Réplication et reprise sur panne (Partie 1)

Dans une base de données NoSQL telle que **MongoDB**, l'un des enjeux majeurs est d'assurer une **haute disponibilité** et une **tolérance aux pannes**. Pour répondre à ces besoins, MongoDB utilise un mécanisme appelé **Replica Set**, basé sur une architecture **Maître–Esclave** (plus exactement *Primary – Secondary*).

Un **Replica Set** est un groupe de serveurs MongoDB contenant les mêmes données.
Il comprend :

- **Un serveur Primary** → Par défaut, le serveur primaire reçoit toutes les lectures et écritures pour assurer la réconciliation des données et garantir une **cohérence forte (accéder toujours à la version finale)**

- **Un ou plusieurs serveurs Secondary** → répliquent automatiquement les données du Primary.

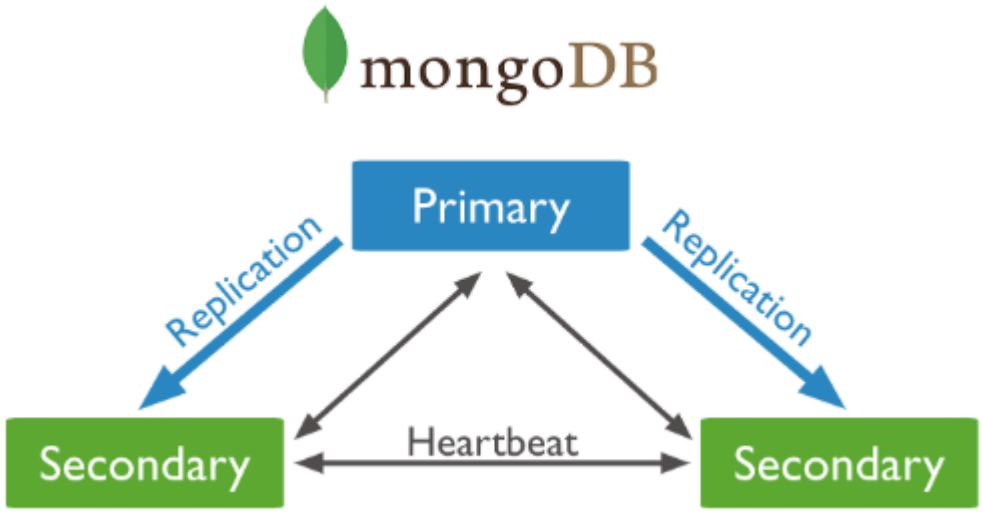
  Remarque : Pour passer à l'échelle et répartir la charge, on peut configurer MongoDB pour autoriser les lectures sur les réplicas secondaires. Mais comme la réplication n'est pas instantanée, ça peut présenter un risque (accéder à une version obsolète)

  La réplication de Mongodb est asynchrone

  La réplication n'est pas la stratégie de MongoDB pour monter en charge, il utilise le partitionnement (voir plus tard), en revanche elle assure l**a tolérance en panne**

- **Optionnel : un Arbitre** → participe à l'élection d'un nouveau Primary, mais ne stocke pas de données.

Ce mécanisme garantit que le système continue de fonctionner même si une machine tombe en panne.



Fonctionnement de la réplication

Le Primary reçoit :

- toutes les écritures,
- la majorité des lectures (sauf configuration spéciale).

✔ Chaque Secondary :

- copie en temps réel le journal d'opérations du Primary,
- applique les changements localement.

Ainsi, toutes les données sont présentes sur plusieurs machines.

Que se passe-t-il si le Primary tombe en panne ?

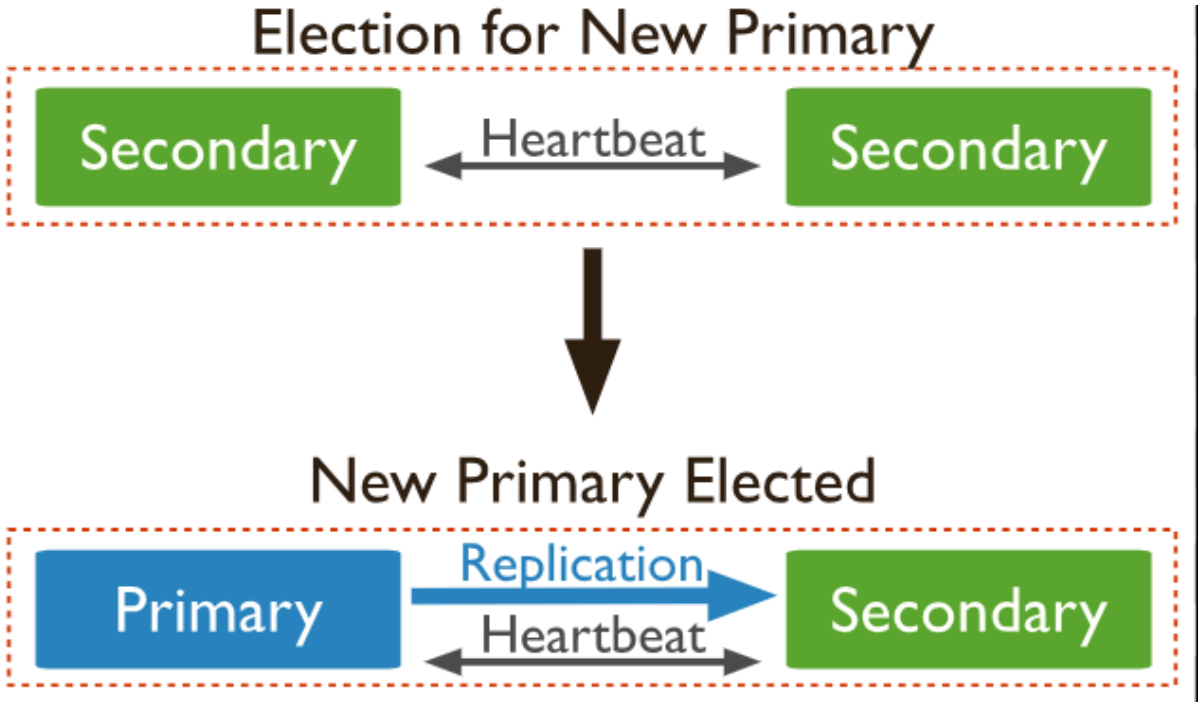
MongoDB lance automatiquement une **élection** pour élire un nouveau Primary.

Règle fondamentale : la classe majoritaire (majority)

Un serveur ne peut être élu Primary que si **la majorité des nœuds** (machines) vote pour lui.

Pourquoi ?

Cela empêche qu'un sous-réseau isolé crée un second Primary → évite les incohérences.



Cas pratique :

Cas 1 : Deux sous-réseaux avec un nombre de machines différent

Sous-réseau 1 : **5 machines**
Sous-réseau 2 : **4 machines**

La majorité = 5 nœuds → située dans le sous-réseau 1
➡️ Le Primary **doit** être dans ce sous-réseau

Si le réseau se coupe en deux :

- Sous-réseau 1 → 5 nœuds → **majorité**, peut élire un nouveau Primary
- Sous-réseau 2 → 4 nœuds → **pas de majorité**, reste en Secondary

➡️ Résultat : le système continue de fonctionner sans corruption des données.

Cas 2 : Deux sous-réseaux avec le même nombre de machines

Sous-réseau 1 : **4 machines**
Sous-réseau 2 : **4 machines**

Problème : aucune partie ne détient la majorité (4 + 4 → majority = 5)

Solution : Ajouter un arbitre (arbitre)

Cet arbitre vote pour un des deux sous-réseaux → donne la majorité.

L'arbitre ne stocke PAS de données, il ne sert **qu'aux élections**.

Manipulations :

Étape 1 – Créer les dossiers de données : disque1, disque2, disque3

Étape 2 – Lancer les 3 serveurs MongoDB

```
ing@ing:~/Bureau/NOSQL_TPS/TP_réplication_MongoDB$ mongod --replSet monreplicase
t --port 27018 --dbpath ~/Bureau/NOSQL_TPS/TP_réplication_MongoDB/disque1
{"t":{"$date":"2025-12-04T14:48:38.516+01:00"},"s":"I",  "c":"NETWORK",  "id":49
15701, "ctx":"main","msg":"Initialized wire specification","attr":{"spec":{"inco
mingExternalClient":{"minWireVersion":0,"maxWireVersion":21},"incomingInternalCl
ient":{"minWireVersion":0,"maxWireVersion":21},"outgoing":{"minWireVersion":6,"m
axWireVersion":21},"isInternalClient":true}}}
{"t":{"$date":"2025-12-04T14:48:38.516+01:00"},"s":"I",  "c":"CONTROL",  "id":23
285,   "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS
1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2025-12-04T14:48:38.517+01:00"},"s":"I",  "c":"NETWORK",  "id":46
48601, "ctx":"main","msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is
 required, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
{"t":{"$date":"2025-12-04T14:48:38.519+01:00"},"s":"I",  "c":"REPL",     "id":51
23008, "ctx":"main","msg":"Successfully registered PrimaryOnlyService","attr":{"
service":"TenantMigrationDonorService","namespace":"config.tenantMigrationDonors
"}}
{"t":{"$date":"2025-12-04T14:48:38.519+01:00"},"s":"I",  "c":"REPL",     "id":51
23008, "ctx":"main","msg":"Successfully registered PrimaryOnlyService","attr":{"
service":"TenantMigrationRecipientService","namespace":"config.tenantMigrationRe
cipients"}}
```

```
ing@ing:~/Bureau/NOSQL_TPS/TP_réplication_MongoDB$ mongod --replSet monreplicaset
 --port 27019 --dbpath ~/Bureau/NOSQL_TPS/TP_réplication_MongoDB/disque2
{"t":{"$date":"2025-12-04T14:50:25.630+01:00"},"s":"I",  "c":"CONTROL",  "id":232
85,   "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.
0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2025-12-04T14:50:25.631+01:00"},"s":"I",  "c":"NETWORK",  "id":491
5701, "ctx":"main","msg":"Initialized wire specification","attr":{"spec":{"incomi
ngExternalClient":{"minWireVersion":0,"maxWireVersion":21},"incomingInternalClien
t":{"minWireVersion":0,"maxWireVersion":21},"outgoing":{"minWireVersion":6,"maxWi
reVersion":21},"isInternalClient":true}}}
{"t":{"$date":"2025-12-04T14:50:25.632+01:00"},"s":"I",  "c":"NETWORK",  "id":464
8601, "ctx":"main","msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is r
equired, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
{"t":{"$date":"2025-12-04T14:50:25.632+01:00"},"s":"I",  "c":"REPL",     "id":512
3008, "ctx":"main","msg":"Successfully registered PrimaryOnlyService","attr":{"se
rvice":"TenantMigrationDonorService","namespace":"config.tenantMigrationDonors"}}
{"t":{"$date":"2025-12-04T14:50:25.632+01:00"},"s":"I",  "c":"REPL",     "id":512
3008, "ctx":"main","msg":"Successfully registered PrimaryOnlyService","attr":{"se
rvice":"TenantMigrationRecipientService","namespace":"config.tenantMigrationRecip
ients"}}
```

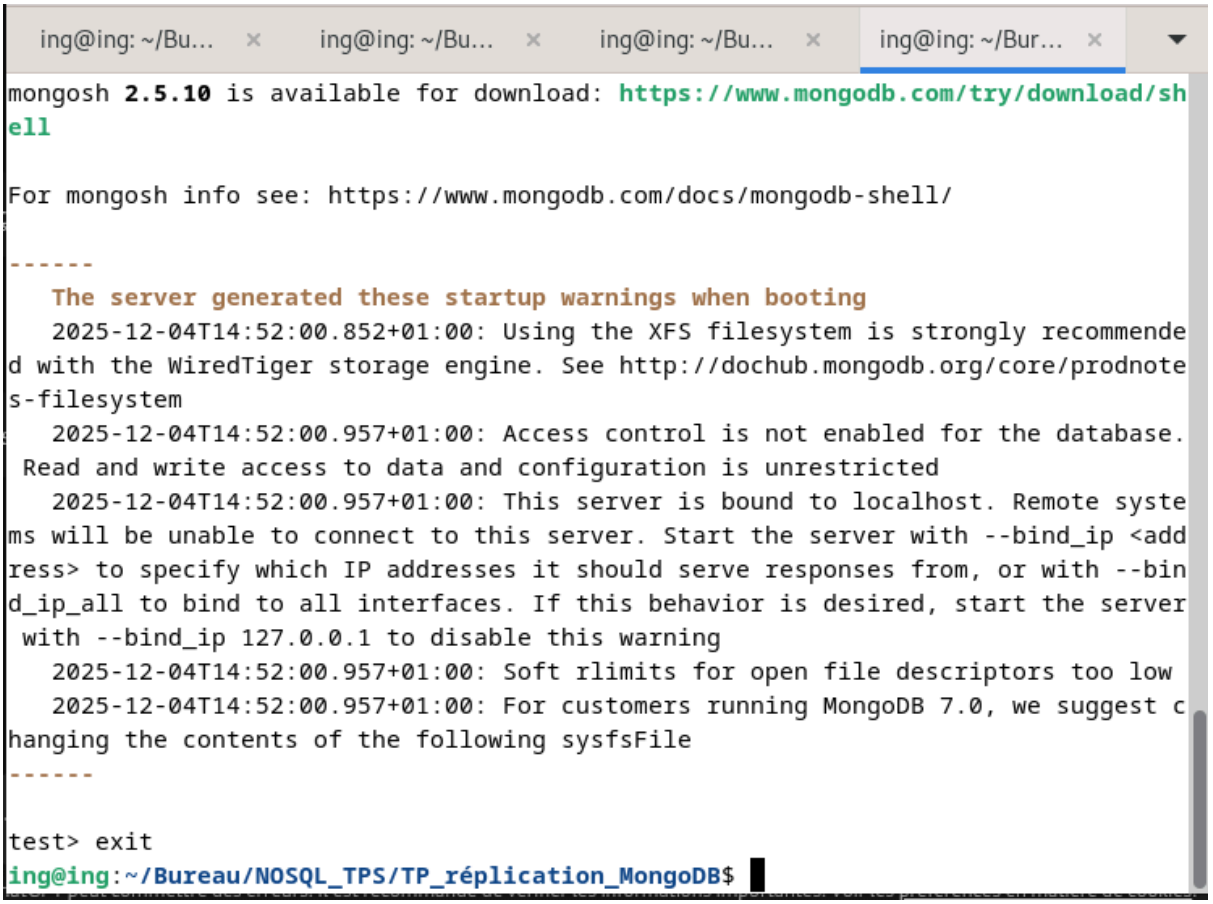
```
ing@ing:~/Bureau/NOSQL_TPS/TP_réplication_MongoDB$ mongod --replSet monreplicaset
 --port 27020 --dbpath ~/Bureau/NOSQL_TPS/TP_réplication_MongoDB/disque3
{"t":{"$date":"2025-12-04T14:52:00.850+01:00"},"s":"I",  "c":"NETWORK",  "id":491
5701, "ctx":"main","msg":"Initialized wire specification","attr":{"spec":{"incomi
ngExternalClient":{"minWireVersion":0,"maxWireVersion":21},"incomingInternalClien
t":{"minWireVersion":0,"maxWireVersion":21},"outgoing":{"minWireVersion":6,"maxWi
reVersion":21},"isInternalClient":true}}}
{"t":{"$date":"2025-12-04T14:52:00.850+01:00"},"s":"I",  "c":"CONTROL",  "id":232
85,   "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.
0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2025-12-04T14:52:00.851+01:00"},"s":"I",  "c":"NETWORK",  "id":464
8601, "ctx":"main","msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is r
equired, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
{"t":{"$date":"2025-12-04T14:52:00.852+01:00"},"s":"I",  "c":"REPL",     "id":512
3008, "ctx":"main","msg":"Successfully registered PrimaryOnlyService","attr":{"se
rvice":"TenantMigrationDonorService","namespace":"config.tenantMigrationDonors"}}
{"t":{"$date":"2025-12-04T14:52:00.852+01:00"},"s":"I",  "c":"REPL",     "id":512
3008, "ctx":"main","msg":"Successfully registered PrimaryOnlyService","attr":{"se
rvice":"TenantMigrationRecipientService","namespace":"config.tenantMigrationRecip
ients"}}
{"t":{"$date":"2025-12-04T14:52:00.852+01:00"},"s":"I",  "c":"CONTROL",  "id":594
5603, "ctx":"main","msg":"Multi threading initialized"}
{"t":{"$date":"2025-12-04T14:52:00.852+01:00"},"s":"I",  "c":"TENANT_M", "id":709
```

- .

Étape 3 – Se connecter à un nœud et initialiser le replica set

Ouvre **un 4e terminal** :

Se connecter au serveur primaire :



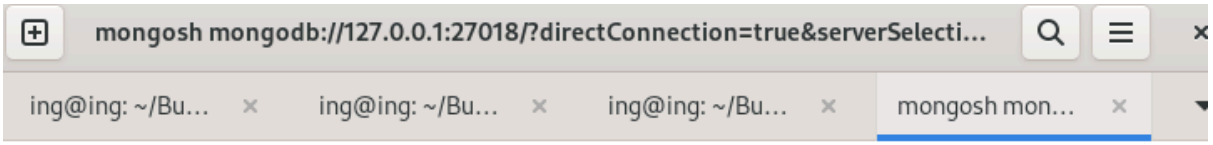
Initialiser le replica set :

```
test> rs.initiate()
{
  info2: 'no configuration specified. Using a default configuration for the se
  me: 'localhost:27018',
  ok: 1
}
monreplicaset [direct: other] test> exit
```

Ajouter les serveurs 2 et 3 au replicaset :

Afficher la configuration actuelle du replicaset :

```
monreplicaset [direct: primary] test> rs.config()
{
  _id: 'monreplicaset',
  version: 5,
  term: 1,
  members: [
    {
      _id: 0,
      host: 'localhost:27018',
      arbiterOnly: false,
      buildIndexes: true,
      hidden: false,
      priority: 1,
      tags: {},
      secondaryDelaySecs: Long('0'),
      votes: 1
    },
    {
      _id: 1,
      host: 'localhost:27019',
      arbiterOnly: false,
      buildIndexes: true,
      hidden: false,
```




```
monreplicaset [direct: secondary] test> rs.isMaster()
{
  topologyVersion: {
    processId: ObjectId('6931979a7307a48c80ba42e7'),
    counter: Long('12')
  },
  hosts: [ 'localhost:27018', 'localhost:27019', 'localhost:27020' ],
  setName: 'monreplicaset',
  setVersion: 5,
  ismaster: false,
  secondary: true,
  primary: 'localhost:27020',
  me: 'localhost:27018',
  lastWrite: {
    opTime: { ts: Timestamp({ t: 1764874866, i: 1 }), t: Long('2') },
    lastWriteDate: ISODate('2025-12-04T19:01:06.000Z'),
    majorityOpTime: { ts: Timestamp({ t: 1764874866, i: 1 }), t: Long('2') },
    majorityWriteDate: ISODate('2025-12-04T19:01:06.000Z')
  },
  maxBsonObjectSize: 16777216,
  maxMessageSizeBytes: 48000000,
  maxWriteBatchSize: 100000,
  localTime: ISODate('2025-12-04T19:01:14.381Z'),
```