# **📦 Project: big-data-infrastructure-demo**

## **🎯 Purpose**

This repository is designed to **demonstrate the skills and concepts** required for the role of a **Big Data Infrastructure Engineer** — using a realistic, modular, and containerized setup. It simulates a **real-time health data pipeline**, transforming clinical records into queryable datasets using tools from the modern data engineering stack.

## **🌍 Environment Description**

This project mimics a healthcare environment where patient records from an **OpenMRS database** (via MySQL) are captured in real time and streamed through a **big data infrastructure** for storage, querying, and analysis.

Key goals:

* Show proficiency with **Hadoop ecosystem components** (HDFS, Hive, YARN)
* Demonstrate **Kafka-based real-time ingestion**
* Use **Debezium** for CDC (Change Data Capture)
* Enable **SQL querying on big data** using Hive
* Optional: Integrate **Prometheus and Grafana** for system monitoring
* Containerized setup using **Docker Compose**
* Deployable and testable locally or in the cloud (e.g., DigitalOcean, AWS)

## **📁 Repository Structure**

bash

CopyEdit

big-data-infrastructure-demo/

│

├── docker/ # Dockerfiles & config for each service

│ ├── mysql/ # MySQL DB w/ OpenMRS schema

│ ├── kafka/ # Kafka and Zookeeper setup

│ ├── debezium/ # Kafka Connect + Debezium

│ ├── hadoop/ # Hadoop (HDFS, NameNode, DataNodes)

│ ├── hive/ # Hive Metastore & HiveServer2

│ ├── prometheus/ # Prometheus config (optional)

│ ├── grafana/ # Grafana dashboards (optional)

│ └── exporters/ # JMX exporters for monitoring (optional)

│

├── compose/ # Docker Compose configurations

│ ├── stage-1-minimal.yaml

│ ├── stage-2-hdfs.yaml

│ ├── stage-3-hive.yaml

│ ├── stage-4-monitoring.yaml

│ └── stage-final.yaml

│

├── pipelines/ # Debezium + Kafka Connect configs

│ ├── debezium-openmrs-source.json

│ └── kafka-connect-hdfs-sink.json

│

├── sql/ # SQL scripts (e.g., Hive table creation)

│ └── hive-create-table.sql

│

├── data/

│ └── openmrs\_sample\_dump.sql # MySQL sample dump of OpenMRS DB

│

├── .github/

│ └── workflows/ci.yaml # GitHub Actions (optional CI/CD)

│

├── README.md # Main instructions

├── roadmap.md # Phase-by-phase project goals

└── LICENSE

## **🏗️ Final System Architecture**

scss

CopyEdit

MySQL (OpenMRS DB Dump)

↓ (via Debezium)

Kafka ←→ Kafka Connect → HDFS (Hadoop)

↓

Hive

↓

Queries via Beeline

↓

(Optional) Monitoring via Prometheus + Grafana

## **🧱 Phase-by-Phase Development Approach**

### **✅ Phase 1: Minimal Pipeline — MySQL + Debezium + Kafka**

* MySQL container with openmrs\_sample\_dump.sql
* Debezium running inside Kafka Connect container
* Kafka and Zookeeper for messaging
* Insert into DB → See CDC event in Kafka

🗂️ File: compose/stage-1-minimal.yaml

### **✅ Phase 2: Add Hadoop — Stream into HDFS**

* Deploy Hadoop (NameNode + 2 DataNodes)
* Kafka Connect uses HDFS Sink connector
* JSON/Avro events written to HDFS

🗂️ File: compose/stage-2-hdfs.yaml

### **✅ Phase 3: Add Hive — Query via SQL**

* Launch Hive Metastore + HiveServer2
* Create Hive external table over HDFS data
* Use Beeline or JDBC to query patient data

🗂️ File: compose/stage-3-hive.yaml

### **✅ Phase 4: Monitoring (Optional)**

* Prometheus scrapes metrics from Kafka, Hadoop, Debezium
* Grafana visualizes ingestion rate, disk usage, job status
* JMX exporters or Prometheus exporters installed

🗂️ File: compose/stage-4-monitoring.yaml

### **✅ Phase 5: GitHub CI/CD & Final Composition**

* GitHub Actions: validate docker-compose + lint configs
* All phases documented and runnable
* Easy-to-clone, run, and test anywhere

🗂️ File: compose/stage-final.yaml  
 🗂️ CI: .github/workflows/ci.yaml

## **🧪 Demo: How to Show the Setup Works**

💡 This is your **interview/demo script** — a live proof that the pipeline works.

### **1. Run the Full Stack**

bash

CopyEdit

docker-compose -f compose/stage-final.yaml up -d

### **2. Import OpenMRS Sample DB into MySQL**

bash

CopyEdit

docker cp ./data/openmrs\_sample\_dump.sql mysql:/openmrs.sql

docker exec -i mysql sh -c 'exec mysql -u root -p$MYSQL\_ROOT\_PASSWORD openmrs' < ./data/openmrs\_sample\_dump.sql

### **3. Manually Insert a Patient Record**

bash

CopyEdit

docker exec -it mysql mysql -u root -popenmrs

USE openmrs;

INSERT INTO patient (patient\_id, gender, birthdate, creator, date\_created)

VALUES (90001, 'F', '1987-05-12', 1, NOW());

INSERT INTO person\_name (person\_name\_id, person\_id, given\_name, family\_name, creator, date\_created)

VALUES (80001, 90001, 'Amina', 'Tshisekedi', 1, NOW());

### **4. Verify Kafka Received the Event**

bash

CopyEdit

docker exec -it kafka-broker kafka-console-consumer \

--bootstrap-server localhost:9092 \

--topic dbserver1.openmrs.patient \

--from-beginning

✅ You should see a JSON message containing the new patient.

### **5. Confirm Data Landed in HDFS**

bash

CopyEdit

docker exec -it hadoop-namenode hdfs dfs -ls /kafka/openmrs.patient/

✅ .json or .avro files appear in HDFS based on Kafka sink connector.

### **6. Query Data in Hive**

bash

CopyEdit

docker exec -it hive-server beeline -u jdbc:hive2://localhost:10000

-- Inside Beeline:

SELECT payload.patient\_id, payload.gender FROM patients;

✅ You should see the inserted patient info.

### **7. (Optional) Visualize in Grafana**

* Open: http://localhost:3000
* Default login: admin / admin
* Explore dashboards: Kafka, Hadoop, Connect