**Projet : Migration et Centralisation de Données Météo**

**1️⃣ Contexte**

* Objectif : Migrer des données météorologiques depuis plusieurs sources (CSV et Airbyte) vers une base MongoDB centralisée.
* Besoin : Automatiser la collecte, la transformation et le chargement (ETL) des données tout en assurant la qualité et la cohérence.

**2️⃣ Sources de données**

* **Fichiers CSV** provenant de :
  + Donnees\_JSON
  + Google Drive (Ichtegem et Madeleine)
* **Formats** : CSV standard et JSON Airbyte encapsulé.
* **Volume** : Plusieurs milliers de lignes par source.

**3️⃣ Architecture logicielle**

* **Script Python (Migration\_code.py)** :
  + Lecture des fichiers CSV depuis AWS S3
  + Normalisation des données JSON/Airbyte
  + Nettoyage (suppression des lignes avec valeurs manquantes)
  + Insertion dans MongoDB
  + Contrôle qualité des données insérées
* **.env** : Centralisation des variables sensibles (AWS et MongoDB).
* **MongoDB local** pour tests, puis conteneurisation avec Docker.

**4️⃣ Technologies utilisées**

| **Categorie** | **Technologie** |
| --- | --- |
| Langage | Python 3.12 |
| Base de données | MongoDB 8 |
| Stockage Cloud | AWS S3 |
| Conteneurisation | Docker + Docker Compose |
| Cloud Deployment | AWS ECS + ECR |
| Gestion des secrets | .env / variables d’environnement |
| Librarie Python | pandas, pymongo, boto3, python-dotenv |

**5️⃣ Workflow ETL**

1. Charger les variables d’environnement (.env).
2. Connexion au bucket S3 → récupération des fichiers CSV.
3. Lecture des fichiers CSV → parsing JSON si nécessaire.
4. Nettoyage des données → suppression des lignes avec valeurs manquantes.
5. Connexion à MongoDB → insertion des données par collection.
6. Contrôle qualité → vérification du nombre de lignes, des colonnes et du taux de valeurs manquantes.

**6️⃣ Conteneurisation**

* Docker Compose définit deux services :
  1. mongodb\_server → MongoDB conteneurisé avec volume persistant.
  2. migration\_container → script Python pour la migration.
* Avantages :
  1. Isolation des services
  2. Reproductibilité
  3. Déploiement simplifié sur cloud

**7️⃣ Déploiement sur AWS**

1. **ECR** : Création du registre et push de l’image Docker de l’application.
2. **ECS** : Déploiement d’une tâche ECS avec l’image Docker.
3. **CloudWatch** : Gestion des logs et monitoring des tâches ECS.
4. Résultat attendu : pipeline ETL automatisé exécuté dans le cloud avec MongoDB comme base centrale.

**8️⃣ Résultats**

* Base Meteo\_data\_db créée avec **3 collections** correspondant aux sources CSV.
* Données chargées et validées avec contrôle qualité automatique.
* Script et conteneur Docker prêts pour le déploiement cloud.

**9️⃣ Prochaines étapes**

* Sécurisation de MongoDB (activation de l’authentification et des rôles).
* Automatisation complète sur ECS avec planification cron.
* Intégration d’un dashboard pour visualiser les données migrées.