Compte Rendu Lab 4

Initiation à MongoDB

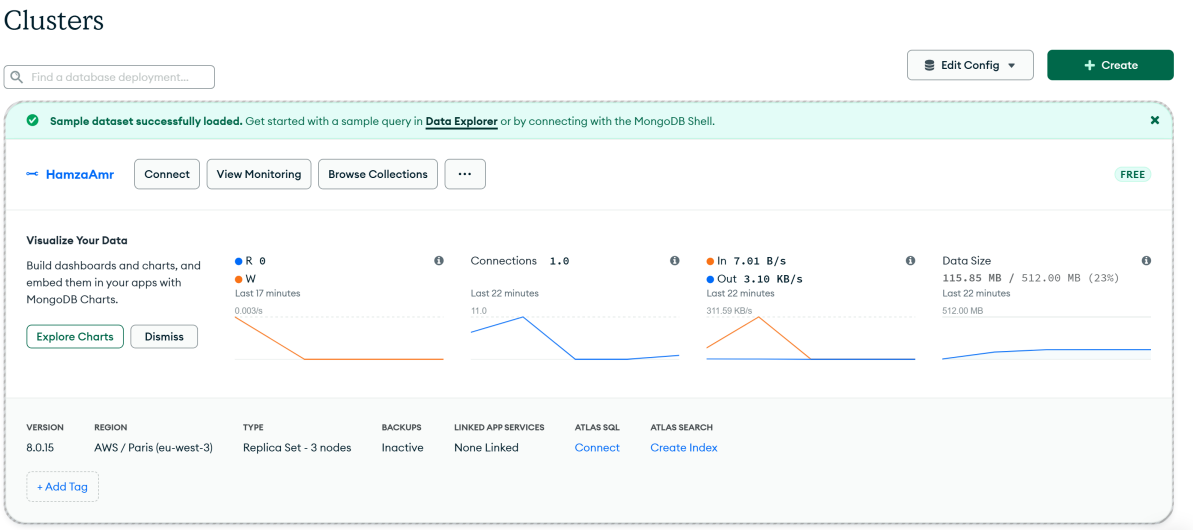
Nom et Prénom : AMRANI Hamza

Filière : IDF

I. Création d'un Compte et Déploiement d'un Cluster

1. Inscription et création du cluster

1. Accéder à MongoDB Atlas : https://www.mongodb.com/atlas
2. S'inscrire ou se connecter avec un compte Gmail
3. Cliquer sur "Créer un cluster" et sélectionner un cluster gratuit (M0 Sandbox)
4. Choisir un fournisseur de cloud et une région
5. Configurer le nom du cluster
6. Cliquer sur "Créer un cluster"
7. Créer un utilisateur avec un mot de passe sécurisé



II. Connexion à MongoDB Atlas

1. Obtention de l'URI de Connexion

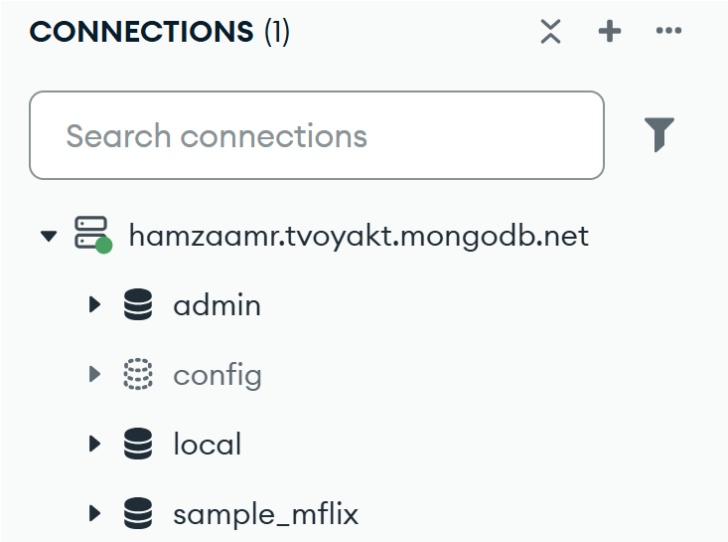
Une fois le cluster créé Copier l'URI de connexion

Exemple d'URI de connexion : en Remplace `<db_password>` par le mot de passe réel avant de vous connecter.

```
mongodb+srv://username:<db_password>@cluster0.apxc2.mongodb.net/
```

2. Connexion via MongoDB Compass

En Utilise l'URI de connexion pour établir la connexion avec MongoDB Compass.



III. Création d'une Base de Données et d'une Collection

1. Création de la Base et de la Collection

Créer (sélectionner) la base de données :

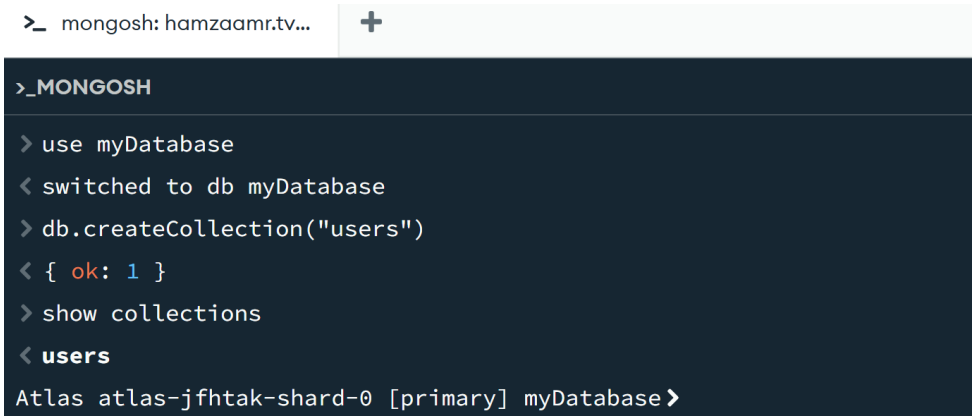
```
use myDatabase
```

Créer une collection nommée "users" :

```
db.createCollection("users")
```

Lister toutes les collections présentes dans la base de données active :

```
show collections
```



IV. Insertion de Documents

1. Insertion d'un Document Unique et de Plusieurs Documents

Insérer un document JSON dans la collection `users` :

```
db.users.insertOne({
    name: "Mohamed",
    age: 20,
    city: "Casablanca"
})
```

Insérer plusieurs documents simultanément :

```
db.users.insertMany([
    { name: "Alice", age: 25, city: "Paris" },
    { name: "Bob", age: 30, city: "Lyon" },
    { name: "Charlie", age: 28, city: "Marseille" }
])
```

2. Insertion Personnalisée

Ajouter un document avec des informations personnalisées :

```
db.users.insertOne({
    name: "AMRANI",
    age: 23,
    city: "Oujda"
})
```

```
> db.users.insertOne({
      name: "AMRANI",
      age: 23,
      city: "Oujda"
  })
< {
    acknowledged: true,
    insertedId: ObjectId('68fc16b99c22566edee5f254')
  }
Atlas atlas-jfhtak-shard-0 [primary] myDatabase>
```

3. Affichage des Documents

Afficher tous les documents de la collection :

```
db.users.find()
```

```
  {
    _id: ObjectId('68fc16b99c22566edee5f254'),
    name: 'AMRANI',
    age: 23,
    city: 'Oujda'
  }
Atlas atlas-jfhtak-shard-0 [primary] myDatabase>
```

V. Requêtes de Récupération

1. Compter, Rechercher et Filtrer les Documents

Compter le nombre total de documents :

```
db.users.count()
```

N.B :

`DeprecationWarning`: Collection.count() est obsolète. Utilisez `countDocuments`.

Méthode recommandée :

```
db.users.countDocuments()
```

Rechercher un utilisateur spécifique :

```
db.users.findOne({ name: "Alice" })
```

Filtrer les utilisateurs par âge (age $>$ 20) :

```
db.users.find({ age: { $gt: 20 } })
```

VI. Mise à Jour et Suppression

1. Modifier un Document

Modifier l'âge d'Alice de 25 à 26 ans :

```
db.users.updateOne(
    { name: "Alice" },
    { $set: { age: 26 } }
)
```

2. Supprimer un Document et une Collection

Supprimer un utilisateur :

```
db.users.deleteOne({ name: "Charlie" })
```

Supprimer entièrement la collection :

```
db.users.drop()
```

```
> db.users.deleteOne({ name: "Charlie" })
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.users.drop()
< true
```

VII. Importation de Fichiers dans une Collection

1. Création de la Base et de la Collection

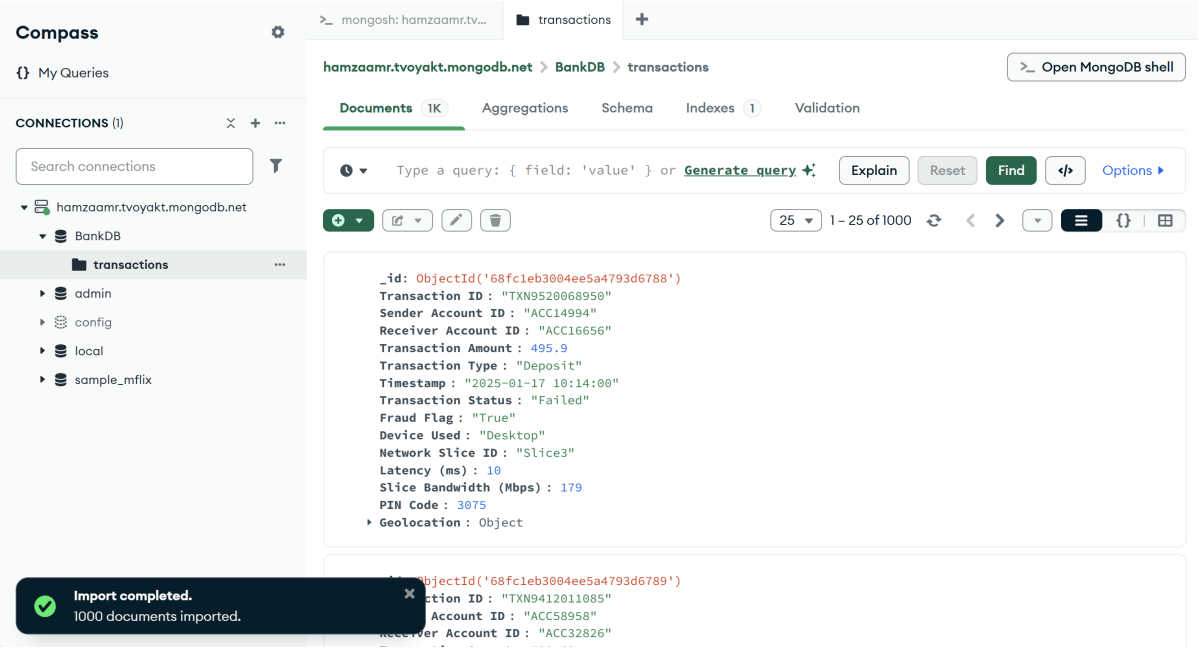
Créer la base de données `BankDB` et la collection `transactions` :

```
use BankDB
db.createCollection("transactions")
```

2. Importation du Fichier

Méthode : Interface Graphique

Utiliser l'interface MongoDB Compass pour importer le fichier `transactions.json` .



VIII. Requêtes d'Analyse

1. Requêtes de Base

a) Compter les Transactions

Nombre total de transactions dans la base :

```
db.transactions.count()
```

```
>_MONGOSH
> db.transactions.count()
< 1000
Atlas atlas-jfhtak-shard-0 [primary] BankDB>
```

b) Afficher une Transaction

Examiner la structure d'une transaction :

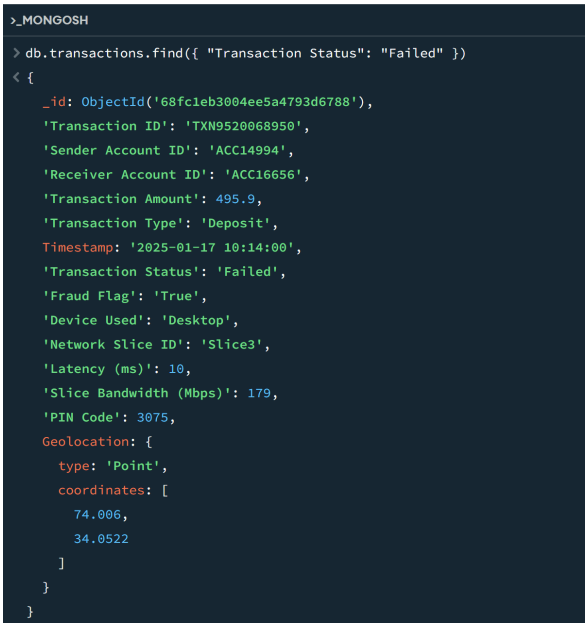
```
db.transactions.findOne()
```

```
> db.transactions.findOne()
< {
    _id: ObjectId('68fc1eb3004ee5a4793d6788'),
    'Transaction ID': 'TXN9520068950',
    'Sender Account ID': 'ACC14994',
    'Receiver Account ID': 'ACC16656',
    'Transaction Amount': 495.9,
    'Transaction Type': 'Deposit',
    Timestamp: '2025-01-17 10:14:00',
    'Transaction Status': 'Failed',
    'Fraud Flag': 'True',
    'Device Used': 'Desktop',
    'Network Slice ID': 'Slice3',
    'Latency (ms)': 10,
    'Slice Bandwidth (Mbps)': 179,
    'PIN Code': 3075,
    Geolocation: {
      type: 'Point',
      coordinates: [
        74.006,
        34.0522
      ]
    }
  }
Atlas atlas-jfhtak-shard-0 [primary] BankDB>
```

c) Transactions Échouées

Trouver toutes les transactions avec un statut "Failed" :

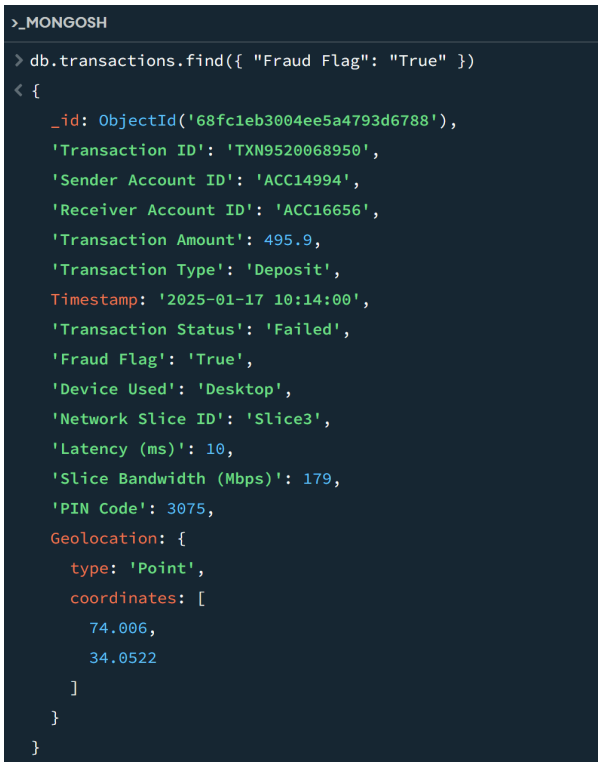
```
db.transactions.find({ "Transaction Status": "Failed" })
```



d) Transactions Suspectes

Vérifier les transactions avec indicateur de fraude :

```
db.transactions.find({ "Fraud Flag": "True" })
```



2. Requêtes d'Agrégation

a) Transactions par Statut

Nombre total de transactions groupées par statut :

```
>_MONGOSH
> db.transactions.aggregate([
      { $group: {
          _id: "$Transaction Status",
          total: { $count: {} }
      } }
  ])
< {
    _id: 'Success',
    total: 487
  }
  {
    _id: 'Failed',
    total: 513
  }
Atlas atlas-jfhtak-shard-0 [primary] BankDB>
```

b) Montant Moyen par Type

Calculer le montant moyen des transactions par type :

```
>_MONGOSH
> db.transactions.aggregate([
      { $group: {
          _id: "$Transaction Type",
          avgAmount: { $avg: "$Transaction Amount" }
      } }
  ])
< {
    _id: 'Withdrawal',
    avgAmount: 733.3745806451612
  }
  {
    _id: 'Transfer',
    avgAmount: 780.1512032085561
  }
  {
    _id: 'Deposit',
    avgAmount: 797.6032278481013
  }
```

c) Top 5 des Comptes Émetteurs

Identifier les 5 comptes qui envoient le plus d'argent :

```
db.transactions.aggregate([
    { $group: {
        _id: "$Sender Account ID",
        totalSent: { $sum: "$Transaction Amount" }
    } },
    { $sort: { totalSent: -1 } },
    { $limit: 5 }
])
```

d) Montant Total par Type

Calculer le montant total des transactions par type :

```
db.transactions.aggregate([
    { $group: {
        _id: "$Transaction Type",
        total: { $sum: "$Transaction Amount" }
    } }
])
```

```
>_MONGOSH
> db.transactions.aggregate([
      { $group: {
          _id: "$Transaction Type",
          total: { $sum: "$Transaction Amount" }
      } }
  ])
< {
    _id: 'Withdrawal',
    total: 227346.12
  }
  {
    _id: 'Deposit',
    total: 252042.62
  }
  {
    _id: 'Transfer',
    total: 291776.55
  }
```

e) Latence Moyenne par Tranche Réseau

Analyser la latence moyenne par "Network Slice ID" :

```
db.transactions.aggregate([
    { $group: {
        _id: "$Network Slice ID",
        avgLatency: { $avg: "$Latency (ms)" }
    } }
])
```

```
> db.transactions.aggregate([
    { $group: {
        _id: "$Network Slice ID",
        avgLatency: { $avg: "$Latency (ms)" }
    } }
])
< {
    _id: 'Slice3',
    avgLatency: 11.86053412462908
  }
  {
    _id: 'Slice1',
    avgLatency: 11.91640866873065
  }
  {
    _id: 'Slice2',
    avgLatency: 11.3
  }
```

f) Échecs par Appareil

Transactions échouées groupées par appareil utilisé :

```
db.transactions.aggregate([
    { $match: { "Transaction Status": "Failed" } },
    { $group: {
        _id: "$Device Used",
        totalFailed: { $count: {} }
    } }
])
```

```
>_MONGOSH
> db.transactions.aggregate([
    { $match: { "Transaction Status": "Failed" } },
    { $group: {
        _id: "$Device Used",
        totalFailed: { $count: {} }
    } }
])
< {
    _id: 'Mobile',
    totalFailed: 270
  }
  {
    _id: 'Desktop',
    totalFailed: 243
  }
```

3. Requêtes Complexes

a) Fraudes avec Montant Élevé

Trouver les transactions suspectes avec un montant supérieur à 1000 :

```
db.transactions.find({
    "Fraud Flag": "True",
    "Transaction Amount": { $gt: 1000 }
})
```

>_MONGOSH

```
> db.transactions.find({
    "Fraud Flag": "True",
    "Transaction Amount": { $gt: 1000 }
  })
< {
    _id: ObjectId('68fc1eb3004ee5a4793d678b'),
    'Transaction ID': 'TXN2214150284',
    'Sender Account ID': 'ACC48650',
    'Receiver Account ID': 'ACC76457',
    'Transaction Amount': 1129.88,
    'Transaction Type': 'Transfer',
    Timestamp: '2025-01-17 10:56:00',
    'Transaction Status': 'Success',
    'Fraud Flag': 'True',
    'Device Used': 'Mobile',
    'Network Slice ID': 'Slice3',
    'Latency (ms)': 10,
    'Slice Bandwidth (Mbps)': 127,
    'PIN Code': 6374,
    Geolocation: {
      type: 'Point',
      coordinates: [
        74.006,
        34.0522
      ]
    }
  }
```

b) Transactions dans une Fourchette

Compter les transactions ayant un montant entre 100 et 200 :

```
db.transactions.count({
    "Transaction Amount": {
        $gte: 100,
        $lte: 200
    }
})
```

```
> db.transactions.count({
      "Transaction Amount": {
          $gte: 100,
          $lte: 200
      }
  })
< 66
Atlas atlas-jfhtak-shard-0 [primary] BankDB>
```

IX. Indicateurs Clés de Performance (KPI)

1. Montant Total des Transactions

Nombre Total de Transactions :

```
db.transactions.count()
```

Montant Total des Transactions

```
db.transactions.aggregate([
    { $group: {
        _id: null,
        totalAmount: { $sum: "$Transaction Amount" }
    } }
])
```

```
>_MONGOSH
> db.transactions.count()
< 1000
> db.transactions.aggregate([
      { $group: {
          _id: null,
          totalAmount: { $sum: "$Transaction Amount" }
      } }
  ])
< {
    _id: null,
    totalAmount: 771165.29
  }
```

2. Taux d'Échec des Transactions

Calculer le pourcentage de transactions échouées :

```
db.transactions.aggregate([
    { $group: {
        _id: "$Transaction Status",
        count: { $count: {} }
    } }
])
```

3. Nombre de Transactions Frauduleuses

```
db.transactions.countDocuments({ "Fraud Flag": "True" })
```

X. Conclusion

Ce premier volet du **Lab 4** a permis d'acquérir une maîtrise des bases de **MongoDB** à travers quatre axes essentiels :

- **Déploiement et connexion :** création d'un cluster sur MongoDB Atlas et connexion via MongoDB Compass.
- **Opérations CRUD :** manipulation flexible des documents (insertion, lecture, mise à jour, suppression).
- **Requêtes avancées :** utilisation d'opérateurs et de filtres complexes pour extraire des données précises.
- **Agrégation et analyse :** exploitation des pipelines d'agrégation pour calculer des indicateurs clés et détecter des anomalies.