Rapport Livrable 3

Ibrahim BENDRISS Polytech INFO 4A

Objectif :

L'objectif de cette tâche est de déployer MongoDB en Replica Set afin d'assurer la haute disponibilité : un nœud est élu PRIMARY (écriture), les autres sont SECONDARY (réplication). En cas de panne du PRIMARY, un SECONDARY peut être élu automatiquement.

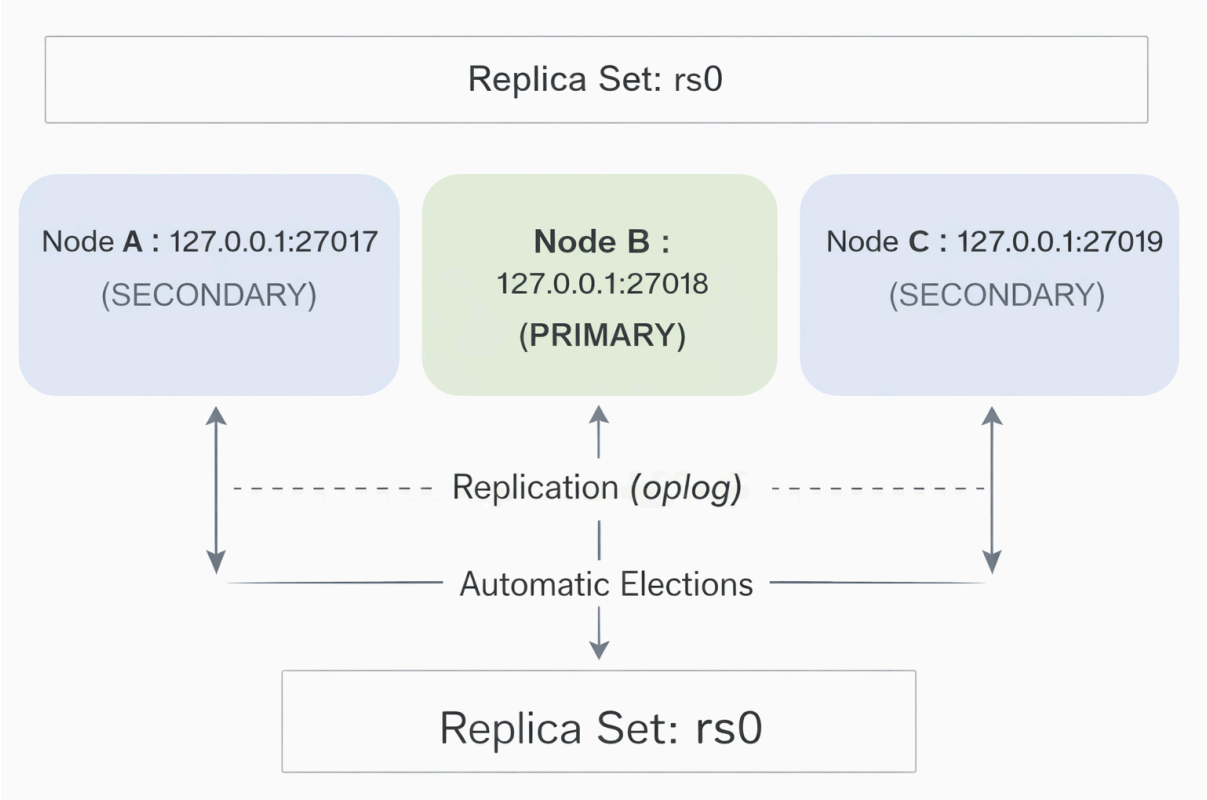
T3.1 - Initialisation du Replica Set :

Architecture (schéma) :

Nœud 1 : 127.0.0.1:27017 → data : data/mongo/db-1

Nœud 2 : 127.0.0.1:27018 → data : data/mongo/db-2

Nœud 3 : 127.0.0.1:27019 → data : data/mongo/db-3



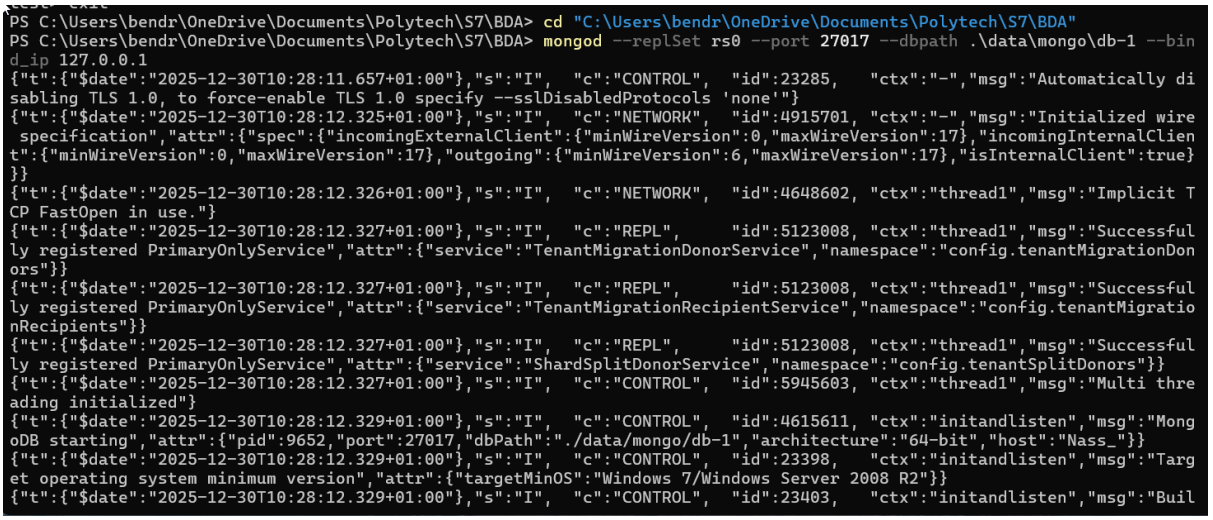
Rapport Livrable 3

Ibrahim BENDRISS Polytech INFO 4A

Lancement des trois instances mongod

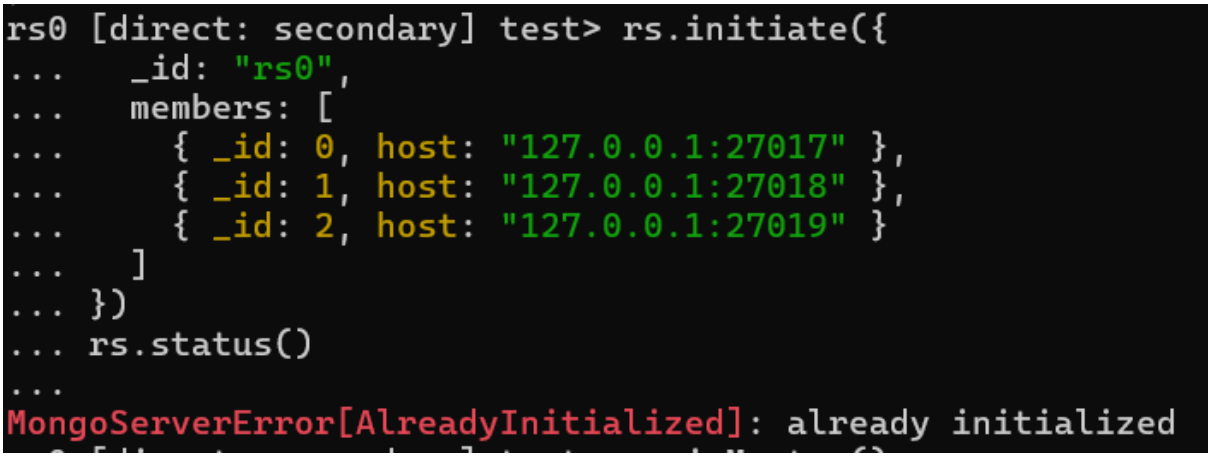
Chaque instance MongoDB est lancée sur un port différent et avec un dbpath dédié. Depuis la racine du projet :

mongod --replSet rs0 --port 27017 --dbpath .\data\mongo\db-1 --bind_ip 127.0.0.1 :



pareil pour –port 27018 et 27019.

Dès que les ports sont actifs, on initialise :



Rapport Livrable 3

Ibrahim BENDRISS Polytech INFO 4A

Une fois les trois serveurs démarrés, l'initialisation est effectuée via mongosh connecté à un nœud :

```
PS C:\Users\bendr> mongosh --port 27017
Current Mongosh Log ID: 69539bc8e02e668eb91e2620
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.5.10
Using MongoDB:          6.0.27
Using Mongosh:          2.5.10

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

------
   The server generated these startup warnings when booting
   2025-12-30T10:28:12.525+01:00: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------
```

Le soucis c'est que rs0 sur 27017 est déclaré comme secondary, ce qu'il faut faire c'est appelé la commande rs.isMaster() ou db.hello(). Sur mon cas, vu qu'un noeud a été élu automatiquement je devais mettre une autre commande pour le trouver :

```
PS C:\Users\bendr> mongosh --port 27017 --quiet --eval "db.hello().isWritablePrimary"
false
PS C:\Users\bendr> mongosh --port 27018 --quiet --eval "db.hello().isWritablePrimary"
true
PS C:\Users\bendr> mongosh --port 27019 --quiet --eval "db.hello().isWritablePrimary"
false
PS C:\Users\bendr>
```

le noeud 2 est le noeud primaire donc c'est chez lui que nous fairons les tests de pannes plus tard dans ce document.

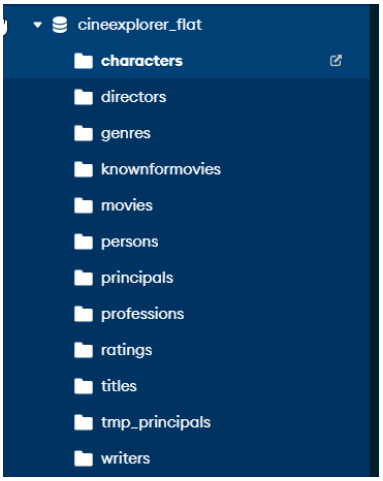
On peut désormais importer les données :

```
C:\Users\bendr\AppData\Local\Python\pythoncore-3.14-64\python.exe: can't open file 'C:\\Users\\bendr\\OneDrive\\Document
s\\Polytech\\S7\\BDA\\phase2_mongodb\\migrate_flat.py': [Errno 2] No such file or directory
PS C:\Users\bendr\OneDrive\Documents\Polytech\S7\BDA> python .\script\phase2_mongodb\migrate_flat.py `
>>   --sqlite ".\data\imdb.db" `
>>   --mongo-uri "mongodb://127.0.0.1:27017,127.0.0.1:27018,127.0.0.1:27019/?replicaSet=rs0" `
>>   --mongo-db "cineexplorer_flat" `
>>   --drop `
>>   --pk-as-id
SQLite DB: .\data\imdb.db
Mongo URI: mongodb://127.0.0.1:27017,127.0.0.1:27018,127.0.0.1:27019/?replicaSet=rs0
Mongo DB : cineexplorer_flat
Tables   : 12
------------------------------------------------------------
characters                     sqlite=  152888 mongo=  152888 inserted=  152888 => OK
directors                      sqlite=   40933 mongo=   40933 inserted=   40933 => OK
genres                         sqlite=   85426 mongo=   85426 inserted=   85426 => OK
knownformovies                 sqlite=  252924 mongo=  252924 inserted=  252924 => OK
movies                         sqlite=   36859 mongo=   36859 inserted=   36859 => OK
persons                        sqlite=  145847 mongo=  145847 inserted=  145847 => OK
principals                     sqlite=  361858 mongo=  361858 inserted=  361858 => OK
professions                    sqlite=  318014 mongo=  318014 inserted=  318014 => OK
ratings                        sqlite=   36859 mongo=   36859 inserted=   36859 => OK
titles                         sqlite=  680196 mongo=  680196 inserted=  680196 => OK
tmp_principals                 sqlite=  361863 mongo=  361863 inserted=  361863 => OK
writers                        sqlite=   82855 mongo=   82855 inserted=   82855 => OK
```

Rapport Livrable 3

Ibrahim BENDRISS Polytech INFO 4A

On check si c'est fait :

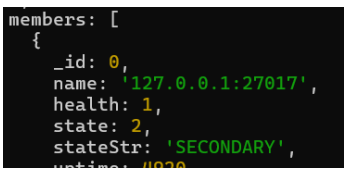


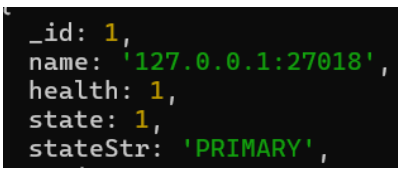
Ça a l'air bon !

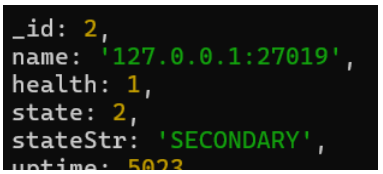
T3.2 - Test de panne :

1.

Les 3 noeuds existes et 27018 est le noeud primaire.







Rapport Livrable 3

Ibrahim BENDRISS Polytech INFO 4A

2.

En insérant les document on voit que tout le monde a l'accès.

```
rs0 [primary] test> use ha_test
... db.events.insertOne({
...   tag: "before_failure",
...   ts: new Date(),
...   msg: "write on replica set",
...   from: "client"
... })
...
switched to db ha_test
```

```
rs0 [direct: primary] ha_test> db.events.insertOne({ tag:"check_before", ts:new Date(), note:"insert before failure" })
{
  acknowledged: true,
  insertedId: ObjectId('6953c4d3ae589d6dc81e2621')
}
```

```
rs0 [direct: primary] ha_test> db.events.find({tag:"check_before"}).sort({ts:-1}).limit(1)
[
  {
    _id: ObjectId('6953c4d3ae589d6dc81e2621'),
    tag: 'check_before',
    ts: ISODate('2025-12-30T12:25:55.848Z'),
    note: 'insert before failure'
  }
]
```

```
rs0 [primary] ha_test> rs.printSecondaryReplicationInfo()
source: 127.0.0.1:27017
{
  syncedTo: 'Tue Dec 30 2025 11:53:34 GMT+0100 (heure normale d'Europe centrale)',
  replLag: '0 secs (0 hrs) behind the primary '
}
---
source: 127.0.0.1:27019
{
  syncedTo: 'Tue Dec 30 2025 11:53:34 GMT+0100 (heure normale d'Europe centrale)',
  replLag: '0 secs (0 hrs) behind the primary '
}
```

3.

J'ai fermé le terminal avec le port 27018 qui était en primary et voilà le résultat :

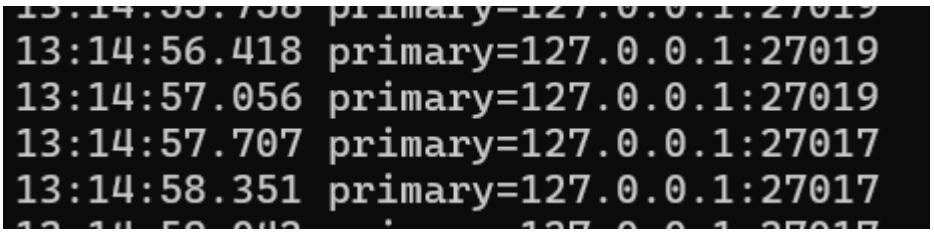
```
PS C:\Users\bendr> $uri="mongodb://127.0.0.1:27017,127.0.0.1:27018,127.0.0.1:27019/?replicaSet=rs0"
PS C:\Users\bendr> mongosh "$uri" --quiet --eval "db.hello()"
{
  topologyVersion: {
    processId: ObjectId('69539b2c3fe003f4de57b667'),
    counter: Long('18')
  },
  hosts: [ '127.0.0.1:27017', '127.0.0.1:27018', '127.0.0.1:27019' ],
  setName: 'rs0',
  setVersion: 1,
  isWritablePrimary: true,
  secondary: false,
  primary: '127.0.0.1:27017',
  me: '127.0.0.1:27017',
```

le port 27017 est passé en primary.

Rapport Livrable 3

Ibrahim BENDRISS Polytech INFO 4A

4.



Temps d'élection : 13:14:57.707 - 13:14:57.056 = 651 millisecondes

```
rs0 [direct: primary] ha_test> db.events.find({tag:"check_before"}).sort({ts:-1}).limit(1)
[
  {
    _id: ObjectId('6953c4d3ae589d6dc81e2621'),
    tag: 'check_before',
    ts: ISODate('2025-12-30T12:25:55.848Z'),
    note: 'insert before failure'
  }
]
```

Les données sont intactes.

5.

```
rs0 [direct: primary] ha_test> db.events.findOne({ tag: "before_failure" })
{
  _id: ObjectId('6953c734ae589d6dc81e2622'),
  tag: 'before_failure',
  ts: ISODate('2025-12-30T12:36:04.472Z'),
  msg: 'insert before failover'
}
rs0 [direct: primary] ha_test> db.events.find({ tag: "before_failure" }).sort({ ts: -1 }).limit(1).toArray()
[
  {
    _id: ObjectId('6953c734ae589d6dc81e2622'),
    tag: 'before_failure',
    ts: ISODate('2025-12-30T12:36:04.472Z'),
    msg: 'insert before failover'
  }
]
```

Les données sont accessibles.

Rapport Livrable 3

Ibrahim BENDRISS Polytech INFO 4A

6.

```
members: [
  {
    _id: 0,
    name: '127.0.0.1:27017',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 1994,
    optime: { ts: Timestamp({ t: 1767098438, i: 318 }), t: Long('14') },
    optimeDate: ISODate('2025-12-30T12:40:38.000Z'),
    lastAppliedWallTime: ISODate('2025-12-30T12:40:38.529Z'),
    lastDurableWallTime: ISODate('2025-12-30T12:40:38.529Z'),
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    electionTime: Timestamp({ t: 1767096897, i: 1 }),
    electionDate: ISODate('2025-12-30T12:14:57.000Z'),
    configVersion: 1,
    configTerm: 14,
    self: true,
    lastHeartbeatMessage: ''
  },
  {
    _id: 1,
    name: '127.0.0.1:27018',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 1668,
    optime: { ts: Timestamp({ t: 1767098438, i: 318 }), t: Long('14') },
    optimeDurable: { ts: Timestamp({ t: 1767098438, i: 318 }), t: Long('14') },
    optimeDate: ISODate('2025-12-30T12:40:38.000Z'),
    optimeDurableDate: ISODate('2025-12-30T12:40:38.000Z'),
    lastAppliedWallTime: ISODate('2025-12-30T12:40:38.529Z'),
    lastDurableWallTime: ISODate('2025-12-30T12:40:38.529Z'),
    lastHeartbeat: ISODate('2025-12-30T12:40:50.147Z'),
    lastHeartbeatRecv: ISODate('2025-12-30T12:40:51.401Z'),
    pingMs: Long('0'),
    lastHeartbeatMessage: '',
    syncSourceHost: '127.0.0.1:27017',
    syncSourceId: 0,
```

le port 27018 a sa reconnexion est bien passé en Secondary.

```
switched to db ha_test
rs0 [direct: primary] ha_test> use ha_test
... db.events.countDocuments()
...
already on db ha_test
rs0 [direct: primary] ha_test> db.events.countDocuments()
2
rs0 [direct: primary] ha_test> rs.printSecondaryReplicationInfo()
source: 127.0.0.1:27018
{
  syncedTo: 'Tue Dec 30 2025 13:41:57 GMT+0100 (heure normale d'Europe centrale)',
  replLag: '0 secs (0 hrs) behind the primary '
}
---
source: 127.0.0.1:27019
{
  syncedTo: 'Tue Dec 30 2025 13:41:57 GMT+0100 (heure normale d'Europe centrale)',
  replLag: '0 secs (0 hrs) behind the primary '
}
rs0 [direct: primary] ha_test>
```

La synchronisation est bien faites sur les 2 replicats secondaires.

Rapport Livrable 3

Ibrahim BENDRISS Polytech INFO 4A

7.

J'ai shutdown les ports 27017 et 27018, il reste plus que 27019 mais le soucis en se connectant on trouve :

```
rs0 [direct: secondary] test> rs.status()
```

Il est toujours secondaire et pas primaire. Ce qui veut dire qu'il peut pas accepter l'écriture mais uniquement la lecture.

Donc si je veux ajouter une date :

```
rs0 [direct: secondary] ha_test> db.events.insertOne({ tag: "write_try", ts: new Date(), msg: "should fail on secondary" })
MongoServerError[NotWritablePrimary]: not primary
```

Il dira not primary car il est secondaire donc aucune écriture par contre si je veux lire un document c'est possible il l'affiche :

```
rs0 [direct: secondary] test> use ha_test
switched to db ha_test
rs0 [direct: secondary] ha_test> db.events.findOne({tag:"before_failure"})
{
  _id: ObjectId('6953c734ae589d6dc81e2622'),
  tag: 'before_failure',
  ts: ISODate('2025-12-30T12:36:04.472Z'),
  msg: 'insert before failover'
}
rs0 [direct: secondary] ha_test>
```

T3.3 - Django :

Avec Django, on fait une installation avec les commandes suivantes :

```
PS C:\Users\bendr\OneDrive\Documents\Polytech\S7\BDA> pip install django pymongo
```

et on créer un environnement :

```
PS C:\Users\bendr\OneDrive\Documents\Polytech\S7\BDA> python -m venv .venv
```

et on crée la configuration et movies :

```
PS C:\Users\bendr\OneDrive\Documents\Polytech\S7\BDA> django-admin startproject config .
PS C:\Users\bendr\OneDrive\Documents\Polytech\S7\BDA> python manage.py startapp movies
```

À l'aide des différents script,
On obtient une vue JSON :


```
{
  "sqlite": {
    "path": "http://127.0.0.1:8000/stats/",
    "tables_counts_sample": {
      "movies": 36859,
      "persons": 145847,
      "characters": 152888,
      "knownformovies": 252924,
      "genres": 85426,
      "ratings": 36859,
      "principals": 361858,
      "directors": 40933,
      "writers": 82855,
      "professions": 318014,
      "titles": 680196,
      "tmp_principals": 361863
    }
  },
  "mongo": {
    "db": "from settings.MONGO_DB_NAME",
    "collections_counts_sample": {
      "directors": 40933,
      "tmp_principals": 361863,
      "persons": 145847,
      "writers": 82855,
      "professions": 318014,
      "characters": 152888,
      "ratings": 36859,
      "knownformovies": 252924,
      "movies": 36859,
      "titles": 680196,
      "genres": 85426,
      "principals": 361858
    },
    "hello": {
      "me": "127.0.0.1:27019",
      "primary": "127.0.0.1:27019",
      "isWritablePrimary": true,
      "setName": "rs0"
    }
  }
}
```


On voit qu'on a la base de donnée SQLite et Mongo, on verra également le hello qui affiche le noeud primaire qui est ici le port 27019.

Conclusion :

Au cours de ce livrable, nous avons mis en place une architecture MongoDB tolérante aux pannes en déployant un replica set rs0 composé de trois nœuds (ports 27017, 27018 et 27019), chacun avec son répertoire de données dédié. Après le démarrage des trois instances mongod, le cluster a été initialisé avec rs.initiate() et nous avons vérifié le bon fonctionnement de l'élection PRIMARY/SECONDARY via db.hello() / mongosh.

Nous avons ensuite réalisé une série de tests de tolérance aux pannes (T3.2) : insertion d'un document de test, arrêt brutal du PRIMARY (simulation réaliste d'une panne), observation de l'élection d'un nouveau PRIMARY, vérification que les données restent accessibles (lecture) et que le service peut reprendre après reconnexion. Le scénario de double panne a également montré la limite attendue du replica set : sans quorum, aucun PRIMARY n'est élu, ce qui empêche les écritures, tandis que les lectures restent possibles depuis un nœud SECONDARY.

Enfin, la partie préparation Django (T3.3) a permis de créer le projet (config) et l'application (movies), puis de mettre en place deux services d'accès aux données : un service SQLite et un service MongoDB (connexion au replica set). Une vue de test /stats/ a été développée pour afficher des statistiques sous forme de JSON (counts de tables SQLite et de collections MongoDB), et la section hello confirme que l'application se connecte correctement au replica set et au nœud PRIMARY.

Globalement, ce livrable valide une infrastructure MongoDB fonctionnelle et robuste, ainsi qu'une base applicative Django prête à exploiter simultanément SQLite (source) et MongoDB (cible) dans la suite du projet.