**REPUBLIQUE DU SENEGAL**

**\***

**ECOLE SUPERIEURE MULTINATIONALE DES TELECOMMUNICATIONS**

**\*\*\*\*\***

Classe :**INGC2**

**Thème: Étude et mise en œuvre du replicaset avec MongoDB**

**Présenté par :**

Sophie Basse SENE

**Professeur :** M.PREIRA

Année Académique : **2022-2023**

**PLAN**

**Ⅰ. INTRODUCTION**

**Ⅱ. Concepts de Base**

**Ⅲ. Mécanismes Internes**

**Ⅳ. Prérequis**

**Ⅴ. Configuration des Instances**

**Ⅵ. Tests et Résultats**

**Ⅰ. INTRODUCTION**

MongoDB est un système de gestion de base de données NoSQL orienté document. Il utilise des documents JSON (ou BSON) avec des schémas dynamiques, ce qui le rend flexible et performant pour diverses applications. MongoDB permet de stocker des données non structurées de manière efficace et offre une scalabilité horizontale grâce à son architecture distribuée.

L’objectif principal d’un système de gestion de base de données comme MongoDB est de conserver vos données de manière claire et sécurisée. Dans cette optique, les ReplicaSets MongoDB constituent un outil important pour la protection des données : la méthode consiste à dupliquer à l’identique les jeux de données et à les répartir sur plusieurs nœuds et sur différents serveurs.

**Ⅱ. Concepts de Base**

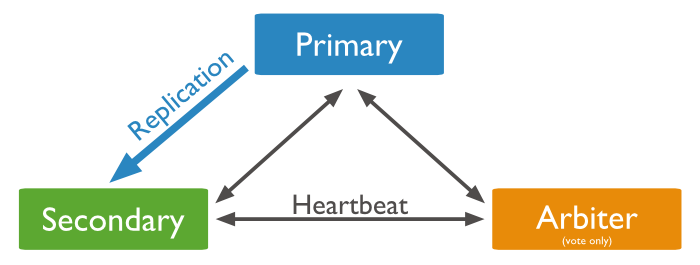
Un Replicaset MongoDB se compose donc d’un nœud primaire (ou maître) et de plusieurs nœuds secondaires (ou esclaves) créés à partir de copies exactes du maître. Il doit y avoir au moins deux nœuds secondaires, mais il est recommandé un minimum de trois copies. Le transfert de données se fait toujours **exclusivement du nœud primaire vers les nœuds secondaires**. Si le nœud maître devait tomber en panne à cause d’une erreur, d’une défaillance du système ou de travaux de maintenance, un nœud esclave prendrait aussitôt la place du maître afin de garantir la disponibilité des données pendant la durée de la panne. Une fois le nœud défectueux réparé, le nœud esclave redevient un nœud secondaire.

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Toutefois, si l’on reste sur une architecture de la sorte, nous ne sommes pas tolérant aux pannes puisque le serveur primaire reste un point critique. Pour cela, nous allons rajouter un arbitre "*arbiter*", dont la tâche est de vérifier l’état du réseau. Ainsi, dès que le primary tombe, un système de vote va permettre d’élire le secondary qui remplacera le serveur tombé en panne. Nous sommes donc tranquilles : cette architecture tient tant qu’au moins un serveur tourne. C’est ce que l’on appelle un **ReplicaSet**.

Un *ReplicaSet* doit contenir au minimum 3 serveurs (1 Primary et 2 Secondary) pour garantir un minimum de tolérance aux pannes. Un ReplicaSet peut contenir jusqu’à 50 serveurs ; toutefois lorsqu’un vote a lieu pour élire le nouveau Primary, au maximum 7 peuvent participer à cette élection (les premiers ayant répondu à l’Arbiter).



**Stratégie de déploiement**  
C’est à vous de définir la stratégie de déploiement des serveurs physiques en fonction de votre architecture réseau. Il est préférable de mettre deux serveurs sur un même rack (remplacement du *primary*), dans le même *datacenter*, et au moins un dans un autre *datacenter* pour garantir la non perte de données.

Nous pourrons par ailleurs associer des niveaux de priorité à chaque serveur dans le cas d’un vote.

**Ⅲ. Mécanismes Internes**

### **Élection du Primaire**

Lorsqu'un ReplicaSet est initialisé ou lorsqu'un primaire existant échoue, une élection est déclenchée pour choisir un nouveau primaire. Les membres secondaires se consultent et votent pour élire le nouveau primaire. Celui qui reçoit la majorité des votes devient le primaire.

### **Propagation des Écritures**

Le primaire journalise chaque opération d'écriture dans un log d'opérations (oplog) et applique l'opération à sa base de données locale. Les membres secondaires répliquent cet oplog et appliquent les mêmes opérations, garantissant ainsi la cohérence des données à travers le ReplicaSet.

### **Synchronisation Initiale**

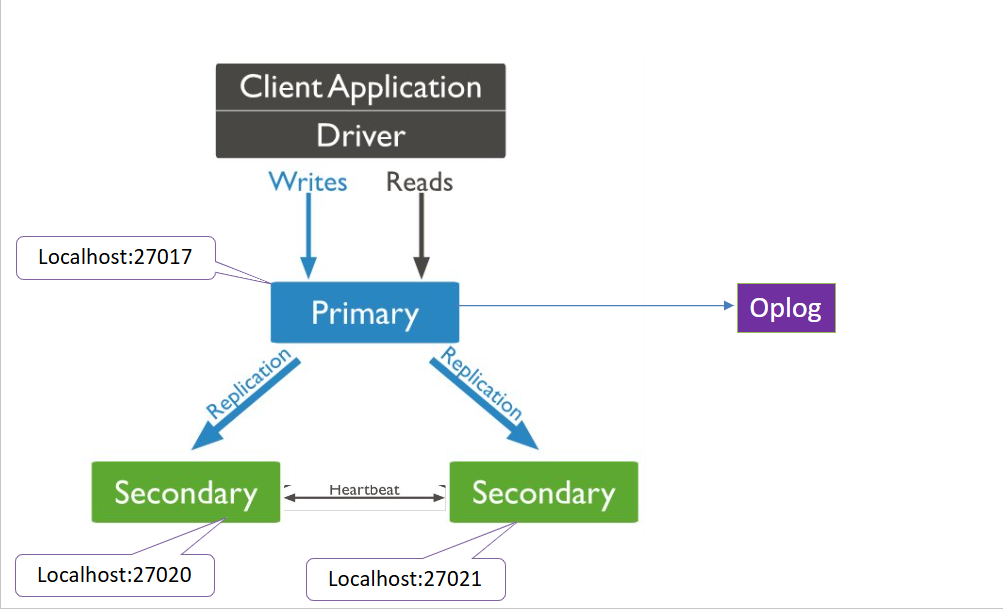
Quand un nouveau membre rejoint le ReplicaSet ou qu'un membre est désynchronisé, une synchronisation initiale est nécessaire. Le membre secondaire copie l'intégralité des données de la source (souvent le primaire) puis applique les opérations manquantes à partir de l'oplog pour se mettre à jour.

### **Confirmation d'Écriture (Write Concern)**

MongoDB offre différents niveaux de garantie pour les opérations d'écriture. Par exemple, **w:1** confirme l'écriture après la journalisation sur le primaire, tandis que **w:majority** confirme après la journalisation sur la majorité des membres du ReplicaSet.

### **Détection et Récupération de Pannes**

Les membres du ReplicaSet surveillent en permanence la disponibilité des autres membres. En cas de défaillance d'un membre, les autres peuvent élire un nouveau primaire et continuer à fonctionner sans intervention manuelle.



**Avantages des ReplicaSets**

**Haute disponibilité**: Les ReplicaSets offrent une redondance des données en maintenant plusieurs copies synchronisées des données. En cas de défaillance d'un nœud, un autre nœud peut prendre le relais, assurant ainsi une disponibilité continue des données.

**Mise à jour en continu**: Les ReplicaSets permettent des mises à jour sans temps d'arrêt en redirigeant le trafic vers les nœuds sains pendant les opérations de maintenance ou de mise à jour.

**Tolérance aux pannes**: Grâce à la redondance des données, les ReplicaSets sont capables de résister aux pannes matérielles ou logicielles d'un ou plusieurs nœuds sans perdre de données.

**Flexibilité de déploiement**: Les ReplicaSets offrent une flexibilité pour déployer des bases de données dans divers environnements, que ce soit sur site, dans le cloud ou dans des configurations hybrides.

**Lecture scalable**: Les ReplicaSets permettent de distribuer la charge de lecture en redirigeant les requêtes de lecture vers les nœuds secondaires, ce qui améliore les performances globales du système et la capacité à gérer un grand nombre de requêtes de lecture.

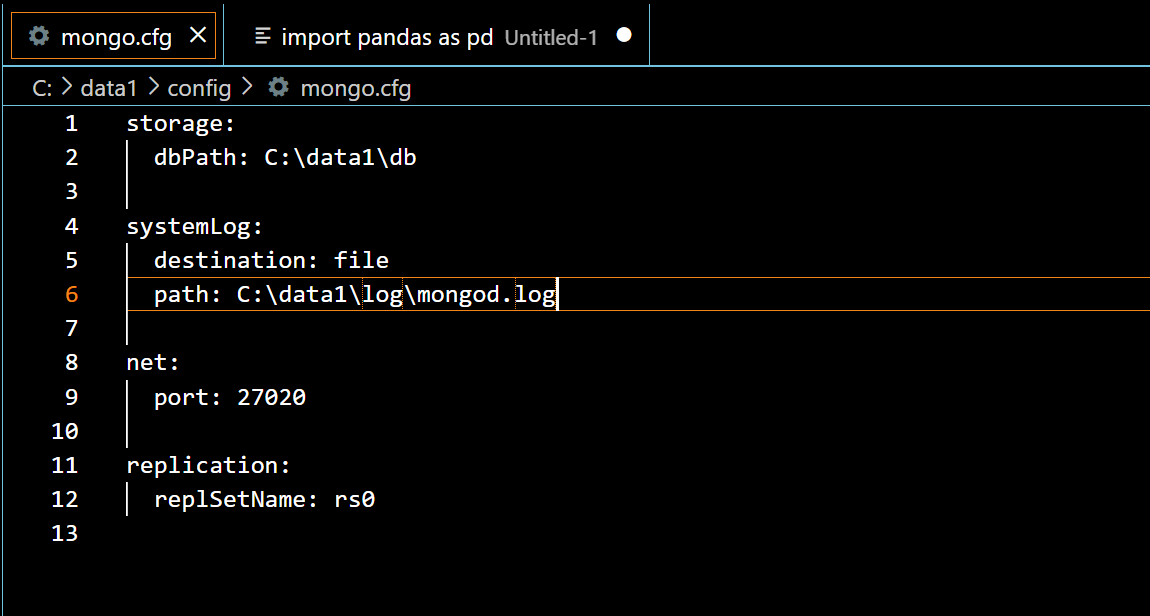
**Ⅳ. Prérequis**

* Installation de MongoDB sur chaque machine ([Installation de MongoDB sur Windows)](https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/)
* Accès réseau entre les serveurs pour permettre la communication entre les membres
* Configuration des ports : MongoDB utilise par défaut le port 27017, qui doit être ouvert et accessible.

**Ⅴ. Configuration des Instances**

Nous allons simuler un ReplicaSet localement. Pour cela, les paramètres de chaque serveur doivent être définis :

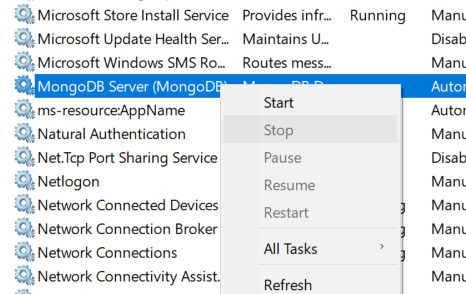
* Créer un répertoire dédié : **/data1, /data2** pour respectivement les deux serveurs
* Dans chacun d'eux nous créons trois dossiers configuration config , db et log
* Dans chaque dossier "config", on crée un fichier "mongo.cfg"



**Initialisation du replicaset**

**Arrêtez MongoDB**

Dans la barre de recherche , recherchez “Services”.Dans la fenêtre Services, localisez le service « MongoDB Server », cliquez dessus avec le bouton droit de la souris et sélectionnez « Arrêter » comme indiqué dans l’image.Windows + Rservices.msc



**Configuration du serveur principal MongoDB**

Commencez par ouvrir l’invite de commande en tant qu’administrateur. Accédez au répertoire dans lequel MongoDB est installé, qui dans ce cas est “C:\Program Files\MongoDB\Server\7.0\bin Files\MongoDB\Server\7.0\bin” ajouter ce chemin directement dans les variables d’environnement

Ensuite, exécutez la commande suivante pour démarrer le serveur primaire et configurer le replicaset nommé 'rs0' sur :localhost:27017

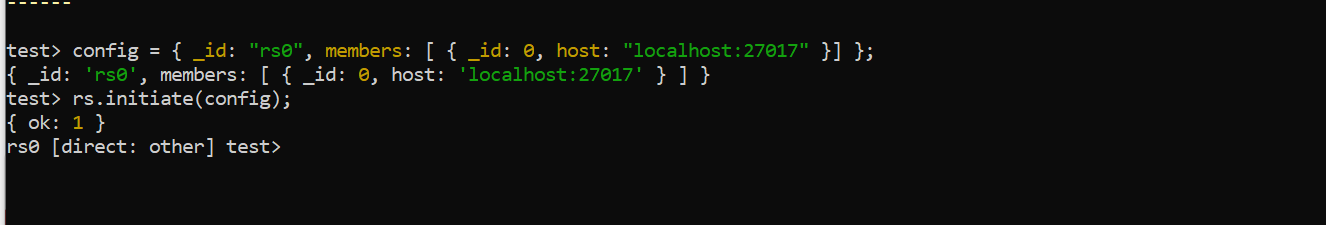
**mongod --dbpath "C:\Program Files\MongoDB\Server\7.0\data" --logpath "C:\Program Files\MongoDB\Server\7.0\log\mongod.log" --port 27017 --storageEngine=wiredTiger --replSet rs0**

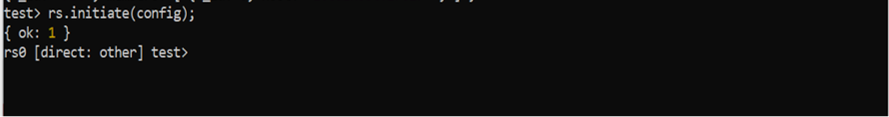
Cette commande définit le chemin d’accès à la base de données et l’emplacement du fichier journal pour le service MongoDB, spécifie le numéro de port, sélectionne le moteur de stockage WiredTiger et initialise la configuration du j réplicas set.

Ouvrez une autre invite de commande en tant qu’administrateur et démarrez le shell MongoDB en exécutant :

**mongosh --port 27017**

Ensuite, configurez le serveur principal avec les commandes suivantes :



Enfin, lancez le replicaset avec cette commande : Ces étapes établiront le serveur principal de votre réplicas set MongoDB nommé « rs0 ».

**Configuration des serveurs secondaires**

Pour établir les serveurs secondaires de votre replicaset MongoDB, procédez comme suit :

* Ouvrez une invite de commandes en tant qu’administrateur.
* Accédez au répertoire dans lequel MongoDB est installé, comme mentionné précédemment.
* Exécutez la commande suivante pour démarrer le premier serveur secondaire avec les chemins d’accès aux données et aux journaux, le numéro de port, le moteur de stockage et le nom du replicaset spécifiés :

**mongod --dbpath "C:\data1\db" --logpath "C:\data1\log\mongod.log" --port 27020 --storageEngine=wiredTiger --replSet rs0**

Répétez le processus pour le deuxième serveur secondaire, en utilisant un autre port (27021) :

**mongod --dbpath "C:\data2\db" --logpath "C:\data2\log\mongod.log" --port 27021 --storageEngine=wiredTiger --replSet rs0**

Ensuite, nous intégrerons les serveurs secondaires dans le replicaSet .

* Ouvrez une invite de commandes en tant qu’administrateur.
* Connectez-vous au serveur principal via le shell MongoDB en exécutant :

mongosh --port 27017

* Ajoutez les serveurs secondaires au replicaset avec les commandes suivantes :

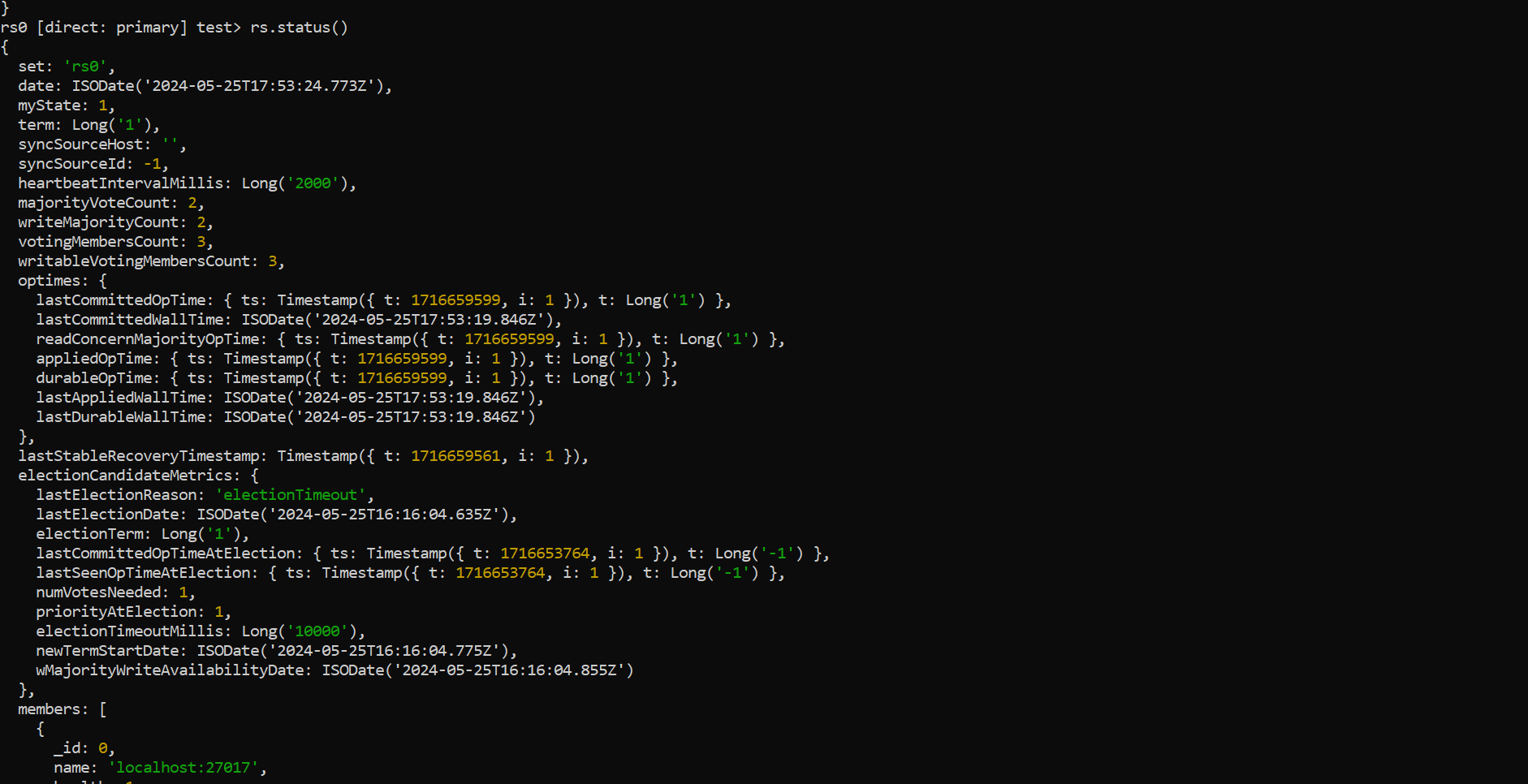
Une image contenant texte, capture d’écran, Logiciel multimédia, Police

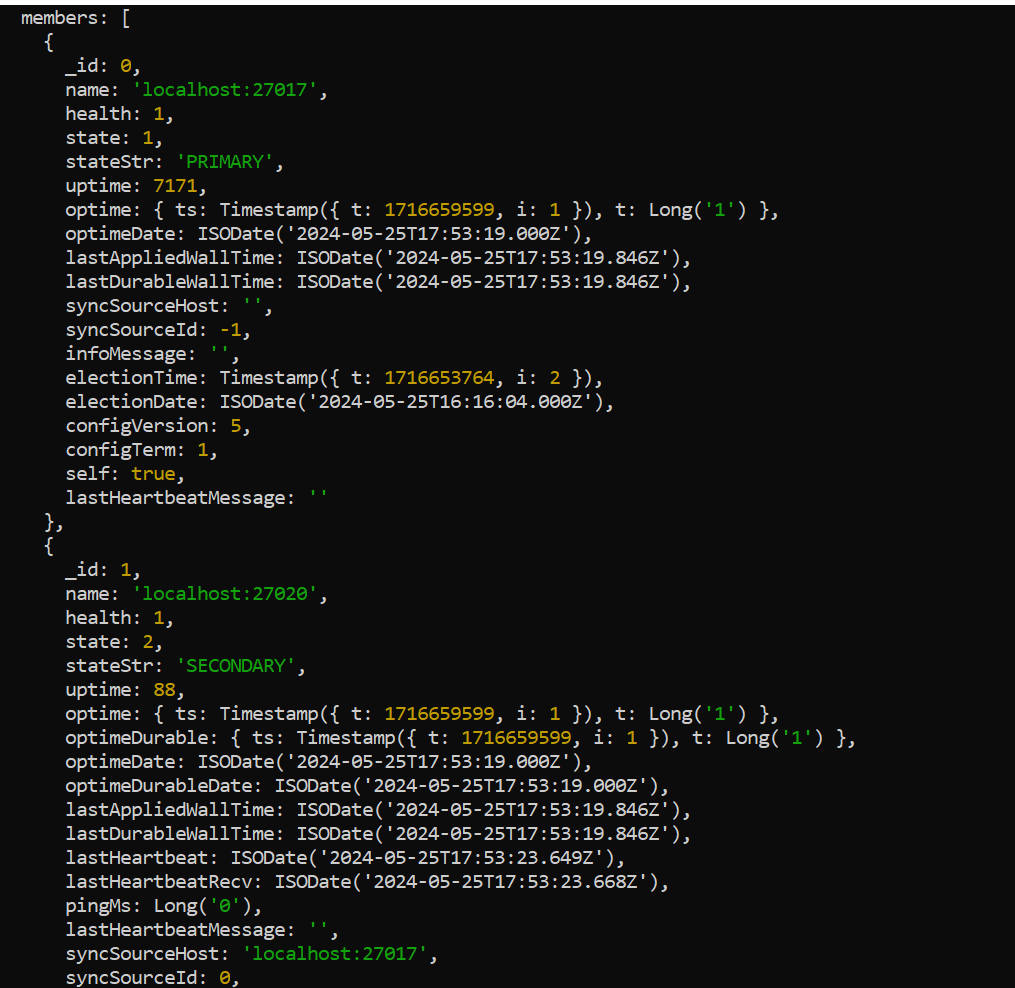
Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquement

* Vérifiez la configuration en exécutant :

rs.status()





Cela affichera l’état du replicaset , confirmant l’ajout des serveurs secondaires.

Pour permettre aux serveurs secondaires de lire à partir du serveur principal. ouvrez une invite de commandes en tant qu’administrateur pour chaque serveur.

Dans le shell MongoDB pour chacun, définissez la préférence de lecture sur 'secondary' pour autoriser les opérations de lecture à partir de ces serveurs. Les commandes sont les suivantes :

Pour le serveur sur le port 27020 :

**mongosh --port 27020**  
**db.getMongo().setReadPref('secondary')**

Pour le serveur sur le port 27021 :

**mongosh --port 27021**  
**db.getMongo().setReadPref('secondary')**

Exécutez ces commandes pour vous assurer que les serveurs secondaires peuvent gérer les opérations de lecture, en maintenant l’efficacité et la fiabilité de votre je réplicas set MongoDB.

#### **Définir l’arbitre**

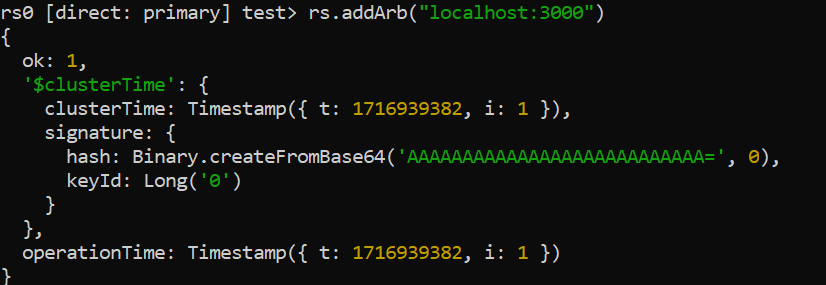
Pour départager les votes, un arbitre sera nécessaire pour désigner le *PRIMARY*. Pour ce faire :

Créons le répertoire c:**/data3**

Exécutons un arbitre en spécifiant le port **30000** :

mongod --port 30000 --dbpath /data3/mongo.cfg --replSet rs0

Ajoutons l'arbitre dans la console d’administration :



Pour consulter le statut actuel du ReplicaSet, il suffit de faire l’instruction suivante : rs.status()Une image contenant texte, capture d’écran

Description générée automatiquement

**Ⅵ. Tests et Résultats**

Ajouter une publication dans la base de données dblp.

Une image contenant texte, capture d’écran, affichage, logiciel

Description générée automatiquement

Vérifier l'ajout dans le serveur secondaire

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

La publication a bien été ajoutée.

Essayer de l'ajouter directement dans le serveur secondaire

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Nous n'avons pas droit à l'écriture.

Arrêter le serveur primaire avec la commande rs.stepDown()

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Il passe directement en serveur secondaire, l’un des serveurs secondaires prend le relai.

Documentation:

<https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4474611-protegez-vous-des-pannes-avec-les-replicaset>

[Create a MongoDB replica set in Windows | by Mubin Mohajer | Medium](https://medium.com/@mubinxyz/create-a-mongodb-replica-set-in-windows-45851f0e59a7)

<http://andreiarion.github.io/TP7_MongoDB_Replication_exercices>

<file:///C:/Users/hp/Downloads/ilide.info-tp3-replicaset-sur-mongodb-lx27architecture-du-replicaset-pr_da285a30a333a41d60481f4cb38b39bc.pdf>