dans un environnement shardé ?
Il est important de concevoir les requêtes et les index de manière à inclure la clé de sharding dans les filtres, afin de favoriser les requêtes ciblées plutôt que les broadcasts. Il faut également surveiller la distribution des chunks, ajuster les indexes, éviter les scans globaux inutiles, et éventuellement recourir au zone sharding ou à l'ajout de shards supplémentaires pour absorber la charge.

23. Que se passe-t-il lorsqu'un shard devient indisponible ?
Si les shards sont des replica sets, la perte d'un nœud secondaire est transparente, et la perte d'un primaire entraîne une élection pour choisir un nouveau primaire, ce qui permet de continuer à servir les données avec une courte interruption. Si un shard complet devient indisponible, les données qu'il contient peuvent être temporairement inaccessibles ou en lecture seule, selon la configuration, et certaines requêtes échoueront tant que le shard n'est pas rétabli.

24. Comment migrer une collection existante vers un schéma shardé ?
Il faut d'abord s'assurer que la collection dispose d'un index correspondant à la future clé de sharding, puis activer le sharding sur la base et appeler sh.shardCollection sur cette collection. MongoDB va alors créer les chunks et commencer à les distribuer entre les shards, ce qui peut entraîner des opérations de migration en arrière-plan ; pour de très gros volumes, on planifie généralement cela en période de faible charge.

25. Quels outils ou métriques utiliser pour diagnostiquer les problèmes de sharding ?
On peut utiliser sh.status(), les collections de la base config, les logs des mongos et des shards, ainsi que les métriques de monitoring (latence, taux d'opérations, taille des chunks, erreurs) fournies par des outils comme MongoDB Atlas ou Percona Monitoring and Management. L'analyse des migrations de chunks, du temps de réponse des requêtes multi-shards, du taux de requêtes broadcast et de la charge par shard permet d'identifier les déséquilibres et les problèmes de conception de la clé de sharding.