OCR - PROJET 7

Concevez et analysez une base de données NoSQL

Mathilde LE SOLLIEC
12-2025

SOMMAIRE

Partie 1

1. Contexte

Crash des données d'une association

L'association NosCités, surveille les plateformes web de locations de courte durée pour mesurer l'impact qu'elles ont sur l'offre de logements dans plusieurs villes en France (Paris et Lyon).

Ils font un inventaire (listing) en scrapant les données des sites comme AirBnb.

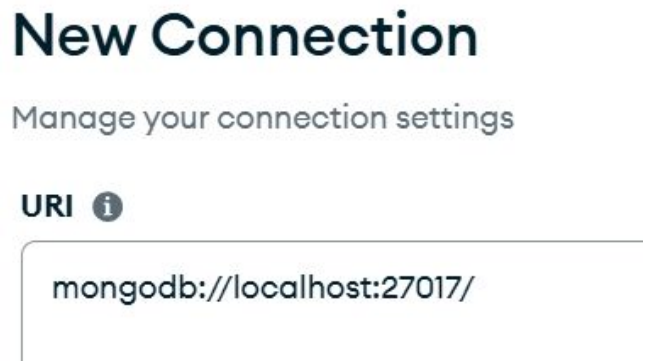
- Un crash total de la base de données des locations à Paris à eu lieu
- Les données sont dans des serveurs, pas dans le cloud.
- Une sauvegarde a été faite avant le crash, l'enjeux est donc :
  1. Restaurer la base de données MongoDB
  2. Analyser son intégrité par toute une série de requêtes
  3. Pérenniser son utilisation pour prévenir de futurs incidents

2. Chargement des données

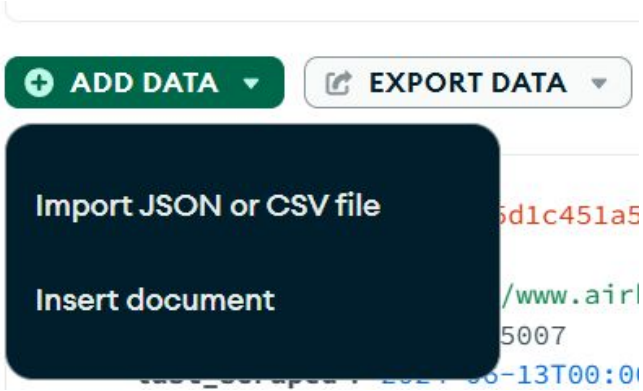
Importation des données sauvegardées

Téléchargement des documents via la plateforme mongo compass.

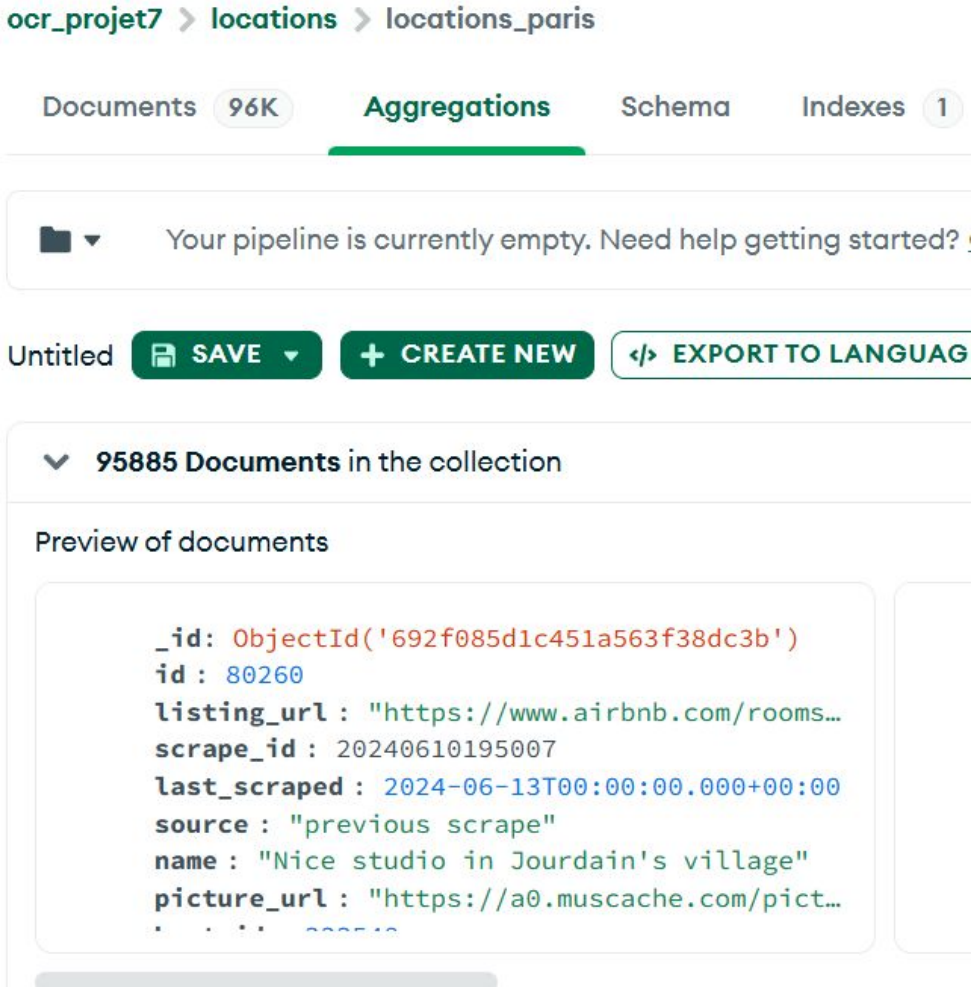
1- créer un connection “locations_paris”



2- Ajouter les données



Importation des données sauvegardées



Etat des lieux de la collection

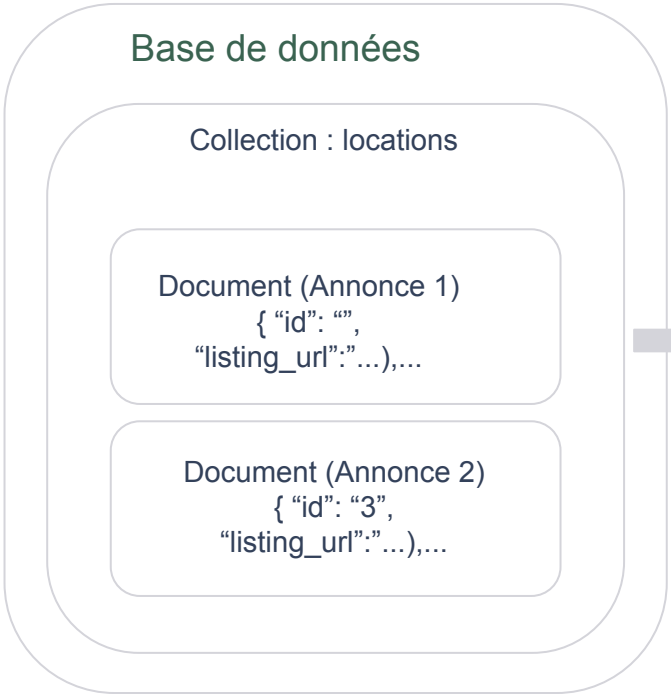
95 885
Documents - Annonces

90 173
Logements libres

```
> db.locations_paris.countDocuments()
< 95885
> db.locations_paris.countDocuments( {has_availability : true})
< 90173
```

MongoDB shell (dans mongo compass)

Collection : 1 document par annonce airbnb avec des informations sur le logements, l'hôte, les reviews...



Chaque document contient différentes caractéristiques (40-50 caracteristiques)
Certaines données sont imbriquées (amenities) :

```
{    "id": 80260,
  "listing_url": "string",
  "scrape_id": 20240610195007,
  "last_scraped": "2024-06-13T00:00:00.000+00:00",
  "source": "string",
  "name": "string",
  "picture_url": "string",
   "host_id": 333548,
   "host_url": "string",
…
  "amenities" : 'Json",
  "price": 120,
  "latitude": 48.8566,
  "longitude": 2.3522,
  "amenities": son,
  "availability_30": 12,
  "number_of_reviews": 37,
  "reviews_per_month": 0.82,
  ...
}
```

Création d'un MongoDB

Pourquoi MongoDB

Souplesse du schéma

Pas de modèle fixe, chaque locations peut avoir des champs très différents.
On peut facilement avoir des champs imbriqués (amenities, host)

Scalabilité et performance

Si l'activité augmente, la base s'agrandit sans interruption
Requêtes rapide
Accès fluide aux informations

Analyse orientée documents

Et non relations (SQL)

Partie 2

3. Requêtes simples (mongoshell)

Requête

- Combien d'annonces y a-t-il par type de location ? (corriger : property type)

```
> db.locations_paris.aggregate([
    { $group: { _id: "$property_type", num_loc: { $sum: 1 } } },
    { $sort: { num_loc: -1 } }
  ])
< {
    _id: 'Entire rental unit',
    num_loc: 80516
  }
  {
    _id: 'Private room in rental unit',
    num_loc: 5980
  }
  {
    _id: 'Entire condo',
    num_loc: 2679
  }
  {
    _id: 'Room in boutique hotel',
    num_loc: 1256
```

Requête

- Quelles sont les 5 annonces de location avec le plus d’ évaluations ? Et combien d’évaluations ont-elles ?

```
locations> db.locations_paris.find({}, {name:1,number_of_reviews:1}).sort({number_of_reviews:-1}).limit(5)
[
  {
    _id: ObjectId('692f08611c451a563f3913d6'),
    name: 'Sweet & cosy room next to Canal Saint Martin ❤️',
    number_of_reviews: 3067
  },
  {
    _id: ObjectId('692f08631c451a563f392cb0'),
    name: 'Double/Twin Room, close to Opera and the Louvre with breakfast included',
    number_of_reviews: 2620
  },
  {
    _id: ObjectId('692f08661c451a563f395150'),
    name: 'Bed in Dorm of 8 Beds "The Big One" in Paris',
    number_of_reviews: 2294
  },
  {
    _id: ObjectId('692f08661c451a563f394f38'),
    name: 'Comfortable bed in shared rooms of 8 in Paris 12e',
    number_of_reviews: 2105
  },
  {
    _id: ObjectId('692f08641c451a563f3940af'),
    name: 'Nice Room for 2 people',
    number_of_reviews: 2048
```

Requête

- Quel est le nombre total d’hôtes différents ?

```
> db.locations_paris.distinct("host_id").length
< 71979
```

- Quel est le nombre de locations réservables instantanément ? Cela représente quelle proportion des annonces ?

```
> db.locations_paris.countDocuments({ instant_bookable: true })
< 22094
> db.locations_paris.countDocuments({ instant_bookable: true })/db.locations_paris.countDocuments({})
< 0.23042185951921573
```

Requête

- Est-ce que des hôtes ont plus de 100 annonces sur les plateformes ? Et si oui qui sont-ils ?

```
> db.locations_paris.distinct("host_id", { host_total_listings_count: { $gt: 100 } }).length
< 113
> db.locations_paris.distinct("host_id", { host_total_listings_count: { $gt: 100 } })
< [
      152242,    939656,   1112584,   2107478,   2503671,   3837356,
     3971743,   4777697,   5027164,   7178965,   7642792,  10725710,
    11593703,  12938211,  12984381,  13013633,  21630783,  22805631,
    24262798,  24495283,  24554647,  26981054,  28313443,  33889201,
    42776295,  50502817,  50978178,  51567288,  62509540,  64224220,
    67879895,  97916688,  99040006, 107243549, 110327807, 114422088,
   117238503, 125797498, 128268770, 129135089, 137504779, 137510244,
   141523466, 141539709, 151077629, 153911376, 154045024, 156775524,
   168305291, 169499449, 179013722, 184945495, 186423516, 193397737,
   204662484, 220191808, 222375907, 226958437, 227131021, 245844815,
   270230184, 300765938, 309430151, 311372920, 314994947, 316050198,
```

- Cela représente quel pourcentage des hôtes ? 0,15%

```
> const nb_big_host = db.locations_paris.distinct("host_id",{host_total_listings_count: {$gt: 100} }).length
> const nb_host =db.locations_paris.distinct("host_id").length
> nb_big_host / nb_host
< 0.0015699023326247934
```

Requête

- Combien y a-t-il de super hôtes différents ?

```
> db.locations_paris.distinct("host_id", { host_is_superhost: true }).length
< 10027
```

- Cela représente quel pourcentage des hôtes ?

```
> const nb_superhost = db.locations_paris.distinct("host_id",{host_is_superhost: true}).length
> nb_superhost/nb_host
< 0.13930451937370622
```

4. Requêtes avec Polars

Outils utilisés

PyMongo	Connexion et extraction des données depuis MongoDB.
Polars	Manipulation et analyse efficace des données tabulaires (calculs, agrégations).

Polars pour analyser et transformer des données déjà chargées en mémoire, avec des calculs rapides et efficaces.

Dans la pratique, on extrait d'abord les données avec MongoDB puis on les traite avec Polars pour les analyses.

```python
# Requêtes
# 1- Taux de réservation moyen par mois par type de logement
avg_reviews_by_type = df.group_by("property_type").agg(
    pl.col("reviews_per_month").mean().alias("avg_reviews_per_month")
)
```

Taux moyen de réservation par type de logement :
shape: (70, 2)

property_type --- str	avg_reviews_per_month --- f64
Cave	0.25
Entire guest suite	1.623333
Private room in guesthouse	1.985333
Shared room in cabin	null
Shared room in bed and breakfa…	0.08
…	…
Room in bed and breakfast	1.192222
Private room in villa	0.45
Private room in serviced apart…	1.32
Private room in loft	1.205606
Shared room in guest suite	0.45

```python
# 2- Médiane du nombre d’avis pour tous les logements
median_reviews = df.select(pl.col("number_of_reviews").median().alias("median_reviews"))
print("Médiane du nombre d'avis :", median_reviews)
```

```
Médiane du nombre d'avis : shape: (1, 1)
┌────────────────┐
│ median_reviews │
│ ---            │
│ f64            │
╞════════════════╡
│ 3.0            │
└────────────────┘
```

```python
# 3- Médiane du nombre d’avis par catégorie d’hôte
median_reviews_host = df.group_by("host_is_superhost").agg(
    pl.col("number_of_reviews").median().alias("median_reviews")
)
print("Médiane des avis par catégorie d'hôte :")
print(median_reviews_host)
```

```
Médiane des avis par catégorie d'hôte :
shape: (3, 2)
```

host_is_superhost	median_reviews
bool	f64
false	2.0
null	12.5
true	24.0

```python
# 3- Médiane du nombre d’avis par catégorie d’hôte
median_reviews_host = df.group_by("host_is_superhost").agg(
    pl.col("number_of_reviews").median().alias("median_reviews")
)
print("Médiane des avis par catégorie d'hôte :")
print(median_reviews_host)
```

```
Médiane des avis par catégorie d'hôte :
shape: (3, 2)
```

host_is_superhost	median_reviews
bool	f64
false	2.0
null	12.5
true	24.0

```python
# 4- Densité de logements par quartier
listings_per_neighbourhood = df.group_by("neighbourhood_cleansed").agg(
    pl.len().alias("num_listings")
).sort("num_listings", descending=True)

print(listings_per_neighbourhood)
```

neighbourhood_cleansed --- str	num_listings --- u32
Buttes-Montmartre	10555
Popincourt	8430
Vaugirard	7802
Batignolles-Monceau	6857
Entrepôt	6558
…	…
Élysée	2898
Hôtel-de-Ville	2821
Palais-Bourbon	2740
Luxembourg	2701
Louvre	2026

```python
# 5- Quartiers avec le plus fort taux de réservation par mois
top_neighbourhoods = df.group_by("neighbourhood_cleansed").agg(
    pl.col("reviews_per_month").mean().alias("avg_reviews_per_month")
).sort("avg_reviews_per_month", descending=True)

print("Quartiers avec le plus fort taux de réservation par mois :")
print(top_neighbourhoods)
```

```
Quartiers avec le plus fort taux de réservation par mois :
shape: (20, 2)
```

neighbourhood_cleansed	avg_reviews_per_month
str	f64
Bourse	1.577269
Louvre	1.384321
Élysée	1.320271
Temple	1.256014
Hôtel-de-Ville	1.243083
…	…
Popincourt	1.058874
Batignolles-Monceau	1.020022
Buttes-Montmartre	1.008666
Buttes-Chaumont	0.882196
Ménilmontant	0.862163

5. Connection à powerBI

```
> db.createView(
    "locations_paris_view",  // nom de la view
    "locations_paris",       // collection source
    [
      {
        $project: {
          _id: 0,
          listing_id: 1,
          listing_name: 1,
          listing_url: 1,
          source: 1,
          license: 1,
          neighbourhood: "$neighbourhood_cleansed",
          neighbourhood_group: "$neighbourhood_group_cleansed",
          latitude: 1,
          longitude: 1,
          property_type: 1,
          room_type: 1,
          accommodates: 1,
          bedrooms: 1,
```

Création de ma view

```
> show collections
< locations_paris
  locations_paris_view [view]
  system.views
```

Objectif

Permettre à **Power BI** (outil SQL) de lire des données **MongoDB** (NoSQL) **comme si c'était une base relationnelle**, via un **connecteur**.

MongoDB

Installation – Driver ODBC MongoDB

traducteur Power BI ↔ BI Connector

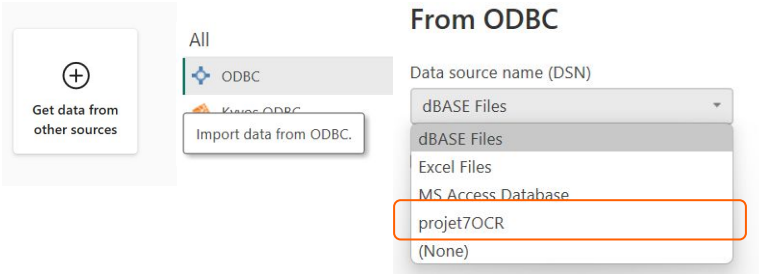
PowerBI

Installation – MongoDB BI Connector (mongosqld)

serveur SQL - traduit le noSQL
mongosqld.exe doit tourner en arrière-plan pendant l'utilisation de Power BI

```
cd "C:\Program Files\MongoDB\Connector for BI\2.14\bin"
.\mongosqld.exe
```

Paramétrage PowerBI



On peut aussi télécharger en CSV mais ce ne sera pas connecter à MongoDB

Tableau de bord

Location de courte durée à Paris

First price

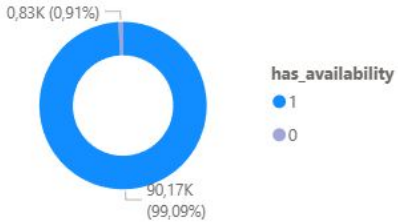
129,44

Average of availability_365

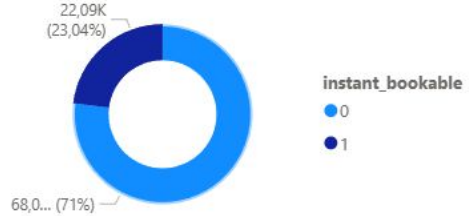
$999.00

Last price

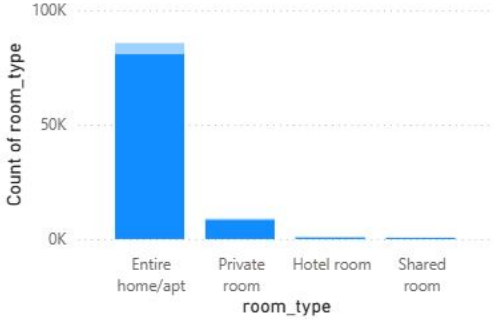
Count of has_availability by has_availability



Count of instant_bookable by instant_bookable



Count of room_type by room_type



Count of room_type by neighbourhood

Partie 3

6. Schéma d’architecture

Schéma d'architecture

Scraping des données

Insertion

Stockage des données

- Shard 1 : Données Paris
  - Primary → Serveur à Paris
  - Secondary → Serveur à Paris (local)
  - Secondary → Serveur à Lyon (distant)
- Shard 2 : Données Lyon
  - Primary → Serveur à Lyon
  - Secondary → Serveur à Lyon (local)
  - Secondary → Serveur à Paris (distant)

Connexion (via connector MongoDB ODBC Driver et PowerBI Connector)

Visualisation

7. Réplication et distribution des données

1- Intégration donnée Lyon dans la collection

```
9973 documents Lyon importés avec city = 'Lyon'
Villes présentes dans la collection : ['Lyon', 'Paris']
```

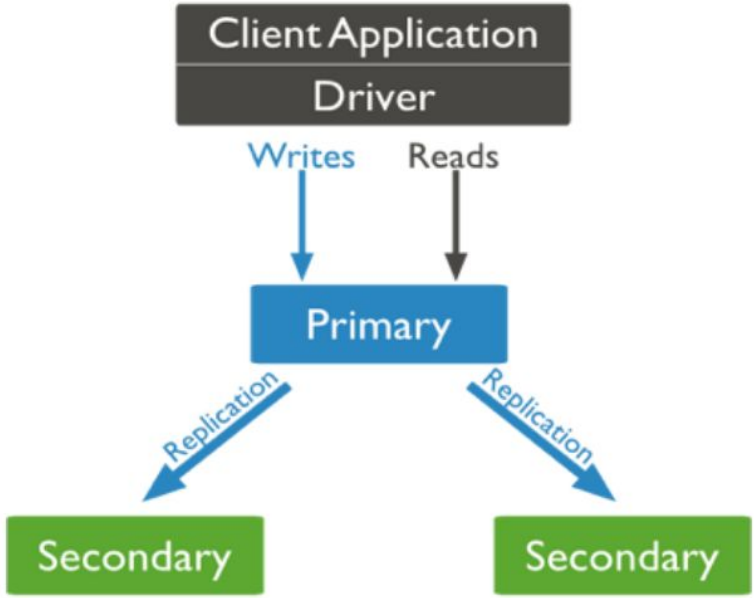
```python
import pandas as pd
from pymongo import MongoClient

# --- 1. Connexion à MongoDB ---
client = MongoClient("mongodb://localhost:27017/")
db = client.locations
collection = db["locations"]

# --- 3. Préparer les données Lyon ---
df_lyon = pd.read_csv("docs data OCR/listings_Lyon+(1).csv")
df_lyon['city']='Lyon'

# --- 4. Convertir chaque ligne en dictionnaire pour MongoDB ---
lyon_docs = df_lyon.to_dict(orient='records')

# --- 5. Importer directement dans MongoDB ---
if lyon_docs:
    collection.insert_many(lyon_docs)
    print(f"{len(lyon_docs)} documents Lyon importés avec city = 'Lyon'")

# --- >6. Vérification ---
cities = collection.distinct("city")
print("Villes présentes dans la collection :", cities)
```

2- Répliquer les données

Objectif : garantir la disponibilité des données.

Création d'un ReplicaSet : différents serveurs MongoDB qui contiennent les mêmes données pour assurer :

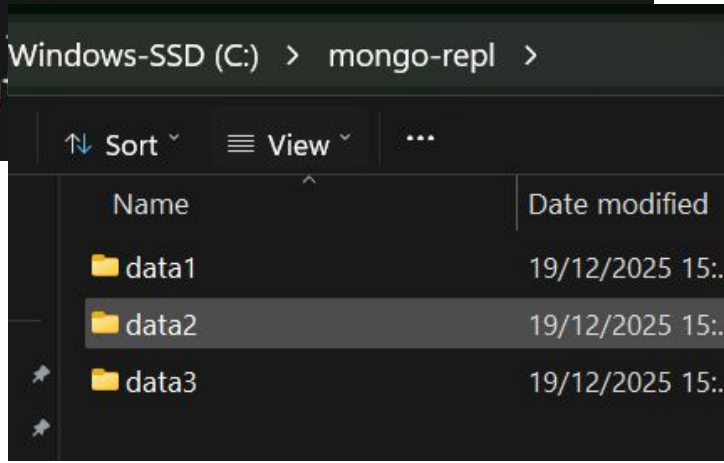
- Si le serveur principal (primary) tombe, un autre prend automatiquement la relève (secondary).
- Primary : reçoit toutes les écritures.
- Secondaries : répliquent les données du primary.



Architecture d'un ReplicaSet

ReplicaSet - Etape 1 - création plusieurs dossiers - en local pour stocker les données

```
PS C:\Users\mathi\OneDrive\Documents\8_OCR\Projet_7> mkdir C:\mongo-repl\data2


    Directory: C:\mongo-repl


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        19/12/2025     15:03                da
```



La base de données a été répliquée en local et simuler la présence de plusieurs serveurs en utilisant plusieurs ports.

3 dossiers pour les serveurs

- 1 PRIMARY (données)
- 2 SECONDARY (copie)

ReplicaSet - Etape 2 - Lancement des ReplicaSets

- Lancé trois instances de MongoDB sur des ports différents (27020, 27021, 27022) pour simuler un ReplicaSet `rs0`.

```
PS C:\Users\mathi\OneDrive\Documents\8_OCR\Projet_7> cd "C:\Program Files\MongoDB\Ser
ver\8.2\bin"
PS C:\Program Files\MongoDB\Server\8.2\bin> .\mongod.exe --replSet rs0 --port 27017 -
-dbpath "C:\mongo-repl\data1"
{"t":{"$date":"2025-12-19T15:13:58.031+01:00"},"s":"I",  "c":"-",        "id":8991200
, "ctx":"thread1","msg":"Shuffling initializers","attr":{"seed":252938700}}
{"t":{"$date":"2025-12-19T15:13:58.141+01:00"},"s":"I",  "c":"CONTROL",  "id":97374,
  "ctx":"thread1","msg":"Automatically disabling TLS 1.0 and TLS 1.1, to force-enable
 TLS 1.1 specify --sslDisabledProtocols 'TLS1_0'; to force-enable TLS 1.0 specify --s
slDisabledProtocols 'none'"}
```

Chacun dans un powershell différent

```
PS C:\Users\mathi\OneDrive\Documents\8_OCR\Projet_7> cd "C:\Program Files\MongoDB\Ser
ver\8.2\bin"
PS C:\Program Files\MongoDB\Server\8.2\bin> .\mongod.exe --replSet rs0 --port 27018 -
-dbpath "C:\mongo-repl\data2"
{"t":{"$date":"2025-12-19T15:24:04.565+01:00"},"s":"I",  "c":"-",        "id":8991200
, "ctx":"thread1","msg":"Shuffling initializers","attr":{"seed":2714388308}}
{"t":{"$date":"2025-12-19T15:24:04.648+01:00"},"s":"I",  "c":"CONTROL",  "id":97374,
  "ctx":"thread1","msg":"Automatically disabling TLS 1.0 and TLS 1.1, to force-enable
```

ReplicaSet - Etape 3 Définition des rôles des serveurs (primary - secondary)

- Avec rs.initiate() dans mongosh.
- Résultat : MongoDB commence la réplication automatique des données entre les membres.

```
locations> rs.initiate({
...   _id: "rs0",
...   members: [
...     { _id: 0, host: "localhost:27020" },  // PRIMARY
...     { _id: 1, host: "localhost:27021" },  // SECONDARY 1 (Lyon)
...     { _id: 2, host: "localhost:27022" }   // SECONDARY 2 (Paris secondaire)
...   ]
... })
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1766155370, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1766155370, i: 1 })
}
```

ReplicaSet - Etape 5 - vérification des données dans les serveurs secondary

```
rs0 [direct: secondary] locations> db.locations.distinct("city")
[ 'Lyon', 'Paris' ]
rs0 [direct: secondary] locations> db.locations.countDocuments()
105858
```

On retrouve bien l'ensemble de notre collection

Etape 6 - Sharing : Distribution des données

Objectif : Répartir les données sur les serveurs pour que

- **Les requêtes sur Paris soient rapides sur le shard Paris**
- **Les requêtes sur Lyon soient rapides sur le shard Lyon**

"Paris" → shard 1

"Lyon" → shard 2

Chaque shard est un ReplicaSet (pour réplication + disponibilité)

Création de la shardkey

Ce sera “City”

```
[direct: mongos] locations> db.locations.createIndex({ City: 1 })
City_1
[direct: mongos] locations> db.locations.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { City: 1 }, name: 'City_1' }
]
```

Splitter les données

```
direct: mongos] locations> sh.splitAt("locations.locations", { City: "Paris" })

 ok: 1,
 '$clusterTime': {
   clusterTime: Timestamp({ t: 1766163836, i: 6 }),
   signature: {
     hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
```

Répartition des sharks

```
[direct: mongos] locations> sh.splitAt("locations.locations", { city: "Lyon" })
... sh.splitAt("locations.locations", { city: "Lyon\0" })
... sh.splitAt("locations.locations", { city: "Paris" })
... sh.splitAt("locations.locations", { city: "Paris\0" })
{
  ok: 1,
```

```
[direct: mongos] locations> sh.moveChunk("locations.locations", { City: "Lyon" }, "rs
0")
{
  millis: 19,
  ok: 1,
```

Vérification
rs0 = paris
rs1 = Lyon

```
[direct: mongos] locations> db.locations.aggregate([
...     { $collStats: { storageStats: {} } },
...     { $project: { shard: 1, count: "$storageStats.count" } }
... ])
[ { shard: 'rs0', count: 105858 }, { shard: 'rs1', count: 9973 } ]
[direct: mongos] locations>
```

8. Logigramme – processus de construction de l’architecture de BDD

Logigramme

Réplication continue
(redondance assurée)
↓
Accès parallèle
(distribution des requêtes)
↓
Système opérationnel

Analyse des besoins
(volume, disponibilité, sensibilité)
↓
◇ Volume important? ◇
↓ OUI
◇ Haute disponibilité? ◇
↓ OUI
Choisir Architecture Sharding + ReplicaSet
↓
Déployer Servers (ReplicaSet)
↓
Définir Shard Key (basée sur les requêtes)
↓
Déployer Shards (chaque shard = ReplicaSet)
↓
Initialiser ReplicaSets (élection Primary)
↓
. Activer le Sharding via Mongos

Suite projet

Projet complet :

se poser également les questions suivantes :

◇ *Données sensibles?* ◇

↓ *OUI*

Planifier sécurité renforcée

(Chiffrement + Authentification)

Tests de charge et failover

↓

◇ Performance OK? ◇

↓ OUI

Déploiement production

↓

Configurer monitoring (métriques + alertes)