

Fichier manquant : logo.png
(Image optionnelle)

Université Sorbonne Paris Nord
Département d'informatique

Réponses aux questions

Réplication et tolérance aux pannes avec
MongoDB

Denis Linde

8 décembre 2025

Table des matières

1	Partie 1 : Compréhension de base	3
2	Partie 2 : Commandes & configuration	3
3	Partie 3 : Résilience et tolérance aux pannes	4
4	Partie 4 : Scénarios pratiques	4
5	Questions complémentaires	5

1 Partie 1 : Compréhension de base

1. Qu'est-ce qu'un Replica Set dans MongoDB ?

Un Replica Set est un groupe d'instances `mongod` hébergeant le même jeu de données. Il assure la redondance et la haute disponibilité en cas de panne d'un serveur.

2. Quel est le rôle du Primary dans un Replica Set ?

Le Primary est le seul nœud autorisé à recevoir les écritures. Il enregistre toutes les modifications dans son journal d'opérations (`oplog`) pour qu'elles soient répliquées.

3. Quel est le rôle essentiel des Secondaries ?

Les Secondaries répliquent les données du Primary de manière asynchrone. Ils assurent la disponibilité par basculement (`failover`) si le Primary tombe et peuvent servir les opérations de lecture.

4. Pourquoi MongoDB n'autorise-t-il pas les écritures sur un Secondary ?

Pour garantir la cohérence des données. MongoDB utilise un modèle "Single Master" : seul le Primary valide l'ordre des opérations pour éviter les conflits d'écriture.

5. Qu'est-ce que la cohérence forte dans le contexte MongoDB ?

C'est la garantie qu'une lecture effectuée après une écriture renvoie toujours la donnée la plus récente. C'est le comportement par défaut lors des lectures sur le Primary.

6. Quelle est la différence entre `readPreference` : "primary" et "secondary" ?

- `primary` : Lecture uniquement sur le Primary (données garanties à jour).
- `secondary` : Lecture sur les Secondaries (risque de données obsolètes dû au délai de réplication).

7. Dans quel cas pourrait-on souhaiter lire sur un Secondary malgré les risques ?

Pour des tâches d'analyse ou de reporting lourdes ne nécessitant pas de données temps réel, afin de ne pas impacter les performances du Primary.

2 Partie 2 : Commandes & configuration

8. Quelle commande permet d'initialiser un Replica Set ?

```
> rs.initiate()
```

9. Comment ajouter un nœud à un Replica Set après son initialisation ?

```
> rs.add("hostname:port")
```

10. Quelle commande permet d'afficher l'état actuel du Replica Set ?

```
> rs.status()
```

11. Comment identifier le rôle actuel (Primary / Secondary / Arbitre) d'un nœud ?

Le prompt du shell l'indique généralement (ex : `PRIMARY>`). Sinon, la commande `rs.isMaster()` fournit le statut exact via les champs `ismaster` ou `secondary`.

12. Quelle commande permet de forcer le basculement du Primary ?

La commande `rs.stepDown()` oblige le Primary actuel à devenir Secondary, déclenchant ainsi une élection immédiate.

3 Partie 3 : Résilience et tolérance aux pannes

13. **Comment peut-on désigner un nœud comme Arbitre ? Pourquoi le faire ?**
Commande : `rs.addArb("host:port")`.

L'arbitre participe aux votes pour atteindre le quorum (majorité) mais ne stocke aucune donnée, ce qui économise les ressources de stockage.

14. **Donnez la commande pour configurer un nœud secondaire avec un délai de réplication (slaveDelay).**

Cela nécessite une reconfiguration du membre :

```
cfg = rs.conf()
cfg.members[n].secondaryDelaySecs = 3600 // delai de 1h
rs.reconfig(cfg)
```

15. **Que se passe-t-il si le Primary tombe en panne et qu'il n'y a pas de majorité ?**

Le Replica Set passe en mode lecture seule. Les Secondaries restants ne peuvent pas élire de nouveau Primary tant que le quorum n'est pas atteint.

16. **Comment MongoDB choisit-il un nouveau Primary ? Quels critères utilise-t-il ?**

Le critère principal est la fraîcheur des données : le Secondary ayant l'oplog le plus récent est favorisé. La priorité configurée et la connectivité jouent aussi un rôle.

17. **Qu'est-ce qu'une élection dans MongoDB ?**

C'est le processus automatique par lequel les membres éligibles votent pour désigner un nouveau Primary suite à une panne ou une maintenance.

18. **Que signifie auto-dégradation du Replica Set ? Dans quel cas cela survient-il ?**

Si le Primary perd la connexion avec la majorité du cluster, il se rétrograde automatiquement en Secondary (step down) pour éviter toute divergence de données (split-brain).

19. **Pourquoi est-il conseillé d'avoir un nombre impair de nœuds dans un Replica Set ?**

Pour garantir qu'une majorité stricte puisse toujours se dégager lors des votes et éviter les situations d'égalité.

20. **Quelles conséquences a une partition réseau sur le fonctionnement du cluster ?**

Le cluster se divise. La partition majoritaire continue de fonctionner (avec un Primary). La partition minoritaire passe en lecture seule (Secondaries) faute de quorum.

4 Partie 4 : Scénarios pratiques

21. **Vous avez 3 nœuds (Primary, Secondary, Arbitre). Que se passe-t-il si le Primary devient injoignable ?**

Le Secondary et l'Arbitre forment une majorité (2/3). Le Secondary sera élu nouveau Primary et le service continuera normalement.

22. **Vous avez configuré un Secondary avec un slaveDelay de 120 secondes. Quelle est son utilité ?**

Il sert de sécurité contre les erreurs humaines (ex : suppression de table). Le délai permet d'intervenir et de récupérer les données avant que l'erreur ne soit répliquée sur ce nœud.

23. **Un client exige une lecture toujours à jour, même en cas de bascule. Quelles options recommanderiez-vous ?**
Utiliser `readConcern: "majority"` (ou `"linearizable"`) couplé à un `writeConcern: "majority"` pour assurer que la donnée lue est validée par le cluster.
24. **Dans une application critique, vous voulez garantir que l'écriture est confirmée par au moins deux nœuds.**
Il faut spécifier l'option `w: 2` dans le Write Concern lors de l'écriture.
25. **Un étudiant a lu depuis un Secondary et récupéré une donnée obsolète. Expliquez pourquoi.**
Ceci est dû au délai de réplication (lag). L'écriture sur le Primary n'avait pas encore été propagée au Secondary au moment de la lecture.
26. **Montrez la commande pour vérifier quel nœud est actuellement Primary.**
- ```
> db.isMaster().primary
```
27. **Expliquez comment forcer une bascule manuelle du Primary sans interruption majeure.**  
En utilisant `rs.stepDown()`. Le Primary termine ses opérations en cours, ferme proprement les connexions et laisse sa place, minimisant le temps d'indisponibilité.
28. **Décrivez la procédure pour ajouter un nouveau nœud secondaire dans un Replica Set en fonctionnement.**  
1. Démarrer une instance `mongod` vide sur un nouveau port.  
2. Exécuter `rs.add("nouveau:port")` sur le Primary.  
3. La synchronisation initiale (Initial Sync) se lance automatiquement.
29. **Quelle commande permet de retirer un nœud défectueux d'un Replica Set ?**
- ```
> rs.remove("hostname:port")
```
30. **Comment configurer un nœud secondaire pour qu'il soit caché ? Pourquoi ?**
Configuration : `hidden: true` et `priority: 0`.
Usage : Pour dédier un nœud aux sauvegardes ou aux analyses sans qu'il ne soit sollicité par les clients standards.
31. **Montrez comment modifier la priorité d'un nœud afin qu'il devienne le Primary préféré.**
- ```
cfg = rs.conf()
cfg.members[0].priority = 2 // Plus élevé que les autres (
 default 1)
rs.reconfig(cfg)
```
32. **Expliquez comment vérifier le délai de réplication d'un Secondary par rapport au Primary.**  
La commande `rs.printSecondaryReplicationInfo()` affiche directement le retard temporel pour chaque membre du set.

## 5 Questions complémentaires

33. **Que fait la commande `rs.freeze()` et dans quel scénario est-elle utile ?**  
Elle empêche un nœud de devenir Primary pendant une durée définie. Utile lors d'opérations de maintenance pour s'assurer qu'un nœud spécifique reste Secondary.

34. **Comment redémarrer un Replica Set sans perdre la configuration ?**  
La configuration est stockée de manière persistante dans la base `local`. Tant que le dossier de données (`dbpath`) est conservé, le cluster redémarre avec sa configuration intacte.
35. **Expliquez comment surveiller en temps réel la réplication.**  
L'outil `mongostat` permet de visualiser l'état des nœuds en temps réel. Les logs MongoDB fournissent également des détails précis sur les événements de réplication.
36. **Qu'est-ce un Arbitre (Arbiter) et pourquoi ne stocke-t-il pas de données ?**  
C'est un membre qui participe exclusivement aux votes. Ne stockant pas de données, il permet d'atteindre le quorum à moindre coût (CPU/Disque) sans gérer la réplication.
37. **Comment vérifier la latence de réplication entre le Primary et les Secondaries ?**  
Via `rs.status()`, en observant le champ `pingMs` (latence réseau) et en comparant les timestamps `optimeDate` des membres.
38. **Quelle commande MongoDB permet d'afficher le retard de réplication des membres secondaires ?**  
`rs.printSecondaryReplicationInfo()`.
39. **Quelle est la différence entre la réplication asynchrone et synchrone ? Quel type utilise MongoDB ?**  
En asynchrone, le Primary valide l'écriture sans attendre les Secondaries (rapide). En synchrone, il attend la confirmation (sûr mais lent). MongoDB est **asynchrone** par défaut, mais peut simuler du synchrone via le `writeConcern`.
40. **Peut-on modifier la configuration d'un Replica Set sans redémarrer les serveurs ?**  
Oui, la commande `rs.reconfig()` permet d'appliquer une nouvelle configuration à chaud.
41. **Que se passe-t-il si un nœud Secondary est en retard de plusieurs minutes ?**  
Tant que le retard est couvert par l'oplog, il rattrape son retard. Si l'oplog a tourné, le nœud passe en état `RECOVERING` et doit resynchroniser toutes les données.
42. **Comment MongoDB gère-t-il les conflits de données lors de la réplication ?**  
Seul un Primary est autorisé et comme le Primary est le seul à avoir le droit d'écrire il n'y a virtuellement aucun conflit possible.
43. **Est-il possible d'avoir plusieurs Primarys simultanément dans un Replica Set ? Pourquoi ?**  
Théoriquement non. En cas de partition réseau ("Split Brain"), deux nœuds peuvent se croire Primary, mais seul celui possédant la majorité pourra valider les écritures `majority`.
44. **Pourquoi est-il déconseillé d'utiliser un Secondary pour des opérations d'écriture même en lecture préférée secondaire ?**  
Ce n'est pas déconseillé, c'est impossible. Le Secondary rejettera systématiquement l'écriture car il ne gère pas l'oplog.
45. **Quelles sont les conséquences d'un réseau instable sur un Replica Set ?**  
Cela entraîne des élections fréquentes ("flapping") et des changements de Primary intempestifs, provoquant des interruptions de service et des latences élevées.