Le partionnement sous MongoDB

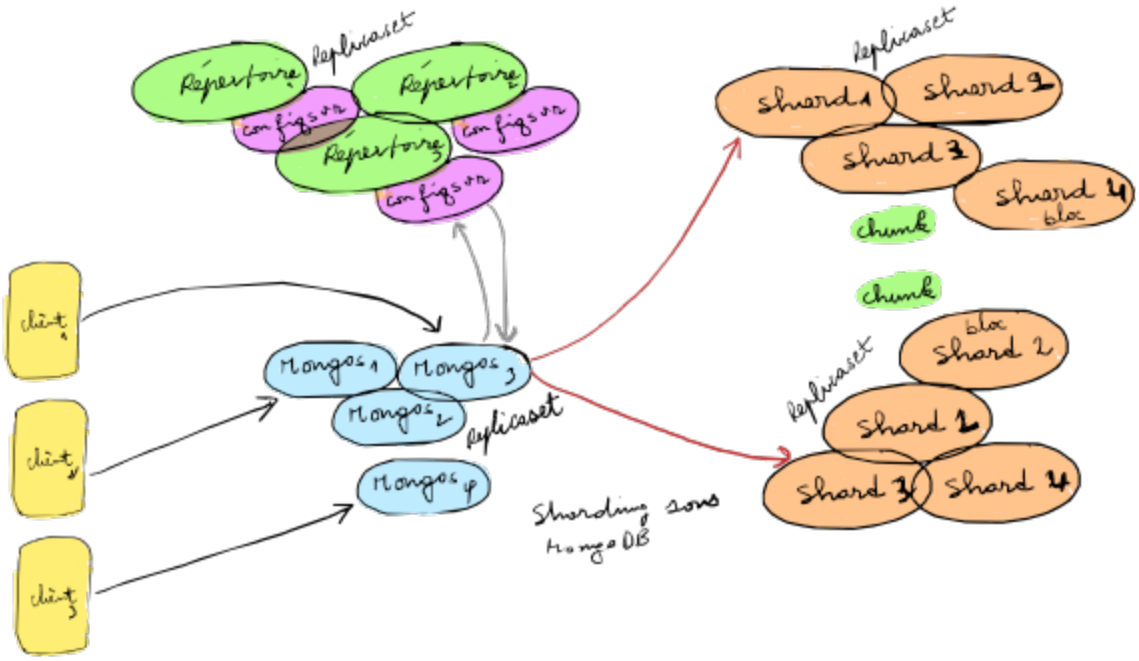
Dans ce TP, vous allez mettre en œuvre un système de sharding sous MongoDB.

Récap :
Le partitionnement (sharding) sert à découper une grande base de données en plusieurs morceaux plus petits, répartis sur plusieurs serveurs. Pour un débutant, tu peux l'imaginer comme une grande bibliothèque qu'on répartit dans plusieurs salles : au lieu de tout mettre dans une seule pièce qui devient pleine et lente à utiliser, on divise les livres par thème et on les met dans plusieurs salles. En MongoDB, ça permet de stocker beaucoup plus de données et de répondre à plus de requêtes en même temps, simplement en ajoutant de nouveaux serveurs (nouveaux shards) quand la charge augmente. Donc le partitionnement est surtout utile pour passer à l'échelle (scalabilité) et garder de bonnes performances quand les données et les utilisateurs se multiplient.

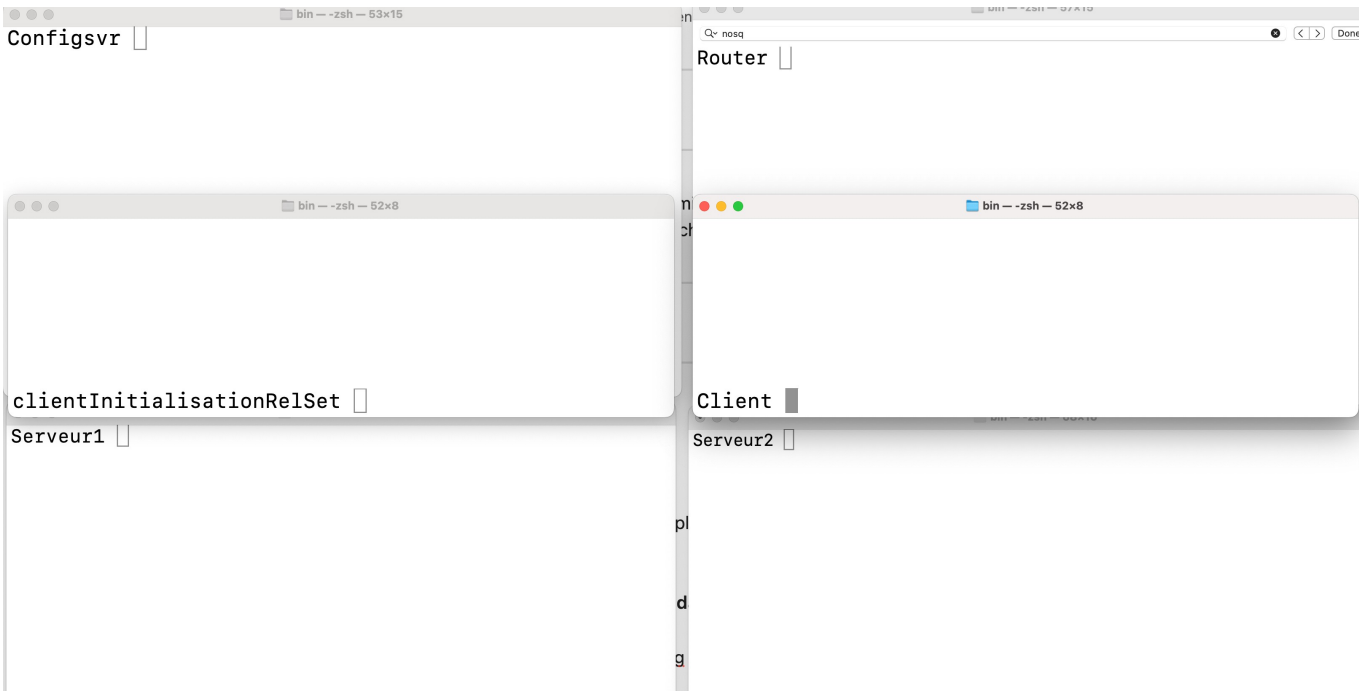
Je vais vous décrire pas à pas comment mettre en place un cluster MongoDB composé de trois éléments essentiels : **un serveur de configuration (config server)**, **un routeur (mongos)**, **et**

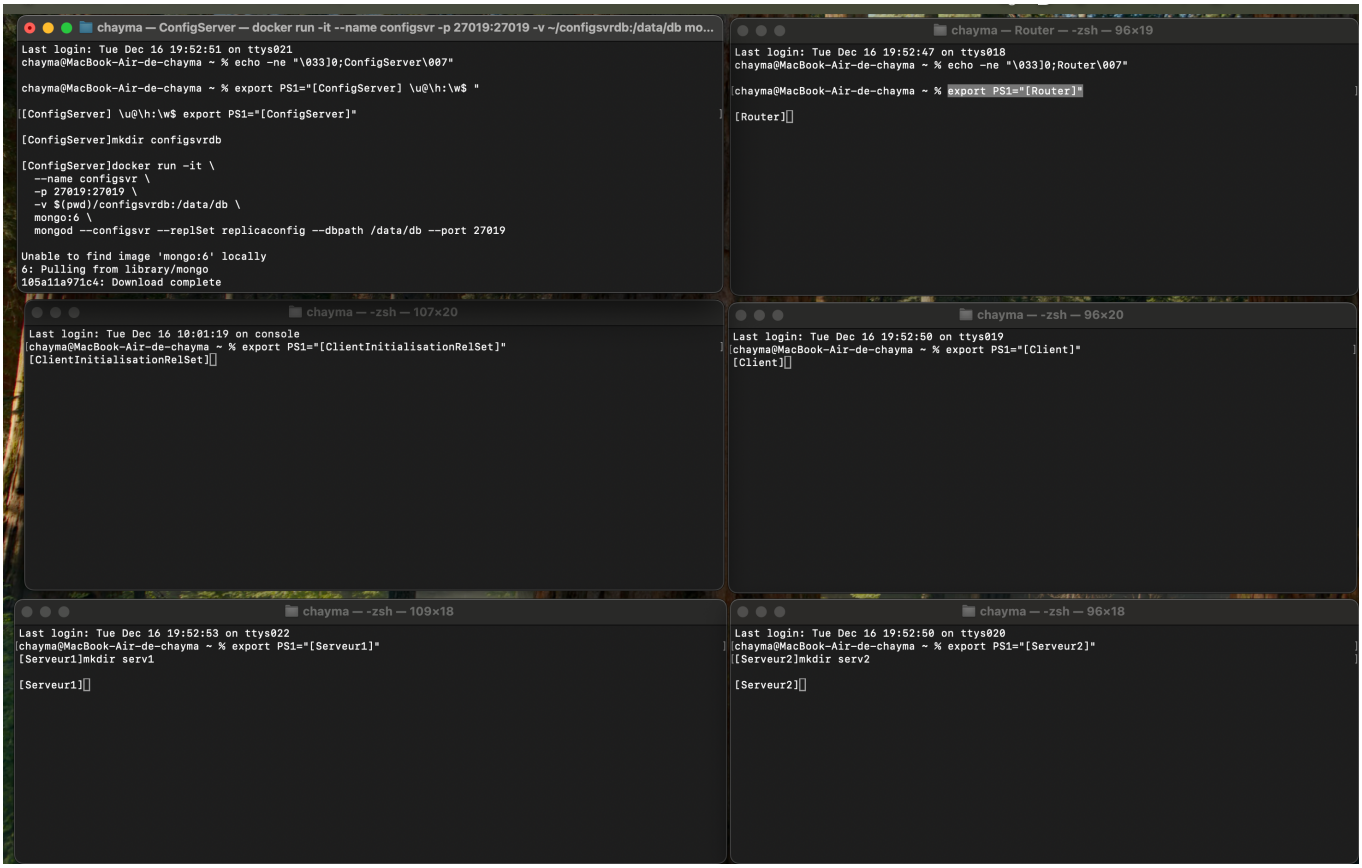
- **deux shards (serveur1 et serveur2)**.
- Je vous rappelle qu'un **replica set** constitue l'unité de base d'une architecture tolérante aux pannes et capable de passer à l'échelle. Dans ce TP, nous allons principalement nous concentrer sur le **partitionnement des données** (sharding). Toutefois, il 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.

```
[ClientInitialisationRelSet]docker exec -it configsvr mongosh --port 27019

$$Current Mongosh Log ID:       6941b0f8aea25217594f87fd
Connecting to:          mongodb://127.0.0.1:27019/?directConnection=true&serverSelectionTimeoutMS=2000&appN
ame=mongosh+2.5.8
Using MongoDB:          6.0.26
Using Mongosh:          2.5.8

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


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

------
   The server generated these startup warnings when booting
   2025-12-16T19:13:04.489+00:00: Access control is not enabled for the database. Read and write access to
```

INITIALISER LE REPLICA SET :

```
test> rs.initiate({
...   _id: "replicaconfig",
...   configsvr: true,
...   members: [
...     { _id: 0, host: "localhost:27019" }
...   ]
... })
[...
{ ok: 1, lastCommittedOpTime: Timestamp({ t: 1765912919, i: 1 }) }
replicaconfig [direct: other] test>
```

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
```

-
-
-
-

pour Serveur 1 :

```
chayma — docker run -it --name shard1 -p 20004:20004 -v ~/serv1:/data/db mongo:6 mongod --shardsv...
Last login: Tue Dec 16 19:52:53 on ttys022
chayma@MacBook-Air-de-chayma ~ % export PS1="[Serveur1]"
[Serveur1]mkdir serv1

[Serveur1]docker run -it \
  --name shard1 \
  -p 20004:20004 \
  -v $(pwd)/serv1:/data/db \
  mongo:6 \
  mongod --shardsvr --replSet replicashard1 --dbpath /data/db --port 20004

{"t":{"$date":"2025-12-16T19:28:46.435+00:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"main","msg":"In
itialized wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":1
7},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":17},"outgoing":{"minWireVersion":6,"maxWireV
ersion":17},"isInternalClient":true}}}
{"t":{"$date":"2025-12-16T19:28:46.437+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Au
tomatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2025-12-16T19:28:46.437+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Im
```

```
[ClientInitialisationRelSet]docker exec -it shard1 mongosh --port 20004

Current Mongosh Log ID: 6941b39e4f1e71cbab4f87fd
Connecting to:          mongodb://127.0.0.1:20004/?directConnection=true&serverSelectionTimeoutMS=2000&appN
ame=mongosh+2.5.8
Using MongoDB:          6.0.26
Using Mongosh:          2.5.8

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

```
test> rs.initiate({
...   _id: "replicashard1",
...   members: [
...     { _id: 0, host: "localhost:20004" }
...   ]
... })
[...
{ ok: 1 }
replicashard1 [direct: other] test>
```

Pour Serveur 2 :

```
chayma — docker run -it --name shard2 -p 20005:20005 -v ~/serv2:/data/db mongo:6 mon...
Last login: Tue Dec 16 19:52:50 on ttys020
chayma@MacBook-Air-de-chayma ~ % export PS1="[Serveur2]"
[Serveur2]mkdir serv2

[Serveur2]docker run -it \
  --name shard2 \
  -p 20005:20005 \
  -v $(pwd)/serv2:/data/db \
  mongo:6 \
  mongod --shardsvr --replSet replicashard2 --dbpath /data/db --port 20005

{"t":{"$date":"2025-12-16T19:34:41.516+00:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"-"
,"msg":"Initialized wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersio
n":0,"maxWireVersion":17},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":17},"out
going":{"minWireVersion":6,"maxWireVersion":17},"isInternalClient":true}}}
{"t":{"$date":"2025-12-16T19:34:41.519+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"ma
in","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtoco
ls 'none'"}
```

```
[ClientInitialisationRelSet]docker exec -it shard2 mongosh --port 20005

Current Mongosh Log ID: 6941b47a43c82dcc924f87fd
Connecting to:          mongodb://127.0.0.1:20005/?directConnection=true&serverSelectionTimeoutMS=2000&appN
ame=mongosh+2.5.8
Using MongoDB:          6.0.26
Using Mongosh:          2.5.8

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


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

```
test>

test> rs.initiate({
...   _id: "replicashard2",
...   members: [
...     { _id: 0, host: "localhost:20005" }
...   ]
... })
...
{ ok: 1 }
replicashard2 [direct: other] test>
```

--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 suivante

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

```
[direct: mongos] test> sh.addShard("replicashard1/shard1:20004")
{
  shardAdded: 'replicashard1',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1765921010, i: 6 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1765921010, i: 6 })
}
[direct: mongos] test>
```

```
sh.addShard("replicashard2/shard2:20005")
{
  shardAdded: 'replicashard2',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1765921158, i: 5 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1765921158, i: 5 })
}
[direct: mongos] test>
```

Les bases de données gérées par MongoDB ne sont pas partitionnées (*shardées*) par défaut. I 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.

```
[direct: mongos] test> ^[[200~sh.enableSharding("mabasefilms")
sh.shardCollection("mabasefilms.films", { "titre": 1 })^[[201~
sh.enableSharding("mabasefilms")
... sh.shardCollection("mabasefilms.films", { "titre": 1 })
{
  collectionsharded: 'mabasefilms.films',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1765921218, i: 32 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1765921218, i: 28 })
}
[direct: mongos] test>
```

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 = partitionnement horizontal des données : une collection est répartie sur plusieurs serveurs (shards).
   But : scaler en capacité de stockage et en débit (plus de données, plus de requêtes) en ajoutant des machines plutôt qu'en grossissant une seule.

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

   - **Réplication** : mêmes données copiées sur plusieurs nœuds → tolérance aux pannes, haute disponibilité, lecture répartie.
   - **Sharding** : les données sont différentes d'un shard à l'autre → scalabilité horizontale et distribution de la charge

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

   Shards **:** ensembles de données

   Config servers: stockent la métadonnée de sharding

mongos: point d'entrée pour les clients, route les requêtes vers les bons shards.

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

   - Conservent la configuration du cluster : bases/collections shardées, chunks, attributions des chunks aux shards, configuration du balancer.
   - Servent de source de vérité pour `mongos`.
   - Sans eux, le cluster ne sait plus où sont les données.

5. Quel est le rôle du **mongos router** ?
   C'est le proxy : les clients se connectent à mongos, pas directement aux shards.
   Il analyse la requête, regarde la métadonnée sur les config servers, décide quels shards contacter, agrège les résultats et les renvoie au client.

6. Comment MongoDB décide-t-il sur quel shard stocker un document ?
   D'après la valeur de la clé de sharding du document.
   En fonction du type de sharding :
   - Ranged sharding : intervalle de clés → shard donné.
   - Hashed sharding : hash→ plage de hash → shard.

   mongos se sert des chunks pour choisir le shard.

7. Qu'est-ce qu'une **clé de sharding** et pourquoi est-elle essentielle ?
   Champ d'un document utilisé pour distribuer les données entre les shards.
   Essentielle car elle conditionne :
   - la répartition de la charge,
   - la possibilité de cibler un shard dans les requêtes,
   - la scalabilité et les performances.

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

   - Haute cardinalité (beaucoup de valeurs différentes).
   - Bonne distribution des insertions/updates (éviter les hotspots).
   - Présente dans les requêtes fréquentes (pour éviter le scatter/gather).
   - Stable (éviter les mises à jour fréquentes de la shard key).
   - Pas monotone (éviter que toutes les nouvelles écritures arrivent sur le même shard).

9. Qu'est-ce qu'un **chunk** dans MongoDB ?
   Un intervalle de valeurs de shard key associé à un shard.
   Tous les documents dont la shard key est dans ce range appartiennent à ce chunk.

C’est l’unité de base utilisée par le balancer pour migrer les données.

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

- Quand un chunk devient trop gros, MongoDB le split en plusieurs chunks plus petits.
- Les nouvelles bornes de shard key divisent l’intervalle initial.
- Ces chunks pourront ensuite être répartis sur plusieurs shards par le balancer.

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

Processus en tâche de fond qui équilibre la distribution des chunks entre les shards.

S’assure qu’aucun shard n’a beaucoup plus de chunks que les autres.

Lance des migrations de chunks lorsque la distribution est déséquilibrée.

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

Quand la différence de nombre de chunks entre shards dépasse un seuil.

Il choisit un chunk sur un shard surchargé, le copie vers un shard moins chargé, met à jour les métadonnées sur les config servers, puis supprime l’ancienne copie.

Les migrations sont contrôlées (une à la fois par défaut) et peuvent être planifiées (fenêtre horaire).

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

Shard qui reçoit beaucoup plus de trafic ou d’insertions que les autres.

Causes typiques **:** mauvaise shard key, monotone, faible cardinalité.

Pour l’éviter :choisir une clé de sharding mieux répartie, utiliser un hashed sharding key, éventuellement pré-splitter les chunks.

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

Toutes les nouvelles écritures arrivent:

- dans le même chunk,
- donc sur le même shard → hot shard.

Le chunk grossit beaucoup → jumbo chunk difficile à déplacer.
Déséquilibre de stockage et de charge.

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

- Sur la base :

```
sh.enableSharding("mabase")
```

- Sur la collection :

```
sh.shardCollection("mabase.maCollection", { shardKeyField: 1 })
```

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

- Depuis `mongos` :

```
sh.addShard("nomReplicaSet/host:port")
```

- MongoDB l'ajoute à la liste des shards et le balancer commencera à migrer des chunks vers ce nouveau shard pour équilibrer.

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

Commandes usuelles dans mongos :

- sh.status() ou db.printShardingStatus() : vue globale (shards, chunks, collections shardées).
- sh.getBalancerState() ou sh.isBalancerRunning() : état du balancer.
- Côté shards : rs.status() pour l'état des replica sets.

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

- Quand on veut uniformiser les écritures et éviter les hotspots.
- Quand les requêtes sont surtout des recherches par égalité sur la shard key.
- Quand on n'a pas besoin de faire beaucoup de requêtes par intervalle sur la shard

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

- Quand on a besoin de requêtes par intervalle
- Quand on veut que des données proches (même client, même région, même période) soient physiquement regroupées**.**
- Utile pour l'archivage, le tri, ou certaines requêtes analytiques.

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

On associe des plages de shard key à des zones ,puis on lie ces zones à des shards.
Intérêt :

- Localiser les données
- Répondre à des contraintes légales
- Optimiser la latence pour des régions géographiques.

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

mongos analyse la requête :

- Si elle contient la shard key ou un filtre ciblant un range → requête
- Sinon → scatter/gather : mongos envoie la requête à tous les shards, agrège les réponses, trie / limite si besoin, puis renvoie au client.

Le but est de réduire au maximum le scatter/gather en choisissant bien la shard key.

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

- Choisir une bonne shard key alignée avec les requêtes fréquentes.
- Faire des requêtes ciblées sur la shard key
- Créer les index appropriés
- Limiter les champs retournés ,utiliser `limit`, `skip` intelligemment.
- Surveiller la répartition des chunks et l'activité du balancer.

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

Si le shard est un **replica set** :

- élection d'un nouveau PRIMARY si seuls certains nœuds tombent.

Si le shard complet est down :

- les données contenues dans ses chunks sont inaccessibles,
- certaines requêtes peuvent échouer ou retourner des résultats incomplets,
- le balancer ne migre plus de chunks vers/de ce shard.

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

- S'assurer qu'il y a un index sur le champ choisi comme shard key.
- Activer le sharding sur la base :

```
sh.enableSharding("mabase")
```

- Sharder la collection existante :

```
sh.shardCollection("mabase.maCollection", { shardKeyField: 1 })
```

- MongoDB va créer les chunks et migrer progressivement les données en fonction de la shard key.

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

Commandes mongo :

- sh.status(), db.printShardingStatus()

- db.serverStatus(), db.currentOp()
- rs.status() sur chaque shard.

Logs des mongod / mongos

Profiler MongoDB (db.setProfilingLevel) pour voir les requêtes lentes.

Outils de monitoring (Atlas, Cloud Manager, Prometheus + Grafana, etc.) :

- latence des requêtes,
- charge par shard