

# I. La première partie : Mise en place d'un Replica Set MongoDB

## 1. Objectif

L'objectif de cette partie est de **comprendre et reproduire le fonctionnement de la réPLICATION dans MongoDB**.

La réPLICATION permet de créer plusieurs copies d'une base de données pour assurer **la haute disponibilité et la tolérance aux pannes**.

Dans cet exemple, nous avons mis en place un **système maître-esclave (primary-secondary)** avec **trois serveurs locaux**.

## 2. Création des dossiers de stockage

Chaque serveur MongoDB a besoin d'un dossier dédié pour stocker ses données :

- **disque1** : pour le serveur 1
- **disque2** : pour le serveur 2
- **disque3** : pour le serveur 3

Cela permet de simuler trois serveurs indépendants sur la même machine.

## 3. Démarrage des instances MongoDB

Nous lançons **trois serveurs MongoDB**, chacun sur un port différent, avec le même nom de replica set (monreplicaset) :

```
# Serveur 1 (primary potentiel)
./mongod --replSet monreplicaset --port 27018 --dbpath disque1
```

```
# Serveur 2
./mongod --replSet monreplicaset --port 27019 --dbpath disque2
```

```
# Serveur 3
./mongod --replSet monreplicaset --port 27020 --dbpath disque3
```

```

hasnaelgarani — mongod --replSet monreplicaset --port 27018 --dbpath...
hasnaelgarani@MacBook-Air-de-hasna ~ % mongod --replSet monreplicaset --port 27018 --dbpath disque1
{"t": {"$date": "2025-12-04T14:57:58.975+01:00"}, "s": "I", "c": "NETWORK", "id": 4915701, "ctx": "thread1", "msg": "Initialized wire specification", "attr": {"spec": {"iincomingExternalClient": {"minWireVersion": 0, "maxWireVersion": 21}, "incomingInternalClient": {"minWireVersion": 0, "maxWireVersion": 21}, "outgoing": {"minWireVersion": 6, "maxWireVersion": 23}, "isInternalClient": true}}}, {"t": {"$date": "2025-12-04T14:57:58.976+01:00"}, "s": "I", "c": "CONTROL", "id": 23285, "ctx": "thread1", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}, {"t": {"$date": "2025-12-04T14:57:58.977+01:00"}, "s": "I", "c": "NETWORK", "id": 4648602, "ctx": "thread1", "msg": "Implicit TCP FastOpen in use."}, {"t": {"$date": "2025-12-04T14:57:58.977+01:00"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "thread1", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationDonorService", "namespace": "config.tenantMigrationDonors"}}, {"t": {"$date": "2025-12-04T14:57:58.977+01:00"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "thread1", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationRecipientService", "namespace": "config.tenantMigrationRecipients"}}, {"t": {"$date": "2025-12-04T14:57:58.977+01:00"}, "s": "I", "c": "CONTROL", "id": 5945603, "ctx": "thread1", "msg": "Multi threading initialized"}, {"t": {"$date": "2025-12-04T14:57:58.977+01:00"}, "s": "I", "c": "TENANT_M", "id": 7091600, "ctx": "thread1", "msg": "Starting TenantMigrationAccessBlockerRegistry"}
```

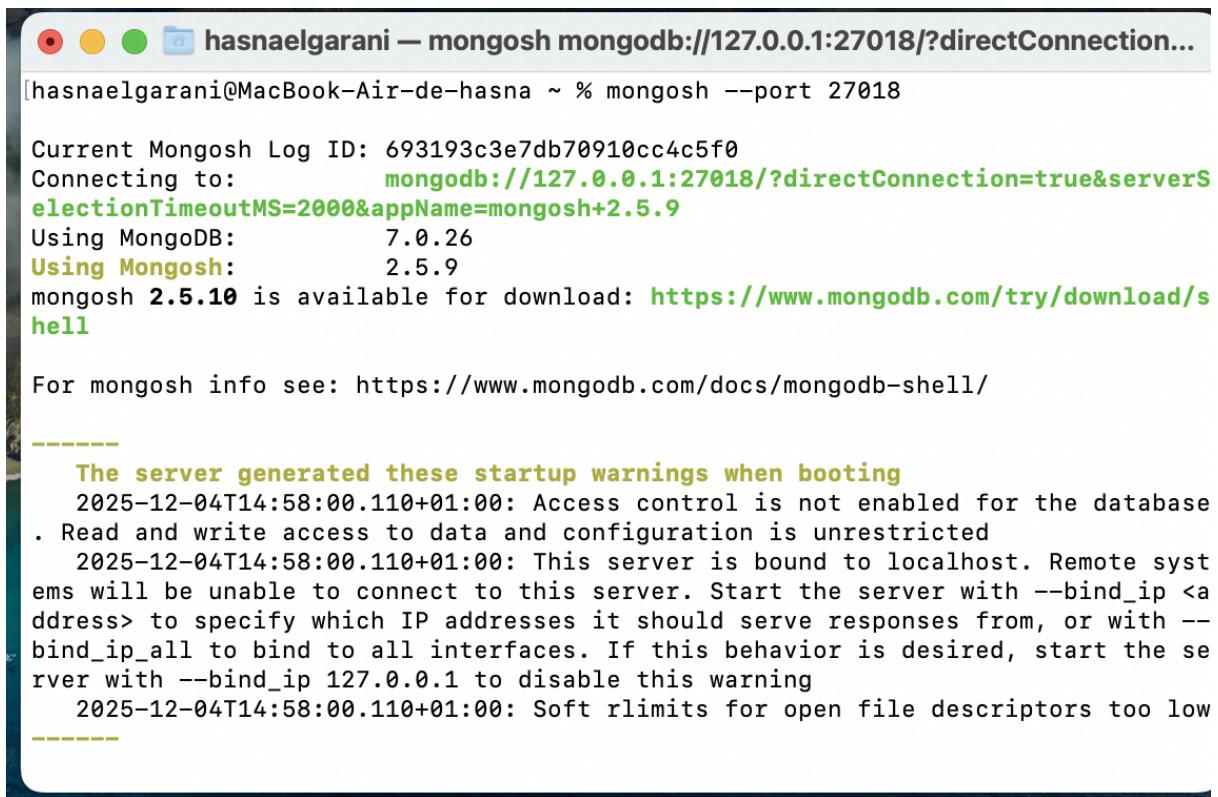
```

hasnaelgarani — mongod --replSet monreplicaset --port 27019 --dbpath...
hasnaelgarani@MacBook-Air-de-hasna ~ % mongod --replSet monreplicaset --port 27019 --dbpath disque2
Last login: Thu Dec 4 14:44:32 on ttys015
{"t": {"$date": "2025-12-04T14:58:19.339+01:00"}, "s": "I", "c": "NETWORK", "id": 4915701, "ctx": "thread1", "msg": "Initialized wire specification", "attr": {"spec": {"iincomingExternalClient": {"minWireVersion": 0, "maxWireVersion": 21}, "incomingInternalClient": {"minWireVersion": 0, "maxWireVersion": 21}, "outgoing": {"minWireVersion": 6, "maxWireVersion": 21}, "isInternalClient": true}}}, {"t": {"$date": "2025-12-04T14:58:19.340+01:00"}, "s": "I", "c": "CONTROL", "id": 23285, "ctx": "thread1", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}, {"t": {"$date": "2025-12-04T14:58:19.342+01:00"}, "s": "I", "c": "NETWORK", "id": 4648602, "ctx": "thread1", "msg": "Implicit TCP FastOpen in use."}, {"t": {"$date": "2025-12-04T14:58:19.342+01:00"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "thread1", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationDonorService", "namespace": "config.tenantMigrationDonors"}}, {"t": {"$date": "2025-12-04T14:58:19.342+01:00"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "thread1", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationRecipientService", "namespace": "config.tenantMigrationRecipients"}}, {"t": {"$date": "2025-12-04T14:58:19.342+01:00"}, "s": "I", "c": "CONTROL", "id": 5945603, "ctx": "thread1", "msg": "Multi threading initialized"}}
```

- L'option `--replSet` indique que chaque serveur fait partie du **même ensemble de réPLICATION**.
- Chaque serveur écoute sur un port différent pour permettre la communication entre eux.

## 4. Connexion au client MongoDB

Nous utilisons **mongosh** pour nous connecter à l'un des serveurs et administrer le replica set :



```
[hasnaelgarani@MacBook-Air-de-hasna ~ % mongosh --port 27018

Current Mongosh Log ID: 693193c3e7db70910cc4c5f0
Connecting to:      mongodb://127.0.0.1:27018/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.9
Using MongoDB:      7.0.26
Using Mongosh:      2.5.9
mongosh 2.5.10 is available for download: https://www.mongodb.com/try/download/shell

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

-----
The server generated these startup warnings when booting
2025-12-04T14:58:00.110+01:00: Access control is not enabled for the database
. Read and write access to data and configuration is unrestricted
2025-12-04T14:58:00.110+01:00: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <a address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
2025-12-04T14:58:00.110+01:00: Soft rlimits for open file descriptors too low
-----
```

- Le port 27018 correspond au serveur que nous voulons initialiser comme **PRIMARY**.
- Cette connexion permet de gérer les membres du replica set.

## 5. Initialisation du Replica Set

Dans le shell MongoDB, nous lançons :



```
hasnaelgarani — mongosh mongodb://127.0.0.1:27018/?directConnection...
test> rs.initiate()
[...
{
  info2: 'no configuration specified. Using a default configuration for the set'
,
  me: 'localhost:27018',
  ok: 1
}
[monreplicaset [direct: other] test> rs.add("localhost:27019")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1764859756, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAA='),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1764859756, i: 1 })
}
[monreplicaset [direct: primary] test> rs.add("localhost:27020")
{
  ok: 1,
  '$clusterTime': {
```

- MongoDB crée automatiquement une configuration par défaut pour le replica set.
- Le serveur sur le port 27018 devient **PRIMARY**.
- Les autres serveurs seront ajoutés comme **SECONDARY** par la suite.

#### Résultat observé :

- ok: 1 indique que l'initialisation a réussi.
- me: "localhost:27018" signifie que ce serveur est actuellement le PRIMARY.

## 6. Ajout des autres membres

Toujours dans le shell MongoDB, nous ajoutons les autres serveurs :

```
[monreplicaset [direct: other] test> rs.add("localhost:27019")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1764859756, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAA='),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1764859756, i: 1 })
}
[monreplicaset [direct: primary] test> rs.add("localhost:27020")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1764859763, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAA='),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1764859763, i: 1 })
}
```

- Ces serveurs deviennent des **SECONDARY** et vont répliquer automatiquement les données du PRIMARY.
- Chaque SECONDARY se synchronise avec le PRIMARY pour obtenir une copie à jour des données.

## 7. Vérification de l'état du Replica Set

Pour s'assurer que tout fonctionne correctement, nous utilisons :

```
hasnaelgarani — mongosh mongodb://127.0.0.1:27018/?directConnection...
monreplicaset [direct: primary] test> rs.status()
[...
{
  set: 'monreplicaset',
  date: ISODate('2025-12-04T14:49:38.581Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1764859768, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2025-12-04T14:49:28.013Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1764859768, i: 1 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1764859768, i: 1 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1764859768, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2025-12-04T14:49:28.013Z'),
    lastDurableWallTime: ISODate('2025-12-04T14:49:28.013Z')
```

```

members: [
  {
    _id: 0,
    name: 'localhost:27018',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 2120,
    optime: { ts: Timestamp({ t: 1764859768, i: 1 }), t: Long('1') },
    optimeDate: ISODate('2025-12-04T14:49:28.000Z'),
    lastAppliedWallTime: ISODate('2025-12-04T14:49:28.013Z'),
    lastDurableWallTime: ISODate('2025-12-04T14:49:28.013Z'),
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    electionTime: Timestamp({ t: 1764857761, i: 2 }),
    electionDate: ISODate('2025-12-04T14:16:01.000Z'),
    configVersion: 5,
    configTerm: 1,
    self: true,
    lastHeartbeatMessage: ''
  },
  {
    _id: 1,
    name: 'localhost:27019',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 22,
    optime: { ts: Timestamp({ t: 1764859768, i: 1 }), t: Long('1') },
    optimeDurable: { ts: Timestamp({ t: 1764859768, i: 1 }), t: Long('1') },
    optimeDate: ISODate('2025-12-04T14:49:28.000Z'),
    optimeDurableDate: ISODate('2025-12-04T14:49:28.000Z'),
    lastAppliedWallTime: ISODate('2025-12-04T14:49:28.013Z'),
    lastDurableWallTime: ISODate('2025-12-04T14:49:28.013Z'),
    lastHeartbeat: ISODate('2025-12-04T14:49:38.145Z'),
    lastHeartbeatRecv: ISODate('2025-12-04T14:49:38.145Z'),
    pingMs: Long('0'),
    lastHeartbeatMessage: '',
    syncSourceHost: 'localhost:27018',
    syncSourceId: 0,
    infoMessage: '',
    configVersion: 5,
    configTerm: 1
  },
  {
    _id: 2,
    name: 'localhost:27020',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 14,
    ...
  }
]

```

### Résultat observé :

- **PRIMARY** : localhost:27018
- **SECONDARY** : localhost:27019 et localhost:27020
- Tous les membres sont **sains** (health: 1) et synchronisés.
- Le champ optime indique l'état de réPLICATION des opérations

## 8. Inspection de la configuration : rs.config()

La commande :

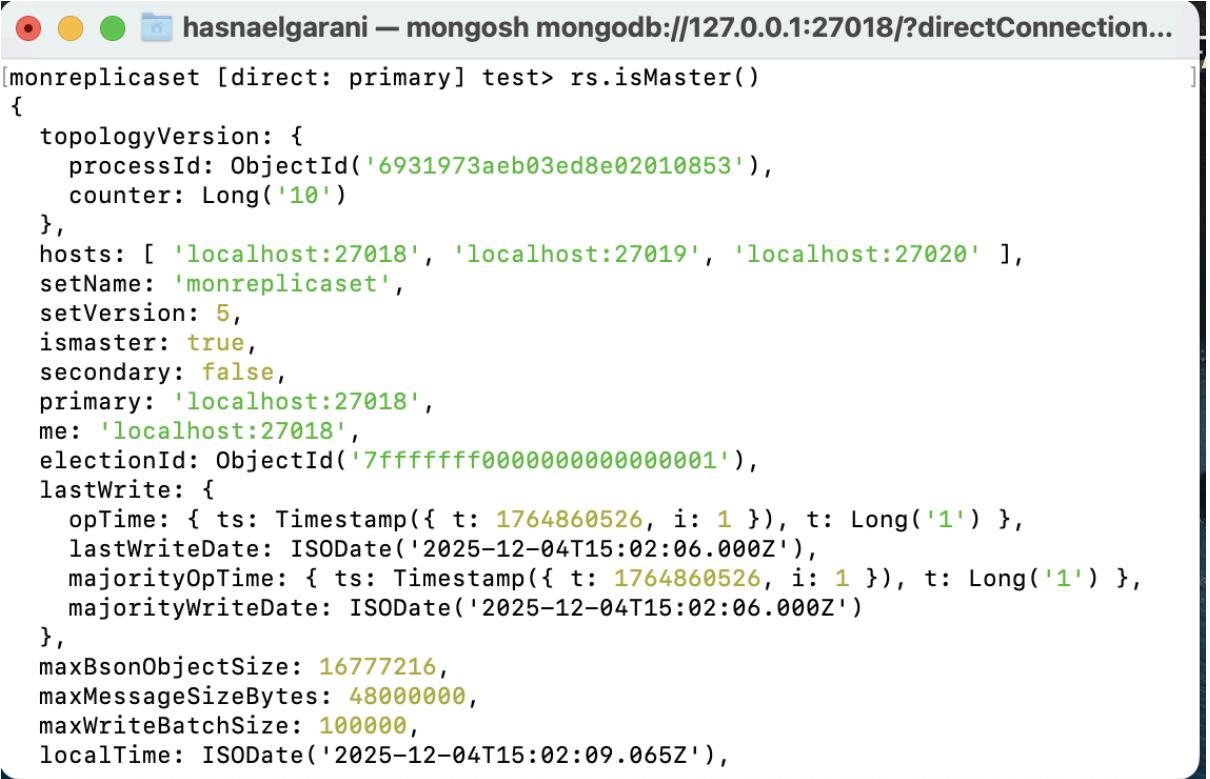


```
[monreplicaset direct: primary] test> rs.config()
{
  _id: 'monreplicaset',
  version: 5,
  term: 1,
  members: [
    {
      _id: 0,
      host: 'localhost:27018',
      arbiterOnly: false,
      buildIndexes: true,
      hidden: false,
      priority: 1,
      tags: {},
      secondaryDelaySecs: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

Cette commande montre que notre replica set est correctement configuré avec trois membres égaux, chacun pouvant devenir PRIMARY si nécessaire.

## 9. Vérification du rôle et de l'état : rs.isMaster()

La commande :



```

hasnaelgarani — mongosh mongodb://127.0.0.1:27018/?directConnection...
[monreplicaset [direct: primary] test> rs.isMaster()
{
  topologyVersion: {
    processId: ObjectId('6931973aeb03ed8e02010853'),
    counter: Long('10')
  },
  hosts: [ 'localhost:27018', 'localhost:27019', 'localhost:27020' ],
  setName: 'monreplicaset',
  setVersion: 5,
  ismaster: true,
  secondary: false,
  primary: 'localhost:27018',
  me: 'localhost:27018',
  electionId: ObjectId('7fffffff0000000000000001'),
  lastWrite: {
    opTime: { ts: Timestamp({ t: 1764860526, i: 1 }), t: Long('1') },
    lastWriteDate: ISODate('2025-12-04T15:02:06.000Z'),
    majorityOpTime: { ts: Timestamp({ t: 1764860526, i: 1 }), t: Long('1') },
    majorityWriteDate: ISODate('2025-12-04T15:02:06.000Z')
  },
  maxBsonObjectSize: 16777216,
  maxMessageSizeBytes: 48000000,
  maxWriteBatchSize: 100000,
  localTime: ISODate('2025-12-04T15:02:09.065Z'),
}

```

Elle permet de **savoir quel serveur est PRIMARY et quels sont les SECONDARY**.

## II. La deuxième partie : RéPLICATION et tolérance aux pannes avec MongoDB

### Partie 1 : Compréhension de base

- Replica Set** : Groupe de serveurs MongoDB qui maintiennent la même donnée pour assurer une haute disponibilité.
- Primary** : Reçoit toutes les écritures et sert de source de vérité.
- Secondaries** : Répliquent les données du Primary et servent de relais en cas de panne.
- Pas d'écritures sur Secondary** : Pour éviter les conflits et garantir un modèle cohérent.
- Cohérence forte** : Une lecture garantit les données les plus récentes du Primary.
- readPreference "primary"** : lecture cohérente. **"secondary"** : lecture possible mais potentiellement en retard.
- Lire sur un Secondary** : Pour distribuer la charge ou pour des analyses non critiques.

## Partie 2 : Commandes & configuration

8. Initialiser : rs.initiate()
9. Ajouter un nœud : rs.add("host:port")
10. État du Replica Set : rs.status()
11. Identifier le rôle : rs.status() (champ *stateStr*)
12. Forcer un basculement : rs.stepDown()
13. Arbitre : rs.addArb("host:port") — utile pour briser les égalités lors des élections.
14. Secondary avec délai : configurer dans rs.conf() → slaveDelay: <seconds> puis rs.reconfig()

## Partie 3 : Résilience et tolérance aux pannes

15. Sans majorité : aucun Primary n'est élu → cluster en lecture seule.
16. Nouveau Primary : élu selon priorité, fraîcheur des données et disponibilité.
17. Élection : Processus automatique qui désigne un Primary.
18. Auto-dégradation : un nœud passe en Secondary s'il perd la majorité.
19. Nombre impair : facilite l'obtention d'une majorité.
20. Partition réseau : un côté perd la majorité → plus de Primary.

## Partie 4 : Scénarios pratiques

21. Primary injoignable : le Secondary devient Primary (si majorité avec l'Arbitre).
22. *slaveDelay 120s* : conserve une copie retardée → utile contre erreurs humaines (rollback).
23. Lecture toujours à jour : readConcern: "majority" + writeConcern: "majority".
24. WriteConcern : w:2
25. Donnée obsolète : réPLICATION asynchrone → éviter avec readPreference: "primary" ou readConcern: "majority".
26. Voir le Primary : rs.status() (stateStr="PRIMARY")
27. Bascule manuelle : rs.stepDown()
28. Ajouter un Secondary : installer MongoDB → ajouter via rs.add()
29. Retirer un nœud : rs.remove("host:port")
30. Nœud caché : hidden: true dans la config — utile pour analytics.
31. Modifier priorité : changer priority dans la config puis rs.reconfig().
32. Vérifier délai de réPLICATION : rs.printSlaveReplicationInfo()
33. rs.freeze() : empêche un nœud de devenir Primary temporairement.
34. Redémarrer sans perdre config : fichiers local conservés → simple restart du service.
35. Surveillance temps réel : logs mongod, rs.printReplicationInfo(),  
rs.status()

## Questions complémentaires :

37. **Arbitre** : ne stocke pas de données, ne fait que voter.
38. Latence : via rs.printSlaveReplicationInfo()
39. Retard précis : rs.printSlaveReplicationInfo()
40. RéPLICATION MongoDB : **asynchrone** (Secondaries ratrappent après coup).

41. Oui, possible avec `rs.reconfig()`
42. Secondary très en retard : risque de rollback ou indisponibilité.
43. Conflits : gérés par oplog et logique *last write wins* du Primary.
44. Plusieurs Primary ? **Non**, l'élection garantit l'unicité.
45. Secondary en écriture : risque d'incohérence → interdit.
46. Réseau instable : réélections fréquentes, pertes de Primary, lectures en retard.