**CDC – Debezium Documentation**

**Prerequisites for Setting Up CDC with Debezium**

1. **System Requirements:**
2. A system with **Docker** and **Docker Compose** installed.
3. At least **4GB RAM** and **sufficient disk space** to run multiple database containers.
4. A Linux-based system (Ubuntu, CentOS) or Windows with **WSL2** for better compatibility.
5. **Installed Software & Dependencies:**
   1. **Docker**: To run containers for various databases.
   2. **Postman**: To test API endpoints.
6. **Network & Security Setup:**
   1. Open necessary **ports**:
      * **MySQL** → 3306
      * **PostgreSQL** → 5432
      * **SQL Server** → 1433
      * **MongoDB** → 27017
      * **Kafka & Debezium Connect** → 8083
   2. Ensure **firewall rules** allow database connections.
   3. Run databases in **Docker bridge mode** to enable inter-container communication.
7. **Database Configuration:**
   1. Set **username and password** for each database as required.
   2. Enable **Change Data Capture (CDC)** in SQL Server and PostgreSQL.
   3. Grant necessary **permissions** for replication and CDC.
8. **Kafka & Debezium Setup:**
   1. Ensure **Kafka is running** for CDC events processing.
   2. Start **Debezium Connect** to manage database connectors.
9. **Testing & Verification:**
   1. Use **Postman** or curl commands to verify API endpoints.
   2. Check CDC logs to confirm data changes are captured.
   3. Ensure **Kafka topics** are receiving events.

**Running Shell Scripts**

The following commands are used to execute shell scripts for setting up CDC (Change Data Capture) and database configurations:

1. Grant Execute Permissions:

**chmod +x cdc\_installation.sh**

**chmod +x database\_installation.sh**

* + chmod +x makes the script executable. This is required before running shell scripts in Linux.

1. Execute the Scripts:

**./cdc\_installation.sh**

**./database\_installation.sh**

* + ./ is used to run the script from the current directory. These scripts automate the installation and setup of CDC and database configurations.

This ensures a seamless and automated deployment of CDC and database setup without manual intervention.

**SQL Server Database Connection Setup**

1. Connect to SQL Server Container

Run the following command to log into the SQL Server instance running in Docker:

**docker exec -it sqlserver /opt/mssql-tools18/bin/sqlcmd -S localhost -U SA -P "Password123!" -C -N**

* sqlcmd is the command-line utility for SQL Server.
* -U SA -P "Password123!" logs in using the SA (System Administrator) user.

2. Create a Database

**CREATE DATABASE inventory;**

**GO**

**USE inventory;**

**GO**

* This creates and selects a database named inventory.

3. Enable Change Data Capture (CDC) for the Database

**EXEC sys.sp\_cdc\_enable\_db;**

**GO**

* Enables CDC at the database level.

4. Create a Table and Insert Data

**CREATE TABLE dbo.customers(name VARCHAR(50) NOT NULL, email VARCHAR(50) NOT NULL);**

**GO**

**INSERT INTO dbo.customers (name, email) VALUES ('John', 'john@gmail.com'), ('Yash', 'yash@gmail.com');**

**GO**

* Creates a customer table and inserts sample records.

5. Enable CDC for the Table

**EXEC sys.sp\_cdc\_enable\_table @source\_schema = 'dbo', @source\_name = 'customers', @role\_name = NULL, @supports\_net\_changes = 0;**

**GO**

* Enables CDC for the customer table.

6. Verify CDC is Enabled

**SELECT name, is\_cdc\_enabled FROM sys.databases WHERE name = DB\_NAME();**

**GO**

* The is\_cdc\_enabled column should return 1 (CDC enabled).

7. Update Records

**UPDATE dbo.customers SET email = 'new\_email@example.com' WHERE name = 'Yash';**

**GO**

* Updates Yash's email.

8. Exit SQL Server Shell

**exit**

**Checking SQL Server CDC Connection in Postman**

This section provides steps to verify the Change Data Capture (CDC) connection in **Postman** for SQL Server.

1. Create a CDC Connector

Send a **POST** request to the following URL:

**URL: http://127.0.0.1:5000/api/connector**

**Method: POST**

**{**

**"db\_type": "mssql",**

**"db\_name": "inventory",**

**"port": "1433",**

**"user\_name": "SA",**

**"password": "Password123!",**

**"kafka\_topic": "dbserver1"**

**}**

* db\_type: Specifies the database type (mssql for SQL Server).
* topic: The topic name for Kafka.
* port: SQL Server's port (1433 by default).
* user & pwd: SQL Server authentication credentials.
* mssql\_db: The database name (inventory).

2. Check CDC Connector Status

Send a **GET** request to verify the connector's status:

**URL: http://127.0.0.1:5000/api/connector/mssql-connector/status**

**Method: GET**

* Ensures the connector is running and active.

3. List Available Topics

To retrieve all Kafka topics related to CDC, send a **GET** request:

**URL: http://127.0.0.1:5000/api/topics**

**Method: GET**

* This confirms that topics are created for CDC-tracked tables.

4. Consume CDC Messages

Send a **POST** request to fetch CDC records from a specific topic:

**URL: http://127.0.0.1:5000/api/consume\_messages**

**Method: POST**

**{**

**"topic": "dbserver1.inventory.dbo.customers"**

**}**

* topic: The topic where CDC changes are stored.
* max\_messages: The number of CDC events to retrieve.

**Outcome**

* If CDC is correctly configured, Postman should return JSON data containing **inserted, updated, or deleted records**.
* If errors occur, check **connector logs** and verify **CDC is enabled** in SQL Server.

**MongoDB Database Connection Setup for CDC**

1. Initialize MongoDB Replica Set

MongoDB needs a replica set for CDC. Run the following command:

**docker exec -it mongodb mongosh --eval "rs.initiate({\_id: 'myrs', members: [{\_id: 0, host: 'host.docker.internal:27017'}]})"**

* rs.initiate() initializes the replica set.
* host.docker.internal allows the database to be accessed from the host machine.

Check Replica Set Status:

**docker exec -it mongodb mongosh --eval "rs.status()"**

* This ensures the replica set is properly configured.

2. Create MongoDB Admin User

Execute the following command to create an admin user:

**docker exec -it mongodb mongosh admin --eval "db.createUser({user: 'admin', pwd: 'adminpassword', roles: [{ role: 'userAdminAnyDatabase', db: 'admin' }, { role: 'readWriteAnyDatabase', db: 'admin' }]})"**

* This user has admin privileges across all databases.

Login to MongoDB

**docker exec -it mongodb mongosh**

3. Create a Custom User for CDC

Inside the MongoDB shell, create a new user for CDC:

**db.createUser({user: "debezium", pwd: "dbz", roles: [{ role: "read", db: "admin" }, { role: "readWrite", db: "config" }, { role: "readWrite", db: "local" }, { role: "readWrite", db: "test\_db" }, { role: "clusterMonitor", db: "admin" }, { role: "readAnyDatabase", db: "admin" }]});**

* Debezium user is created with the necessary CDC permissions.

4. Create and Populate Database

Switch to the Database

**use test\_db;**

Create a Collection

**db.createCollection("mycollection");**

Insert Sample Data

**db.mycollection.insertOne({"name": "abc", "age": 25});**

Verify the Data

**db.mycollection.find();**

5. Exit the MongoDB Shell

**exit**

* This closes the MongoDB CLI session.

**To check connection in Postman for MongoDB**

1. Creating the MongoDB Connector

To create a connector for MongoDB, send a POST request to:

**URL: http://127.0.0.1:5000/api/connector**

**Method: POST**

**{**

**"db\_type": "mongodb",**

**"db\_name": "test\_db",**

**"port": "27017",**

**"user\_name": "debezium",**

**"password": "dbz",**

**"kafka\_topic": "mongo12"**

**}**

2. Checking Connector Status

To verify if the MongoDB CDC connector is running, send a GET request to:

**URL: http://127.0.0.1:5000/api/connector/mongodb-connector/status**

**Method: GET**

3. Listing Available Topics

To get a list of available topics that CDC is monitoring, send a GET request to:

**URL: http://127.0.0.1:5000/api/topics**

**Method: GET**

4. Consuming Messages from a Topic

To consume messages from the MongoDB CDC topic, send a POST request to:

**URL: http://127.0.0.1:5000/api/cdc/consume\_messages**

**Method: POST**

**{**

**"topic": "mongo12.test\_db.mycollection"**

**}**

This setup ensures that MongoDB changes are captured and can be monitored via the CDC system.

**MariaDB Database Connection Setup**

1. Logging into MariaDB

To log in using the root user, run:

**docker exec -it mariadb /usr/bin/mariadb -u root -proot**

2. Granting Necessary Privileges

Grant permissions for the user to enable CDC:

**GRANT RELOAD, REPLICATION CLIENT, REPLICATION SLAVE ON \*.\* TO 'mrdbuser'@'%';**

3. Creating and Using the Database

Switch to the MariaDB shell and create the database:

**USE mrdb;**

4. Creating a Table

Create a customers table with auto-incrementing IDs:

**CREATE TABLE customers (id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100), created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP);**

5. Inserting Sample Data

Insert sample customer records:

**INSERT INTO customers (name, email) VALUES ('John Doe', 'john@example.com'), ('Jane Smith', 'jane@example.com');**

6. Exiting MariaDB Shell

To exit the MariaDB shell, run:

**exit**

**To check connection in Postman for MariaDB**

**MariaDB CDC Connector Setup (Using Postman API Requests)**

This section provides API endpoints for setting up and managing a Change Data Capture (CDC) connector for MariaDB.

1. Creating the Connector

Use the following POST request to create a CDC connector for MariaDB:

**URL: http://127.0.0.1:5000/api/connector**

**Method: POST**

**{**

**"db\_type": "mrdb",**

**"db\_name": "mrdb",**

**"port": "3306",**

**"user\_name": "mrdbuser",**

**"password": "mrdbpw",**

**"kafka\_topic": "mrdb1"**

**}**

2. Checking Connector Status

To verify if the connector is running:

**URL:** [**http://127.0.0.1:5000/api/connector/mrdb-connector/status**](http://127.0.0.1:5000/api/connector/mrdb-connector/status)

**Method: GET**

3. Listing Available Topics

Retrieve a list of available topics:

**URL:** [**http://127.0.0.1:5000/api/topics**](http://127.0.0.1:5000/api/topics)

**Method: GET**

4. Consuming CDC Messages

Fetch CDC messages from the MariaDB customers table:

**URL: http://127.0.0.1:5000/api/consume\_messages**

**Method: POST**

**{**

**"topic": "mrdb1.mrdb.customers"**

**}**

These API endpoints help manage the CDC pipeline, allowing you to track and process database changes effectively.

**PostgreSQL Database Connection Set Up**

1. Login to PostgreSQL

**docker exec -it postgres psql -U postgres -d inventory**

2. Verify Replication Slots and Publications

Check if replication slots and publications already exist:

**SELECT \* FROM pg\_replication\_slots;**

**SELECT \* FROM pg\_publication;**

3. Create a Table and Insert Sample Data

Create a customers table and insert initial records:

**CREATE TABLE customers (id SERIAL PRIMARY KEY, name VARCHAR(100), email VARCHAR(100));**

**INSERT INTO customers (name, email) VALUES ('John Doe', 'john@example.com'), ('Jane Smith', 'jane@example.com');**

4. Enable Logical Replication (If Not Already Configured)

Create a publication for the customers table:

**CREATE PUBLICATION dbz\_publication FOR TABLE customers;**

Create a logical replication slot for CDC:

**SELECT pg\_create\_logical\_replication\_slot('dbz\_slot', 'pgoutput');**

5. Exit from PostgreSQL Shell

**exit**

**To check connection in Postman for PostgreSQL**

1. Create a Connector

**URL: http://127.0.0.1:5000/api/connector**

**Method: POST**

**{**

**"db\_type": "postgres",**

**"db\_name": "inventory",**

**"port": "5432",**

**"user\_name": "postgres",**

**"password": "postgres",**

**"kafka\_topic": "postgres\_tp"**

**}**

2. Check Connector Status

To verify if the connector is running, send a GET request to:

**URL: http://127.0.0.1:5000/api/connector/psql-connector/status**

**Method: GET**

3. List Available Topics

To fetch a list of available topics, send a GET request to:

**URL: http://127.0.0.1:5000/api/topics**

**Method: GET**

4. Consume Messages from a Topic

To consume messages from the customers table, send a POST request to:

**URL: http://127.0.0.1:5000/api/messages**

**Method: POST**

**{**

**"topic": "postgres\_tp.public.customers"**

**}**

**MySQL Database Connection Set Up**

1. Login to MySQL

Run the following command to access MySQL:

**docker exec -it mysql mysql -u mysqluser -p**

When prompted, enter the password:

**mysqlpw**

2. Grant Permissions for CDC

Execute the following commands to grant necessary privileges:

**GRANT RELOAD, SHOW DATABASES, REPLICATION SLAVE, REPLICATION CLIENT, SELECT ON \*.\* TO 'mysqluser'@'%';**

**FLUSH PRIVILEGES;**

3. Use the Database

Switch to the inventory database:

**USE inventory;**

4. Create Customers Table

**CREATE TABLE customers (id INT PRIMARY KEY AUTO\_INCREMENT, name VARCHAR(255) NOT NULL, email VARCHAR(255) NOT NULL);**

5. Insert Sample Data

**INSERT INTO customers (name, email) VALUES ('John Doe', 'john@example.com'), ('Jane Smith', 'jane@example.com');**

6. Exit MySQL Shell

**exit**

**To check connection in Postman for MySQL**

1. Create Connector

**URL: http://127.0.0.1:5000/api/cdc/connector**

**Method: POST**

**{**

**"db\_type": "mysql",**

**"db\_name": "inventory",**

**"port": "3307",**

**"user\_name": "mysqluser",**

**"password": "mysqlpw",**

**"kafka\_topic": "mysql\_tp"**

**}**

2. Check Connector Status

**URL: http://127.0.0.1:5000/api/connector/mysql-connector/status**

**Method: GET**

3. List Available Topics

**URL: http://127.0.0.1:5000/api/topics**

**Method: GET**

4. Consume Messages from CDC Topic

**URL: http://127.0.0.1:5000/api/consume\_messages**

**Method: POST**

**{**

**"topic": "mysql\_tp.inventory.customers"**

**}**