**Table des matières**

[Partie 1 – MongoDB Standalone 2](#_Toc200982026)

[1. Méthode de déploiement 2](#_Toc200982027)

[Option A : Installation native sur une machine Linux (Ubuntu) 2](#_Toc200982028)

[Option B : Déploiement dans un conteneur Docker 2](#_Toc200982029)

[2. Activation de l’authentification et création d’un utilisateur admin 2](#_Toc200982030)

[Activer l’authentification 2](#_Toc200982031)

[Créer un utilisateur administrateur 3](#_Toc200982032)

[3. Connexion à la base 3](#_Toc200982033)

[Via la CLI (mongosh) 3](#_Toc200982034)

[Via MongoDB Compass (interface graphique) 3](#_Toc200982035)

[4. Création de la base de données *testdb* et de la collection *test* 4](#_Toc200982036)

[5. Opérations sur les documents 4](#_Toc200982037)

[Insertion de documents 4](#_Toc200982038)

[Requête simple 4](#_Toc200982039)

[Mise à jour d’un document 4](#_Toc200982040)

[Suppression d’un document 4](#_Toc200982041)

[6. Vérification du fonctionnement 5](#_Toc200982042)

[Partie 2 – MongoDB Replica Set 6](#_Toc200982043)

[1. Déploiement du Replica Set 6](#_Toc200982044)

[1.1 Créer un réseau Docker dédié 6](#_Toc200982045)

[1.2 Lancer les 3 containers MongoDB 6](#_Toc200982046)

[2. Initialisation du Replica Set 6](#_Toc200982047)

[2.1 Se connecter au premier conteneur (mongo1) 6](#_Toc200982048)

[2.2 Initialiser le replica set 6](#_Toc200982049)

[2.3 Vérification de l’état du replica set 6](#_Toc200982050)

[3. Utilisation du Replica Set 7](#_Toc200982051)

[3.1 Insérer un document sur le PRIMARY 7](#_Toc200982052)

[3.2 Lire depuis un SECONDARY 7](#_Toc200982053)

[4. Connexions possibles 7](#_Toc200982054)

[URI standard de connexion à un Replica Set : 7](#_Toc200982055)

[URI avec préférences de lecture (read preference) 7](#_Toc200982056)

[5. Modes de lecture et écriture dans un Replica Set 8](#_Toc200982057)

[Écriture (write concern) 8](#_Toc200982058)

[Lecture (read preference) 8](#_Toc200982059)

[6. Exemple de script complet d’initialisation 8](#_Toc200982060)

[init-replica.js : 8](#_Toc200982061)

[Exécution : 9](#_Toc200982062)

# Partie 1 – MongoDB Standalone

## 1. Méthode de déploiement

### Option A : Installation native sur une machine Linux (Ubuntu)

1. Ajouter la clé publique MongoDB et le dépôt officiel :

wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -

echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list

1. Mettre à jour les paquets et installer MongoDB :

sudo apt-get update

sudo apt-get install -y mongodb-org

1. Démarrer et activer MongoDB au démarrage :

sudo systemctl start mongod

sudo systemctl enable mongod

### Option B : Déploiement dans un conteneur Docker

Exécuter la commande suivante pour lancer un conteneur MongoDB standalone :

docker run -d --name mongodb-standalone -p 27017:27017 mongo:6.0

## 2. Activation de l’authentification et création d’un utilisateur admin

### Activer l’authentification

1. Modifier le fichier de configuration MongoDB (par défaut /etc/mongod.conf sur Linux) en ajoutant :

security:

authorization: "enabled"

1. Redémarrer MongoDB pour appliquer la modification :

sudo systemctl restart mongod

### Créer un utilisateur administrateur

Avant d’activer l’authentification, lancer mongosh :

mongosh

Dans le shell MongoDB, exécuter :

use admin

db.createUser({

user: "admin",

pwd: "motdepassefort",

roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase" ]

})

## 3. Connexion à la base

### Via la CLI (mongosh)

mongosh -u admin -p motdepassefort --authenticationDatabase admin

### Via MongoDB Compass (interface graphique)

* Hostname : localhost
* Port : 27017
* Authentication : Username / Password
* Username : admin
* Password : motdepassefort
* Authentication Database : admin

## 4. Création de la base de données *testdb* et de la collection *test*

Dans le shell MongoDB connecté en admin :

use testdb

db.createCollection("test")

## 5. Opérations sur les documents

### Insertion de documents

db.test.insertMany([

{ nom: "Alice", age: 30 },

{ nom: "Bob", age: 25 },

{ nom: "Charlie", age: 35 }

])

### Requête simple

Afficher tous les documents avec age supérieur à 28 :

db.test.find({ age: { $gt: 28 } })

### Mise à jour d’un document

Modifier l’âge de "Alice" à 31 ans :

db.test.updateOne(

{ nom: "Alice" },

{ $set: { age: 31 } }

)

### Suppression d’un document

Supprimer le document où nom est "Bob" :

db.test.deleteOne({ nom: "Bob" })

## 6. Vérification du fonctionnement

* Utiliser mongosh avec authentification pour tester les commandes.
* Utiliser MongoDB Compass avec les paramètres de connexion pour vérifier la base et les données via interface graphique.

# Partie 2 – MongoDB Replica Set

## 1. Déploiement du Replica Set

*Option : Utilisation de Docker (recommandé pour la simplicité)*

### 1.1 Créer un réseau Docker dédié

docker network create mongo-net

### 1.2 Lancer les 3 containers MongoDB

docker run -d --name mongo1 --net mongo-net -p 27017:27017 mongo --replSet rs0

docker run -d --name mongo2 --net mongo-net mongo --replSet rs0

docker run -d --name mongo3 --net mongo-net mongo --replSet rs0

## 2. Initialisation du Replica Set

### 2.1 Se connecter au premier conteneur (mongo1)

docker exec -it mongo1 mongosh

### 2.2 Initialiser le replica set

rs.initiate({

\_id: "rs0",

members: [

{ \_id: 0, host: "mongo1:27017" },

{ \_id: 1, host: "mongo2:27017" },

{ \_id: 2, host: "mongo3:27017" }

]

})

### 2.3 Vérification de l’état du replica set

rs.status()

Vous devriez voir un PRIMARY et deux SECONDARY dans la sortie.

# 3. Utilisation du Replica Set

## 3.1 Insérer un document sur le PRIMARY

Dans mongosh connecté à *mongo1* (PRIMARY) :

use testdb

db.test.insertOne({ nom: "Replica Test", valeur: 42 })

## 3.2 Lire depuis un SECONDARY

Se connecter à un SECONDARY (par exemple *mongo2*) :

docker exec -it mongo2 mongosh

Par défaut, les SECONDARY n’acceptent pas la lecture directe. Pour activer :

rs.slaveOk()

db.test.find()

## 4. Connexions possibles

### URI standard de connexion à un Replica Set :

mongosh "mongodb://mongo1:27017,mongo2:27017,mongo3:27017/testdb?replicaSet=rs0"

### URI avec préférences de lecture (read preference)

* Lire à partir du PRIMARY (par défaut) :

mongosh "mongodb://mongo1:27017,mongo2:27017,mongo3:27017/testdb?replicaSet=rs0&readPreference=primary"

* Lire à partir d’un SECONDARY :

mongosh "mongodb://mongo1:27017,mongo2:27017,mongo3:27017/testdb?replicaSet=rs0&readPreference=secondary"

## 5. Modes de lecture et écriture dans un Replica Set

### Écriture (write concern)

* Les écritures se font **uniquement sur le PRIMARY**.
* Niveau de garantie configuré avec w :
  + w: 1 : écrit sur PRIMARY uniquement.
  + w: "majority" : écrit sur la majorité des nœuds.

### Lecture (read preference)

* primary : (défaut) lit depuis le PRIMARY.
* primaryPreferred : lit depuis PRIMARY si possible, sinon SECONDARY.
* secondary : lit exclusivement depuis un SECONDARY.
* secondaryPreferred : lit depuis SECONDARY si possible.
* nearest : lit depuis le nœud avec la latence la plus faible.

## 6. Exemple de script complet d’initialisation

### init-replica.js :

rs.initiate({

\_id: "rs0",

members: [

{ \_id: 0, host: "mongo1:27017" },

{ \_id: 1, host: "mongo2:27017" },

{ \_id: 2, host: "mongo3:27017" }

]

})

### Exécution :

docker exec -it mongo1 mongosh < init-replica.js

## Bonus : Script docker-compose pour déployer automatiquement un Replica Set MongoDB à 3 membres avec initialisation intégrée

version: '3.8'

services:

mongo1:

image: mongo:6.0

container\_name: mongo1

restart: always

ports:

- 27017:27017

command: ["mongod", "--replSet", "rs0", "--bind\_ip\_all"]

networks:

- mongo-net

mongo2:

image: mongo:6.0

container\_name: mongo2

restart: always

ports:

- 27018:27017

command: ["mongod", "--replSet", "rs0", "--bind\_ip\_all"]

networks:

- mongo-net

mongo3:

image: mongo:6.0

container\_name: mongo3

restart: always

ports:

- 27019:27017

command: ["mongod", "--replSet", "rs0", "--bind\_ip\_all"]

networks:

- mongo-net

networks:

mongo-net:

driver: bridge

### Script d'initialisation du Replica Set

Crée un fichier init-replica.js dans le même dossier :

rs.initiate({

\_id: "rs0",

members: [

{ \_id: 0, host: "mongo1:27017" },

{ \_id: 1, host: "mongo2:27017" },

{ \_id: 2, host: "mongo3:27017" }

]

})

### Instructions pour lancer le cluster et initialiser

1) Lancer les containers :

docker-compose up -d  
2) Attendre que les containers soient bien démarrés

3) Initialiser le replica set via mongo1 :

docker exec -it mongo1 mongosh < init-replica.js

4) Vérifier le statut

docker exec -it mongo1 mongosh -c "rs.status()"

### Connexion

Tu peux te connecter à ce replica set avec l’URL :

mongosh "mongodb://localhost:27017,localhost:27018,localhost:27019/?replicaSet=rs0"

# Partie 3 : Intégration dans une application Node.js

## 1. Dépendances utilisées

* **Node.js** (v14+ recommandé)
* **mongodb** : le driver officiel MongoDB pour Node.js

Installation :

npm init -y

npm install mongodb

## 2. Code de connexion à MongoDB avec commentaires

const { MongoClient } = require('mongodb');

// Exemple d'URI pour standalone sans authentification

const uriStandalone = "mongodb://localhost:27017";

// Exemple d'URI pour replica set avec auth et préférences de lecture

const uriReplicaSet = "mongodb://admin:motdepassefort@localhost:27017,localhost:27018,localhost:27019/testdb?replicaSet=rs0&readPreference=primary";

// Fonction pour connecter à MongoDB

async function connectToMongo(uri) {

const client = new MongoClient(uri, {

useUnifiedTopology: true, // utilisation du moteur de topologie moderne

});

try {

await client.connect();

console.log("Connecté à MongoDB");

return client;

} catch (err) {

console.error("Erreur de connexion :", err);

process.exit(1);

}

}

**Explications URI :**

* mongodb:// : protocole de connexion
* Liste d’hôtes séparés par des virgules (pour replica set)
* Option replicaSet=rs0 : indique qu’on se connecte à un replica set nommé rs0
* readPreference=primary : lecture sur le primaire (optionnel)
* Authentification avec admin:motdepassefort@ (si activée)

## 3. Code des actions (insertion, requêtes, update, suppression)

async function run() {

// Change ici selon ta config : standalone ou replica set

const uri = uriReplicaSet;

const client = await connectToMongo(uri);

const db = client.db("testdb");

const collection = db.collection("test");

// Insertion de documents

const insertResult = await collection.insertMany([

{ nom: "Alice", age: 30 },

{ nom: "Bob", age: 25 },

{ nom: "Charlie", age: 35 }

]);

console.log(`${insertResult.insertedCount} documents insérés`);

// Requête simple avec filtre

const results = await collection.find({ age: { $gt: 28 } }).toArray();

console.log("Documents avec age > 28 :", results);

// Mise à jour

const updateResult = await collection.updateOne(

{ nom: "Alice" },

{ $set: { age: 31 } }

);

console.log(`Documents modifiés : ${updateResult.modifiedCount}`);

// Suppression

const deleteResult = await collection.deleteOne({ nom: "Bob" });

console.log(`Documents supprimés : ${deleteResult.deletedCount}`);

// Fermeture de la connexion

await client.close();

console.log("Connexion fermée");

}

run().catch(console.error);

## 4. Méthodes de connexion possibles

* Connexion simple à un serveur standalone

const uri = "mongodb://localhost:27017";

* Connexion à un replica set

const uri = "mongodb://host1,host2,host3/dbname?replicaSet=rs0";

* Connexion avec authentification

const uri = "mongodb://user:password@host1,host2,host3/dbname?replicaSet=rs0";

Options supplémentaires

* readPreference : primary, secondary, etc.
* w : niveau de write concern (ex: w=majority

## 5. Résultats des tests

Exemple de sortie console après exécution :

Connecté à MongoDB

3 documents insérés

Documents avec age > 28 : [ { nom: 'Alice', age: 30, \_id: ... }, { nom: 'Charlie', age: 35, \_id: ... } ]

Documents modifiés : 1

Documents supprimés : 1

Connexion fermée