Project Guide: An Automated Pipeline with Spark & MongoDB

1. Project Overview & Architecture

We will build a fully automated, cloud-based data pipeline. A GitHub Actions workflow will run on a schedule to extract 311 data and load it into a raw collection in MongoDB Atlas. A Spark job, also running in GitHub Actions, will then transform that raw data and save it to a clean, aggregated collection. Your Tableau dashboard will have a live connection to this final collection, providing a fast, near real-time view of the data.

The Architecture:

KC 311 API ⟶ Python Script (on GitHub Actions) ⟶ MongoDB Atlas (Raw Data) ⟶ PySpark Job (on GitHub Actions) ⟶ MongoDB Atlas (Aggregated Data) ⟶ Tableau Desktop

---

2. The Foundation: Setting Up Your Tools

Step A: Configure your MongoDB Atlas Cloud Database (15 mins)

This will be our central data warehouse in the cloud.

1. **Sign Up:** Go to MongoDB Atlas and create a free account.
2. **Create a Free Cluster:** Follow the prompts to deploy a new **M0 Sandbox** cluster. This is their "free forever" tier. You can choose any cloud provider and region.
3. **Create a Database User:** In the "Database Access" section, create a new user with a secure password. You will use these credentials in your scripts.
4. **Configure Network Access:** This is a crucial step. To allow your GitHub Actions workflow to connect, you must allow access from anywhere.
    - Go to the "Network Access" section.
    - Click "Add IP Address".
    - Select **"Allow Access from Anywhere"**. This will enter `0.0.0.0/0` in the IP address field.
    - Confirm the entry.
5. **Get Your Connection String:**
    - Go back to your Database "Overview" and click the "Connect" button.
    - Select "Drivers".
    - It will show you a **Connection String**. Copy this string and save it. It will look something like `mongodb+srv://<username>:<password>@cluster-name.mongodb.net/`. You will replace `<password>` with the password you created.

Step B: Configure Your Local Tools

You'll need these for writing and testing your code before automating it.

1. **Install Spark:** Use Homebrew in your terminal.

   Bash

   ```
   brew install apache-spark
   ```
2. **Create Project & Virtual Environment:**

   Bash

   ```
   mkdir kc_311_mongo_project
   cd kc_311_mongo_project
   python3 -m venv venv
   source venv/bin/activate
   ```
3. **Install Python Libraries:**

   Bash

   ```
   pip install pandas sodapy pymongo
   ```

---

3. The Pipeline Code

Create the following two Python scripts in your project folder.

Script 1: ingest_data.py

This script fetches the data and loads it into a raw_requests collection in MongoDB.

Python

```
# ingest_data.py
import os
from sodapy import Socrata
from pymongo import MongoClient

# --- Configuration ---
SOCRATA_DOMAIN = "data.kcmo.org"
SOCRATA_DATASET_ID = "d4px-6rwg"
# Your connection string will be securely passed by GitHub Actions
MONGO_CONNECTION_STRING = os.environ.get("MONGO_CONNECTION_STRING")
DB_NAME = "kc_311_db"
COLLECTION_NAME = "raw_requests"

def ingest_data():
    if not MONGO_CONNECTION_STRING:
        raise ValueError("MONGO_CONNECTION_STRING environment variable not set!")

    print("Connecting to MongoDB Atlas...")
    client = MongoClient(MONGO_CONNECTION_STRING)
    db = client[DB_NAME]
    collection = db[COLLECTION_NAME]

    print("Fetching data from Socrata API...")
```

```
    socrata_client = Socrata(SOCRATA_DOMAIN, None)
    results = socrata_client.get(SOCRATA_DATASET_ID, limit=10000)

    if not results:
        print("No data fetched. Exiting.")
        return

    print(f"Fetched {len(results)} records. Deleting old raw data...")
    # Clear the collection before inserting new data
    collection.delete_many({})

    print(f"Inserting {len(results)} new records into '{COLLECTION_NAME}'...")
    collection.insert_many(results)

    print("Ingestion complete.")
    client.close()

if __name__ == "__main__":
    ingest_data()
```

Script 2: transform_data.py

This is your PySpark job. It reads from the raw collection, transforms the data, and saves it to a clean mart_daily_summary collection.

Python

```
# transform_data.py
import os
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, date_trunc, count

# --- Configuration ---
MONGO_CONNECTION_STRING = os.environ.get("MONGO_CONNECTION_STRING")
DB_NAME = "kc_311_db"
RAW_COLLECTION = "raw_requests"
MART_COLLECTION = "mart_daily_summary"

def main():
    if not MONGO_CONNECTION_STRING:
        raise ValueError("MONGO_CONNECTION_STRING environment variable not set!")

    spark = SparkSession.builder \
        .appName("KC311MongoTransformation") \
        .config("spark.mongodb.input.uri",
f"{MONGO_CONNECTION_STRING}.{DB_NAME}.{RAW_COLLECTION}") \
        .config("spark.mongodb.output.uri",
f"{MONGO_CONNECTION_STRING}.{DB_NAME}.{MART_COLLECTION}") \
        .getOrCreate()

    print("Reading raw data from MongoDB into Spark...")
    df = spark.read.format("mongodb").load()

    print("Transforming data...")
    mart_df = df.select(
        col("creation_date").cast("timestamp"),
        col("category"),
```

```
        col("status")
    ).filter(col("category").isNotNull()) \
     .withColumn("request_date", date_trunc("day", col("creation_date"))) \
     .groupBy("request_date", "category", "status") \
     .agg(count("*").alias("number_of_requests"))

    print(f"Writing transformed data to '{MART_COLLECTION}' collection...")
    mart_df.write.format("mongodb").mode("overwrite").save()

    print("Transformation complete.")
    spark.stop()

if __name__ == "__main__":
    main()
```

---

4. Automation with GitHub Actions

This is how your project will run itself.

1. **Create a GitHub Repository:** Go to GitHub, create a new public repository, and push your two Python scripts to it.
2. **Add Your Connection String as a Secret:**
    - In your GitHub repo, go to "Settings" > "Secrets and variables" > "Actions".
    - Click "New repository secret".
    - Name the secret `MONGO_CONNECTION_STRING`.
    - Paste your full MongoDB Atlas connection string (with your password included) as the value.
3. **Create the Workflow File:** In your project folder, create the directories `.github/workflows/`. Inside that, create a file named `pipeline.yml`.

    YAML

    ```yaml
    # .github/workflows/pipeline.yml
    name: Run Hourly 311 Data Pipeline

    on:
      schedule:
        # Runs at minute 30 of every hour
        - cron: '30 * * * *'
      workflow_dispatch: # Allows you to run it manually

    jobs:
      run-pipeline:
        runs-on: ubuntu-latest
        steps:
          - name: Checkout repository code
            uses: actions/checkout@v4

          - name: Set up Python 3.9
            uses: actions/setup-python@v4
            with:
              python-version: 3.9
    ```

```
    - name: Set up Java for Spark
      uses: actions/setup-java@v4
      with:
        distribution: 'temurin'
        java-version: '11'

    - name: Install Python dependencies
      run: pip install pandas sodapy pymongo

    - name: Run Ingestion Script
      env:
        MONGO_CONNECTION_STRING: ${{ secrets.MONGO_CONNECTION_STRING }}
      run: python ingest_data.py

    - name: Run Spark Transformation Script
      env:
        MONGO_CONNECTION_STRING: ${{ secrets.MONGO_CONNECTION_STRING }}
      run: |
        # Download and set up Spark
        wget https://archive.apache.org/dist/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz
        tar -xvzf spark-3.5.0-bin-hadoop3.tgz
        # Run the Spark job, providing the MongoDB connector package
        spark-3.5.0-bin-hadoop3/bin/spark-submit \
          --packages org.mongodb.spark:mongo-spark-connector_2.12:10.2.1 \
          transform_data.py
```

Commit and push this YAML file. Your pipeline is now fully automated! It will run every hour in the cloud.

---

5. Visualization with Tableau

1. **Install the MongoDB BI Connector:** Tableau needs this to talk to MongoDB. Follow the official instructions on Tableau's website for your operating system.
2. **Connect Tableau to MongoDB Atlas:**
    - Open Tableau Desktop.
    - Under "Connect", find and select **MongoDB BI Connector**.
    - Enter the connection details from your Atlas cluster (server host, username, password).
    - For "Database," enter `kc_311_db`.
3. **Build Your Dashboard:**
    - Tableau will now show your MongoDB collections as if they are SQL tables.
    - Drag your clean `mart_daily_summary` collection onto the canvas.
    - Go to a new Worksheet and build your charts. Your dashboard will be fast because it's only querying the clean, aggregated data.
    - Use the "Refresh" button in Tableau to pull the latest data from Atlas whenever you open the dashboard.

You now have a complete, professional, and automated data pipeline that is guaranteed to impress.