Steps 3
4: Seeding Sample Database and Health-Checking Containers

Feature-Store Thesis Documentation

May 5, 2025

Overview

This document covers:

1. **Step 3**: Seed the Debezium sample database (`inventory`).
2. **Step 4**: Verify all Docker Compose services are healthy.

1 Step 3: Seeding the Sample Database

Debezium's tutorial provides an `inventory.sql` script to create tables and sample data.

3.1 Download the SQL Script

```
curl -LO \
  https://raw.githubusercontent.com/debezium/debezium-examples/main/tutorial/
      inventory.sql
```

3.2 Execute the Script in the MySQL Container

Replace <mysql-container> with your container name.

```
docker exec -i <mysql-container> \
  mysql -uroot -p"$MYSQL_ROOT_PASSWORD" < inventory.sql
```

This command:

- Connects to the MySQL client inside the container.
- Reads and executes DDL & DML to create the 'inventory' schema, tables, and demo rows.

Ensure no errors are printed; all tables ('customers', 'products', 'orders', etc.) should exist.

2 Step 4: Health-Checking Containers

Verify that all services are running and ready.

4.1 List Service Status

```
docker compose ps
```

4.2 Expected Output Example

Name	Command	State	Ports
configs-zookeeper-1	/docker-entrypoint.sh zookeeper	Up	2181/tcp, 2888/tcp, 3888/tcp
configs-kafka-1	/docker-entrypoint.sh kafka	Up	9092/tcp
configs-mysql-1	mysqld	Up	3306/tcp, 33060/tcp
configs-connect-1	start-connect	Up	8083/tcp

4.3 Troubleshooting

- If any service is not ‘Up‘, inspect its logs:

  ```
  docker compose logs <service-name>
  ```

- Common issues:
  - Port conflicts — ensure no other processes bind the same ports.
  - Configuration errors — review env vars and mounted files (e.g., ‘my.cnf‘).