Compte Rendu Technique - Livrable 3

DRAVET Timothée - FISA INFO 4A

Phase 3 : Distribution et Replica Set

1 Introduction

Cette troisième phase marque le passage d'une instance MongoDB unique à une architecture distribuée en **Replica Set**. L'enjeu est de garantir la continuité de service du projet CineExplorer, même en cas de défaillance matérielle d'un ou plusieurs serveurs de base de données.

2 Migration des données

Pour cette étape, la migration des données vers l'infrastructure distribuée a été réalisée en utilisant exactement la même méthode que celle employée précédemment (insertion par lots via le driver PyMongo), assurant ainsi la cohérence du jeu de données entre les différentes phases du projet.

3 Architecture du Replica Set

3.1 Configuration logicielle

Nous avons déployé un cluster composé de trois nœuds. Cette configuration est le standard minimal pour assurer une tolérance aux pannes efficace tout en maintenant un mécanisme d'élection fonctionnel.

Les nœuds sont configurés comme suit :

— **Primaire (initial) :** localhost :27017
— **Secondaire 1 :** localhost :27018
— **Secondaire 2 :** localhost :27019

3.2 Initialisation du cluster

Le script d'initialisation utilise la commande d'administration `replSetInitiate`. Voici l'implémentation de la configuration :

Listing 1 – Configuration et Initialisation (init_replica.py)

```
from pymongo import MongoClient

client = MongoClient("localhost:27017", directConnection=True)

config = {
    '_id': "rs0",
    'members': [
        {'_id': 0, 'host': 'localhost:27017'},
        {'_id': 1, 'host': 'localhost:27018'},
        {'_id': 2, 'host': 'localhost:27019'}
    ]
}

try:
    client.admin.command("replSetInitiate", config)
    print("Replica Set initialise avec succes.")
except Exception as e:
    print(f"Statut : {e}")
```

4 Tests de Failover et Analyse de Résilience

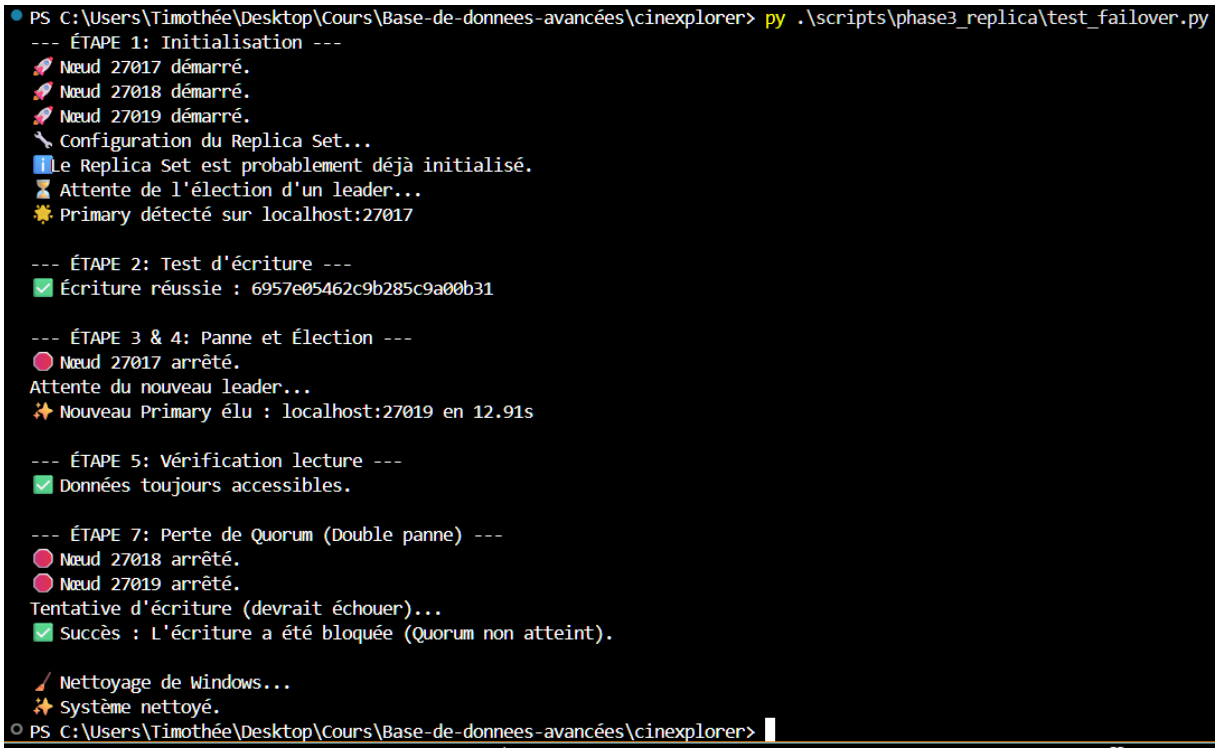
4.1 Protocole de test

Le script `test_failover.py` a été conçu pour automatiser la détection de panne. Il suit un cycle de vie critique : écriture initiale, coupure du nœud maître, mesure du temps de réélection, et vérification de la persistance.

4.2 Résultats du Test de Basculement

Le résultat de l'exécution montre une excellente réactivité du système :

- **Élection du nouveau Primary :** Après l'arrêt du port 27017, le cluster a élu le port **27019** comme nouveau leader en **12.91 secondes**.
- **Intégrité des données :** Les données écrites avant la panne sur le nœud 27017 ont été automatiquement répliquées sur le nœud 27019 et sont restées accessibles immédiatement après l'élection.



4.3 Compréhension du Quorum et Sécurité des données

Le test a poussé la résilience dans ses retranchements en simulant une double panne (arrêt des ports 27018 et 27019).

Analyse théorique : MongoDB utilise un système de vote majoritaire. Avec un seul nœud restant sur trois, le **Quorum** (calculé par la formule $n/2+1$) n'est plus atteint. Le système, par sécurité, empêche toute écriture pour éviter le phénomène de *Split-Brain* (où deux nœuds croiraient être maîtres simultanément). Le test confirme ce comportement :

```
Succès : L'écriture a été bloquée (Quorum non atteint).
```

5 Monitoring du système

Afin de vérifier en temps réel l'état de santé de nos membres, nous utilisons un script de statut détaillé :

Listing 2 – Vérification de l'état des membres

```
from pymongo import MongoClient

client = MongoClient("localhost:27017", directConnection=True)

status = client.admin.command("replSetGetStatus")
print(f"Cluster : {status['set']}")

for member in status['members']:
    print(f"Machine : {member['name']} | Role : {member['stateStr
        ']}")
```

6 Conclusion

La Phase 3 démontre que notre base de données n'est plus un point de défaillance unique. La transition vers un Replica Set permet d'absorber la perte d'un serveur sans interruption de service pour l'utilisateur final, tout en garantissant une cohérence stricte des données grâce au respect des règles de majorité (Quorum).

Dans le rendu final nous explorerons les possibilités en termes d'interface web à l'aide de Django.