Debezium Configuration: ms-spring-boot-debezium-master-slave

This service describes the steps of registering and interacting with a Debezium connector

Introduction

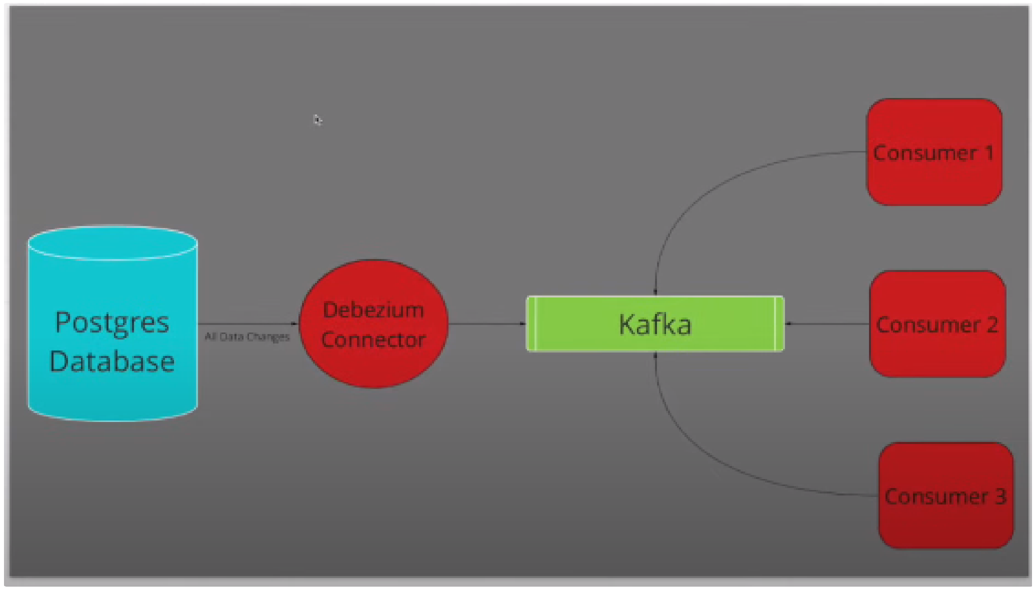
Debezium is an open-source platform for change data capture (CDC) that captures and streams database changes in real-time.

Description

Let’s take the example of a restaurant.

You’re making a dinner reservation. Usually, you’ll make this reservation through a valet. The valet then instructs waiter(s) to lay your table. The valet may also relay any special dietsry requirements to the waiter who can then relay them to the chef. In case you cancel or reschedule, the valet informs the waiter and table structure is freed for other customers. Likewise happens if you don’t honor your reservation in time.

The restauranter is the service/application triggering data changes to the DB. The valet is a message broker system such as Kafka or ActiveMQ. The waiter can be viewed as the Debezium connector who relays the data changes in a user-friendly manner, and the team behind the counter are the ones affected by changes to meals (orders), so they are the DB.



Debezium’s place within a service

Setup

Setup *docker-compose.yml*

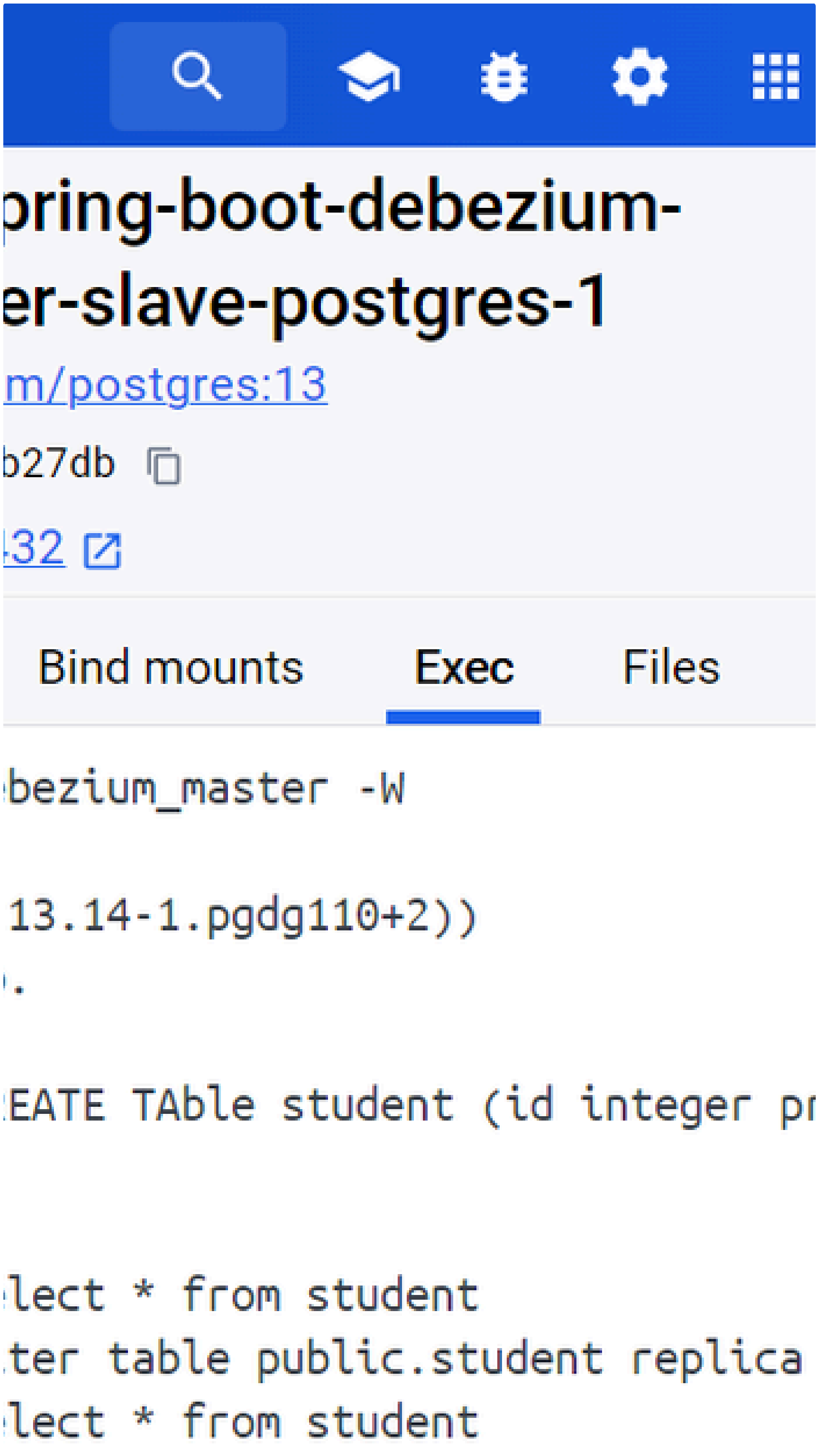
```yaml
version: '3.7'
services:
  postgres:
    image: debezium/postgres:13
    ports:
      - 5432:5432
    volumes:
      - ./app:/app
    environment:
      - POSTGRES_USER=<DB USER>
      - POSTGRES_PASSWORD=<DB PASS>
      - POSTGRES_DB=<DB NAME>

  pgadmin:
    image: dpage/pgadmin4
    ports:
      - 5051:80
    environment:
      - PGADMIN_DEFAULT_EMAIL=<YOUR USERNAME>
      - PGADMIN_DEFAULT_PASSWORD=<YOUR PASSWORD>
    depends_on:
      - postgres
    restart: always

  zookeeper:
    image: confluentinc/cp-zookeeper:6.2.1
    ports:
      - 2181:2181
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    restart: always

  kafka:
    image: confluentinc/cp-enterprise-kafka:6.2.1
    ports:
      - 9092:9092
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_JMX_PORT: 9991
    depends_on: [zookeeper]
    restart: always

  debezium:
    image: debezium/connect:1.6
    ports:
      - 8083:8083
    environment:
      BOOTSTRAP_SERVERS: kafka:9092
      GROUP_ID: 1
      CONFIG_STORAGE_TOPIC: connect_configs
      OFFSET_STORAGE_TOPIC: connect_offsets
      KEY_CONVERTER: io.confluent.connect.avro.AvroConverter
      VALUE_CONVERTER: io.confluent.connect.avro.AvroConverter
      CONNECT_KEY_CONVERTER_SCHEMA_REGISTRY_URL: http://schema-registry:8081
      CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_URL: http://schema-registry:8081
      STATUS_STORAGE_TOPIC: debezium_connect_status
      CONFIG_STORAGE_REPLICATION_FACTOR: 1
      OFFSET_STORAGE_REPLICATION_FACTOR: 1
      STATUS_STORAGE_REPLICATION_FACTOR: 1
```

```yaml
      OFFSET_FLUSH_INTERVAL_MS: 60000
    depends_on: [kafka]
    restart: always

  schema-registry:
    image: confluentinc/cp-schema-registry:6.2.1
    ports:
      - 8081:8081
    environment:
      SCHEMA_REGISTRY_KAFKASTORE_CONNECTION_URL: zookeeper:2181
      SCHEMA_REGISTRY_HOST_NAME: schema-registry
      SCHEMA_REGISTRY_LISTENERS: http://localhost:8081, http://schema-registry:8081
    depends_on: [zookeeper,kafka]
    restart: always

kafka_manager:
  image: hlebalbau/kafka-manager:stable
  restart: always
  ports:
    "9000:9000"
  depends_on: [zookeeper, kafka]

  environment:
    ZK_HOSTS: "Zookeeper: 2181"
    APPLICATION_SECRET: "random-secret"
  command: -Dpidfile.path=/dev/null
```

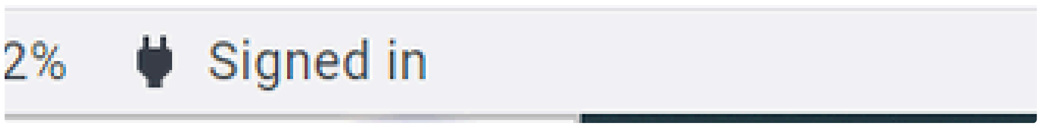
Run the file with this command to create and run the images

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

Point to the relation where data changes are to be monitored



```
bezium_master -W

13.14-1.pgdg110+2))
'.

EATE TAble student (id integer pr


lect * from student
ter table public.student replica
lect * from student
```



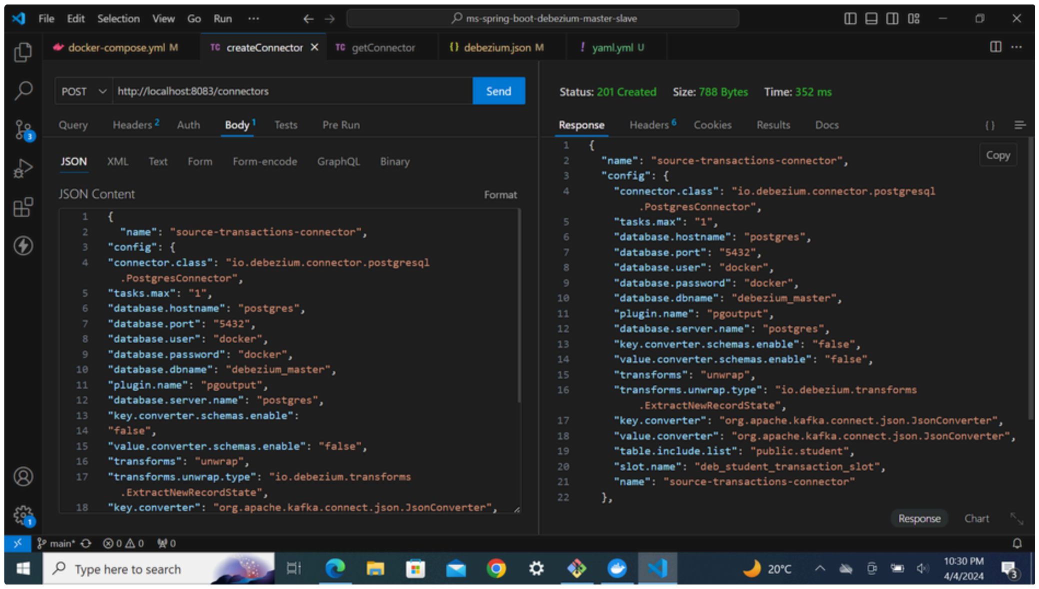
Setup the connector in *debezium.json*

```json
{
  "name": "<GIVE YOUR TRANSACTION A NAME>",
  "config": {
    "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
    "tasks.max": "1",
    "database.hostname": "<HOST IP>",
    "database.port": "<PORT>",
    "database.user": "<DB USER>",
    "database.password": "<DB PASS>",
    "database.dbname": "<DB NAME>",
    "plugin.name": "pgoutput",
    "database.server.name": "postgres",
    "key.converter.schemas.enable": "false",
    "value.converter.schemas.enable": "false",
    "transforms": "unwrap",
    "transforms.unwrap.type": "io.debezium.transforms.ExtractNewRecordState",
    "key.converter": "org.apache.kafka.connect.json.JsonConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "table.include.list": "<YOUR TABLE>",
    "slot.name": "<GIVE YOUR SLOT A NAME>"
  }
}
```

Register the connector

Endpoint:

http://localhost:8083/connectors



http://localhost:8083/connectors

```
{
  "name": "source-transactions-connector",
"config": {
"connector.class": "io.debezium.connector.postgresql
  .PostgresConnector",
"tasks.max": "1",
"database.hostname": "postgres",
"database.port": "5432",
"database.user": "docker",
"database.password": "docker",
"database.dbname": "debezium_master",
"plugin.name": "pgoutput",
"database.server.name": "postgres",
"key.converter.schemas.enable":
"false",
"value.converter.schemas.enable": "false",
"transforms": "unwrap",
"transforms.unwrap.type": "io.debezium.transforms
  .ExtractNewRecordState",
"key.converter": "org.apache.kafka.connect.json.JsonConverter",
```

Status: 201 Created Size: 788 Bytes Time: 352 ms

```
{
  "name": "source-transactions-connector",
  "config": {
    "connector.class": "io.debezium.connector.postgresql
        .PostgresConnector",
    "tasks.max": "1",
    "database.hostname": "postgres",
    "database.port": "5432",
    "database.user": "docker",
    "database.password": "docker",
    "database.dbname": "debezium_master",
    "plugin.name": "pgoutput",
    "database.server.name": "postgres",
    "key.converter.schemas.enable": "false",
    "value.converter.schemas.enable": "false",
    "transforms": "unwrap",
    "transforms.unwrap.type": "io.debezium.transforms
        .ExtractNewRecordState",
    "key.converter": "org.apache.kafka.connect.json.JsonConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "table.include.list": "public.student",
    "slot.name": "deb_student_transaction_slot",
    "name": "source-transactions-connector"
  },
```

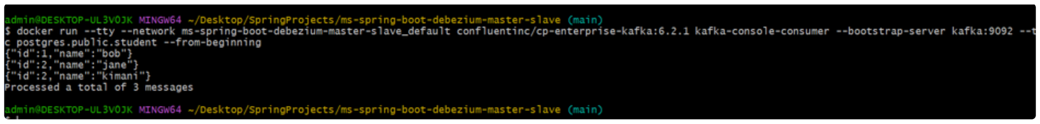
Debezium connector registered

OR :

```
curl -i -X POST -H "Accept:application/json" -H "Content-Type:application/json" 127.0.0.1:8083/connectors --data
```

Start listening for changes

```
docker run --tty --network ms-spring-boot-debezium-master-slave_default confluentinc/cp-enterprise-kafka:6.2.1 ka
```



```
admin@DESKTOP-UL3VOJK MINGW64 ~/Desktop/SpringProjects/ms-spring-boot-debezium-master-slave (main)
$ docker run --tty --network ms-spring-boot-debezium-master-slave_default confluentinc/cp-enterprise-kafka:6.2.1 kafka-console-consumer --bootstrap-server kafka:9092 --t
c postgres.public.student --from-beginning
{"id":1,"name":"bob"}
{"id":2,"name":"jane"}
{"id":2,"name":"kimani"}
Processed a total of 3 messages

admin@DESKTOP-UL3VOJK MINGW64 ~/Desktop/SpringProjects/ms-spring-boot-debezium-master-slave (main)
```

DB changes streamed through Debezium

Resources

Resource	Endpoint
Postman collection	https://api.postman.com/collections/24452708-0d9242ad-a5c1-4bd3-8be5-461af1902c5b?access_key=PMAT-01HTZM4ABQ12WF6B658502WPCH

Stream your PostgreSQL changes into Kafka with Debezium	https://www.youtube.com/watch?v=YZRHqRznO-o
How to Stream Data using Apache Kafka & Debezium from Postgres | Real Time ETL | ETL | Part 2	https://www.youtube.com/watch?v=xh9rVSqNHMI
Official Debezium Documentation	Tutorial :: Debezium Documentation