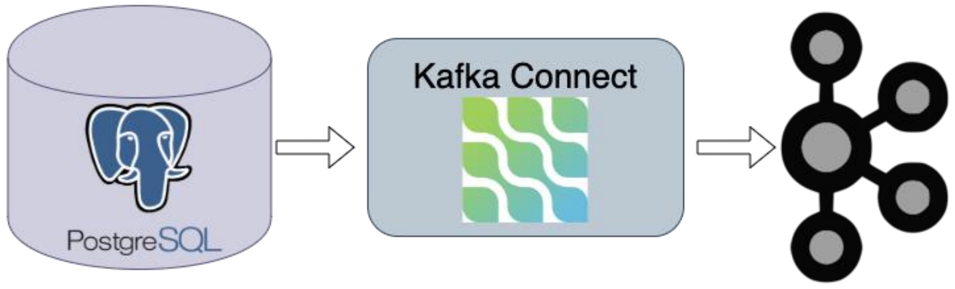
Debezium, Kafka, and Postgres POC Documentation



Introduction

This proof of concept (POC) demonstrates the use of **Debezium** for **Change Data Capture (CDC)** in a PostgreSQL database, capturing changes triggered by **CREATE**, **UPDATE**, or **DELETE** operations. These changes are then streamed into **Kafka** and consumed by the **baas-charges-module** (demo project) after being serialized by the **baas-analytical-module** (demo project). The consumed data is then deserialized for further processing.

This POC leverages **Docker Compose** to deploy and manage containers for each component, enabling an efficient setup and interaction between the CDC, Kafka, and data processing modules.

Components Overview

1. Postgres

- Acts as the source database where data changes are captured.
- Configured with CDC (Change Data Capture) to capture row-level changes in tables.

2. Debezium

- An open-source CDC tool that tracks changes in databases and streams those changes to Kafka.
- Listens to the Postgres instance and streams any changes (inserts, updates, deletes) to Kafka topics.

3. Kafka

- Serves as the messaging backbone, where all data changes are streamed to different topics for downstream consumption.
- Kafka is configured with Zookeeper for managing its cluster metadata.

4. Zookeeper

- Required by Kafka to manage and coordinate distributed instances of Kafka.

5. Application for triggering request to the DB

- Required for CDC.

6. Consumer/Listner application

- Required for processing the messages from the Kafka.

Prerequisites

Ensure **Docker** and **Docker Compose** are installed.

```
docker --version
docker-compose --version
```

Docker Compose Configuration

The `docker-compose.yml` file below configures each component in a networked environment, enabling them to communicate with each other seamlessly.

```
version: '3.8'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.0.0
    container_name: zookeeper
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - 2181:2181

  kafka:
    image: confluentinc/cp-kafka:7.0.0
    container_name: kafka
    depends_on:
      - zookeeper
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
    ports:
      - 9092:9092

  postgres:
    image: postgres:13
    container_name: postgres
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: password
      POSTGRES_DB: testdb
    ports:
      - 5432:5432
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql:ro
    logging:
      options:
        max-size: "10m"
        max-file: "3"

  debezium:
    image: debezium/connect:1.8
    container_name: debezium
    depends_on:
      - kafka
      - postgres
    environment:
      BOOTSTRAP_SERVERS: kafka:9092
      GROUP_ID: 1
      CONFIG_STORAGE_TOPIC: debezium-connect-configs
      OFFSET_STORAGE_TOPIC: debezium-connect-offsets
      STATUS_STORAGE_TOPIC: debezium-connect-status
    ports:
      - 8083:8083
    logging:
      options:
        max-size: "10m"
        max-file: "3"
```

Explanation of `docker-compose.yml` Configurations

Zookeeper

- **Image**: confluentinc/cp-zookeeper:7.0.0 – This is the Zookeeper image required by Kafka.
- **Environment Variables**:
  - ZOOKEEPER_CLIENT_PORT: Port for Zookeeper clients to connect to.
  - ZOOKEEPER_TICK_TIME: Interval for Zookeeper heartbeats.
- **Ports**: Exposes port 2181 for Zookeeper client connections.

Kafka

- **Image**: confluentinc/cp-kafka:7.0.0
- **Dependencies**: Depends on the zookeeper container.
- **Environment Variables**:
  - KAFKA_BROKER_ID: Unique ID for the Kafka broker.
  - KAFKA_ZOOKEEPER_CONNECT: Connection string for Zookeeper.
  - KAFKA_ADVERTISED_LISTENERS: Advertised listener for Kafka to allow other services to connect.

    - KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: Specifies replication factor for internal Kafka topics.

- **Ports**: Exposes port 9092 for Kafka connections.

Postgres

- **Image**: postgres:13
- **Environment Variables**:
    - POSTGRES_USER: Username for the Postgres database.
    - POSTGRES_PASSWORD: Password for the Postgres database.
    - POSTGRES_DB: Name of the database created on startup.
- **Ports**: Exposes port 5432 for database connections.
- **Volumes**: Uses an init.sql file to initialize the database with sample data.
- **Logging Options**:
    - max-size: Sets maximum log size.
    - max-file: Sets maximum number of log files to retain.

Debezium

- **Image**: debezium/connect:1.8
- **Dependencies**: Depends on both kafka and postgres services.
- **Environment Variables**:
    - BOOTSTRAP_SERVERS: Kafka broker address.
    - GROUP_ID: Consumer group ID for Debezium.
    - CONFIG_STORAGE_TOPIC, OFFSET_STORAGE_TOPIC, STATUS_STORAGE_TOPIC: Kafka topics for storing Debezium configuration, offsets, and status.
- **Ports**: Exposes port 8083 for Debezium REST API.

Starting the Services

To start the services defined in the Docker Compose file, run:

```
docker-compose up -d
```

This command runs the containers in detached mode.

Registering a Debezium Connector

Once all services are up and running, register a Debezium connector to monitor changes in Postgres. The following example assumes Debezium listens to the `testdb` database.

```
curl -i -X POST -H "Accept:application/json" -H "Content-Type:application/json" \
 http://localhost:8083/connectors/ \
 -d '{
     "name": "postgres-connector",
     "config": {
        "connector.class":
"io.debezium.connector.postgresql.PostgresConnector",
        "tasks.max": "1",
        "database.hostname": "postgres",
        "database.port": "5432",
        "database.user": "postgres",
        "database.password": "password",
        "database.dbname": "testdb",
        "database.server.name": "pg-changes",
        "table.include.list": "public.api",
        "plugin.name": "pgoutput"
     }
   }'
```

Explanation of the Connector Configuration

- **connector.class**: Specifies the Debezium connector for PostgreSQL.
- **tasks.max**: Number of tasks to run concurrently.
- **database.hostname**: Hostname of the Postgres container.
- **database.port**: Port for Postgres.
- **database.user** and **database.password**: Credentials for connecting to Postgres.
- **database.dbname**: Name of the monitored database.
- **database.server.name**: Logical name used for topic naming in Kafka.
- **table.include.list**: Specifies which tables to monitor.
- **plugin.name**: Sets the output plugin, pgoutput for PostgreSQL.

Verifying Data Capture in Kafka

After configuring Debezium, changes in the `public.api` table in Postgres will be streamed to the Kafka topic `pg-changes.public.api`.

You can use `kafka-console-consumer` to check messages in the topic:

```
docker exec -it kafka kafka-console-consumer --bootstrap-server
kafka:9092 --topic pg-changes.public.api --from-beginning
```

Testing the POC

1. **Insert, Update, or Delete** data in the public.api/public.api_usage table in Postgres using api's from baas-analytical-module.
2. **Monitor** the Kafka topic for change events.
3. **Verify** the CDC pipeline by observing the streamed data in Kafka.
4. Observe the message consumed by baas-charges-module.

Stopping and Cleaning Up

To stop and remove the containers, use:

```
docker-compose down
```