**MQT Lab 7**

**Lab1:**

**1. Login into admin application http://localhost:8080**

**Login as**

System: PostgreSQL

Server: postgres

Username: demo

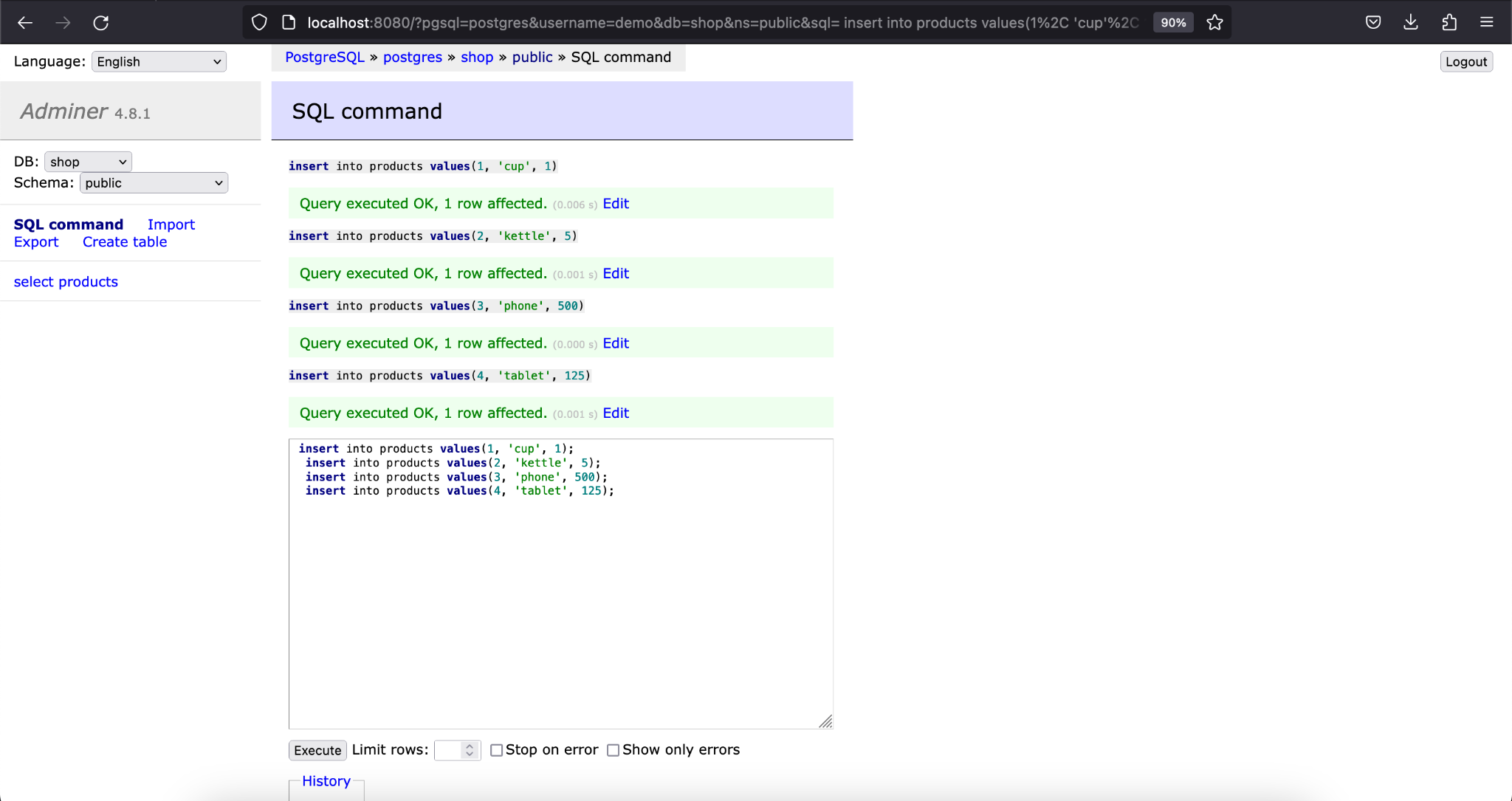
Password: demo

Database: shop

**2. Create a database (SQL command)**



**3. Enter SQL Command**

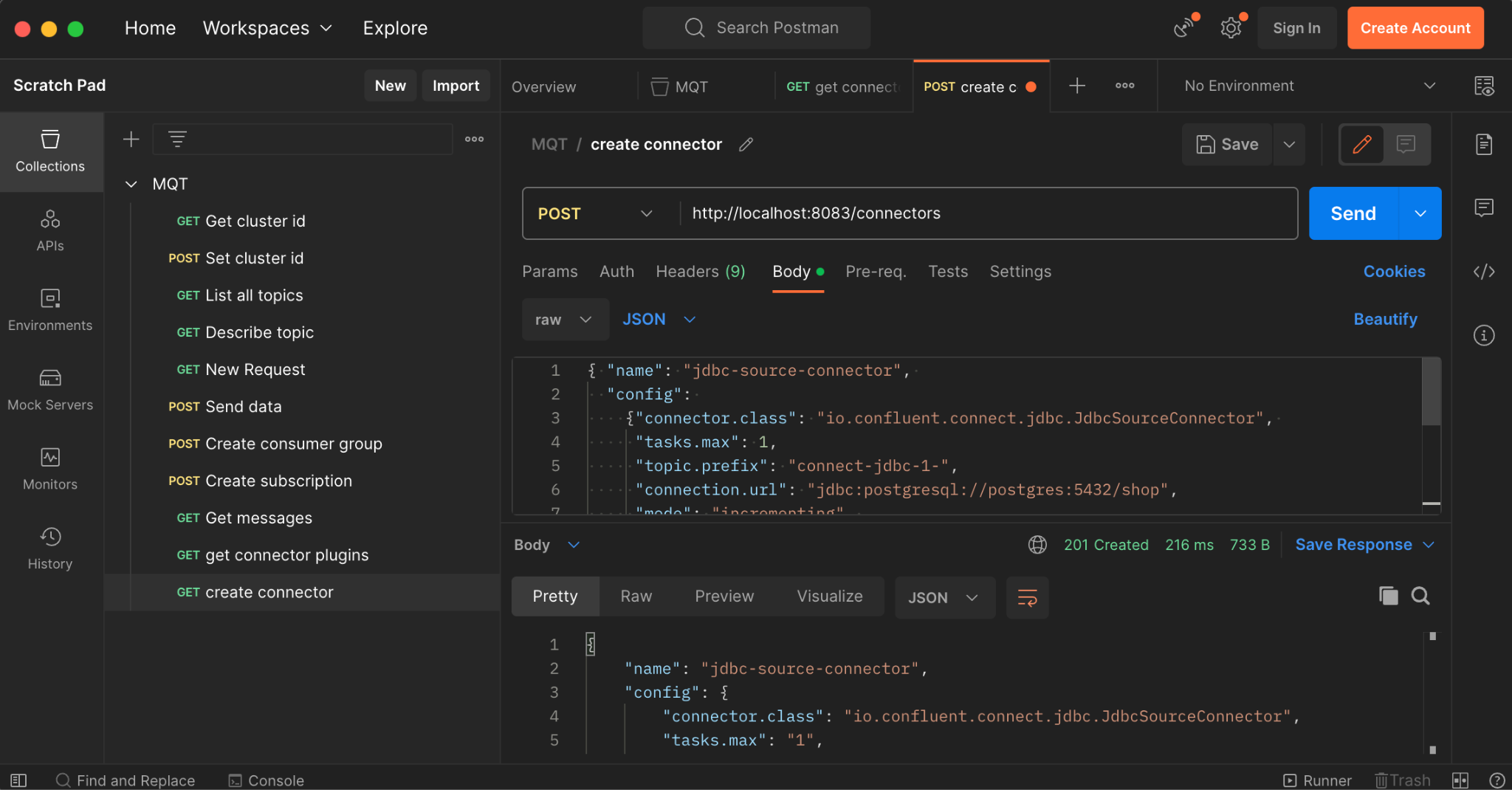


**4. Check connector-plugins through control center or through**

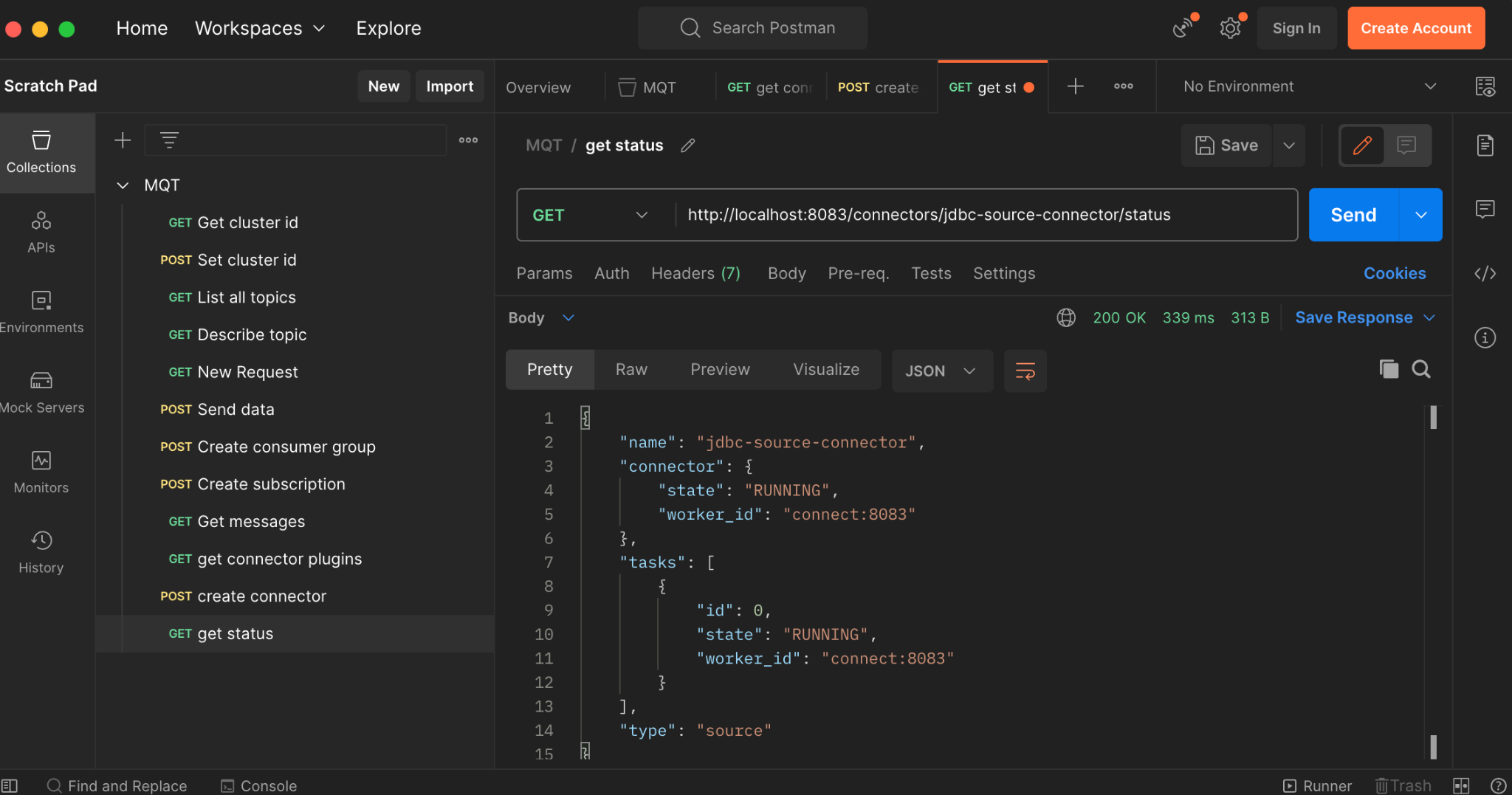
curl http://localhost:8083/connector-plugins from Windows PowerShell or MacOS shell

or @GET http://localhost:8083/connector-plugins from Postman

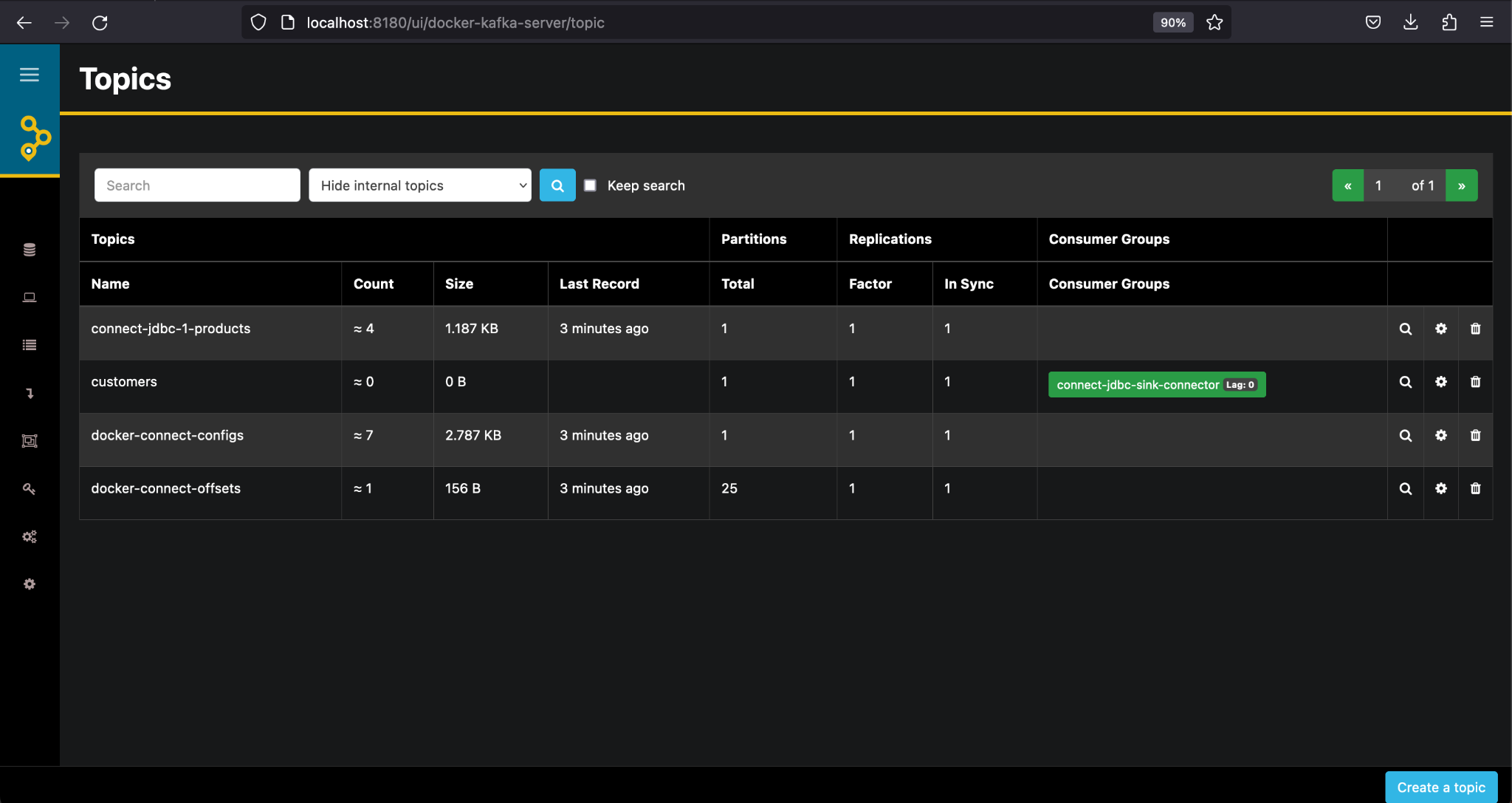
**5. Add JDBC source connector**

****

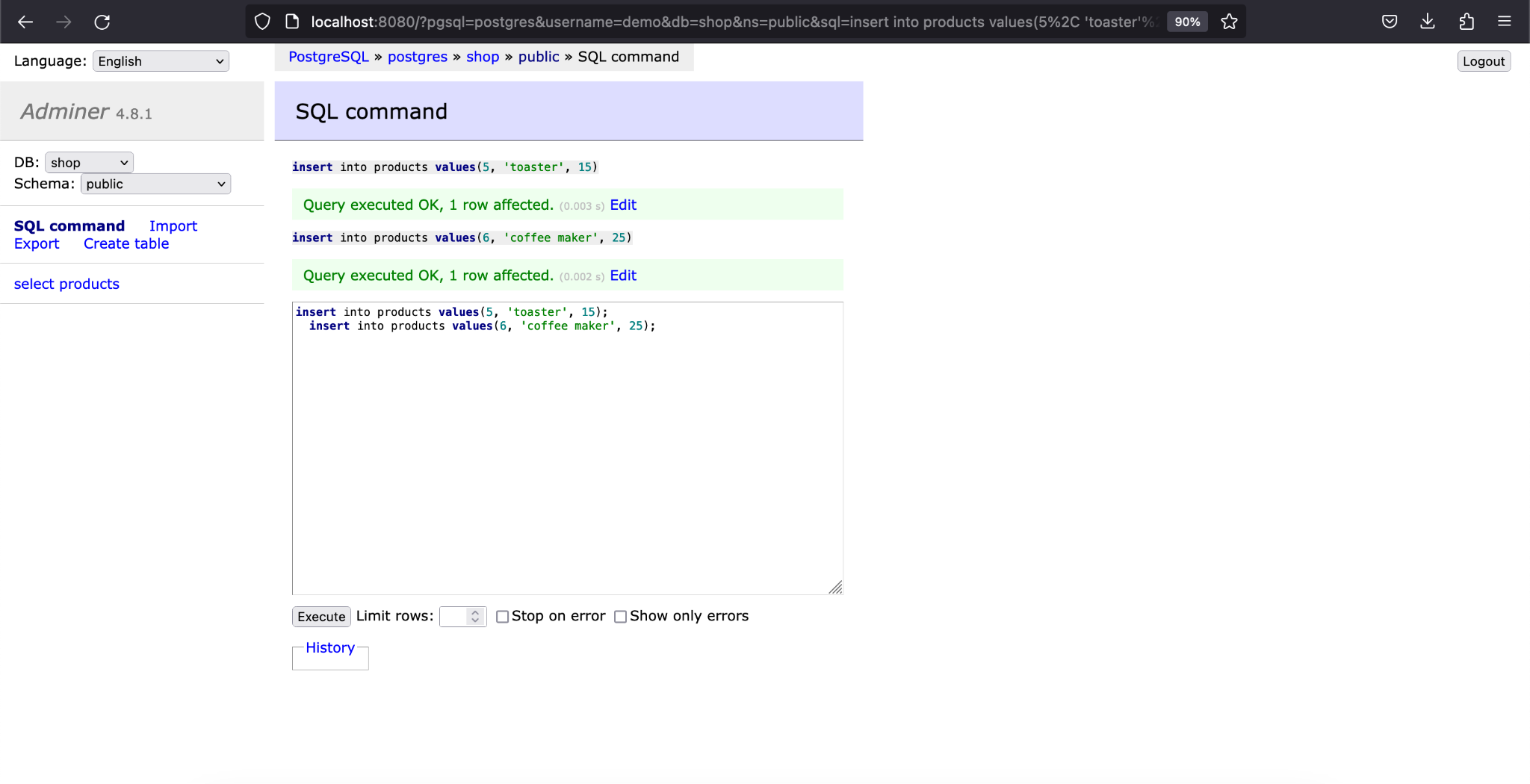
**6. Check connectors through Control center or through REST**



**7. Check messages**

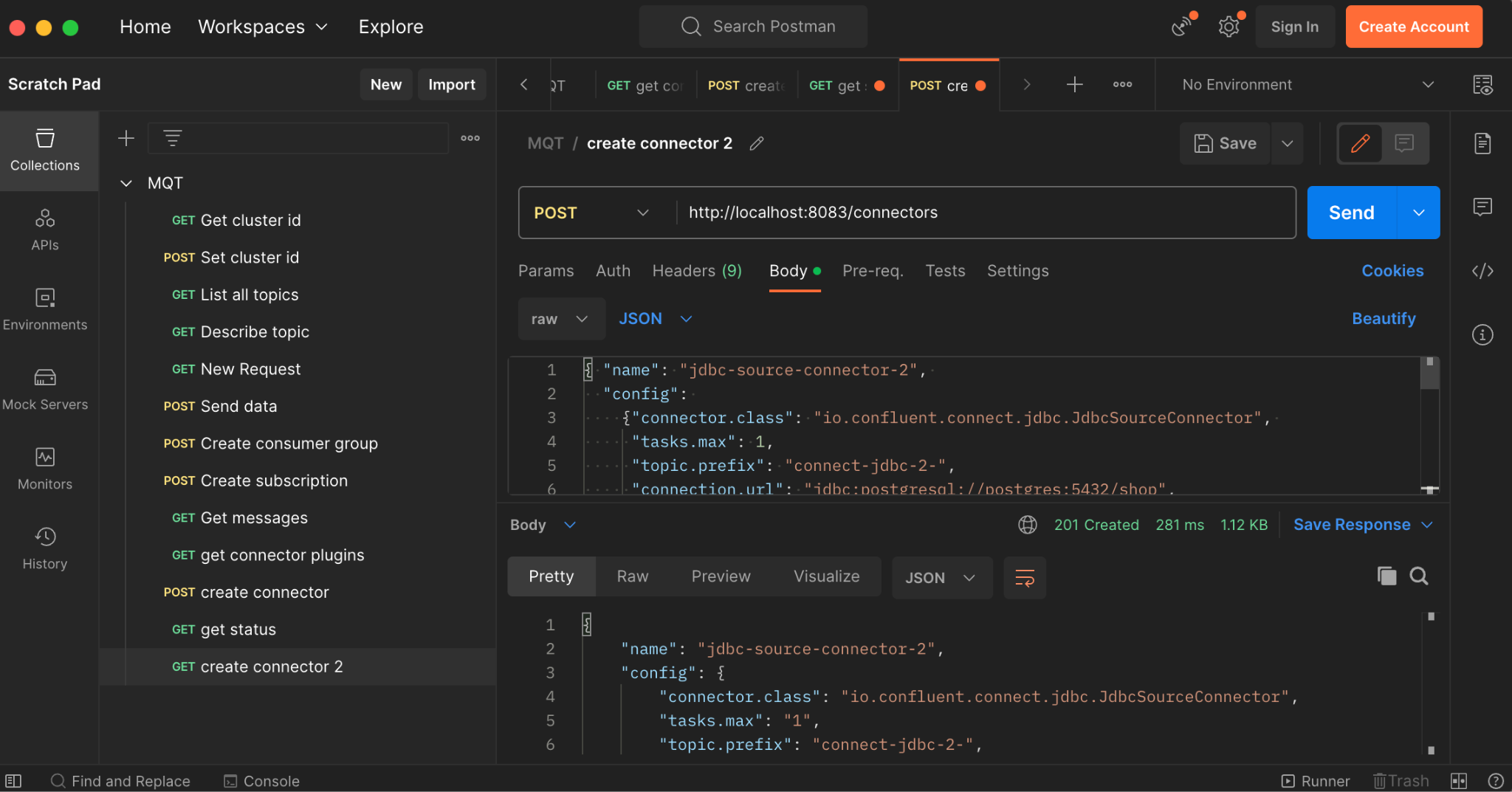


**8. Add more data**

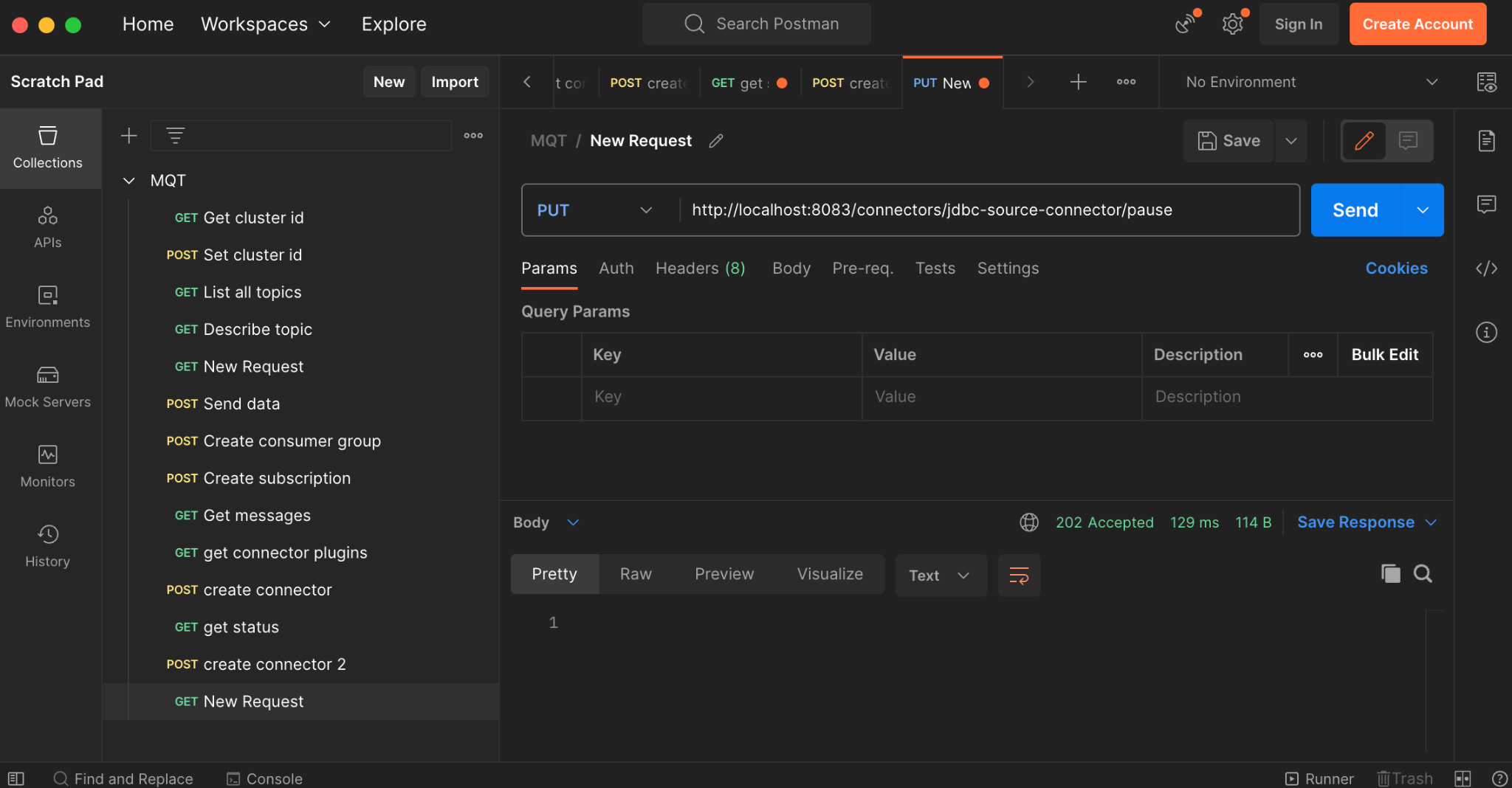


**Lab 2:**

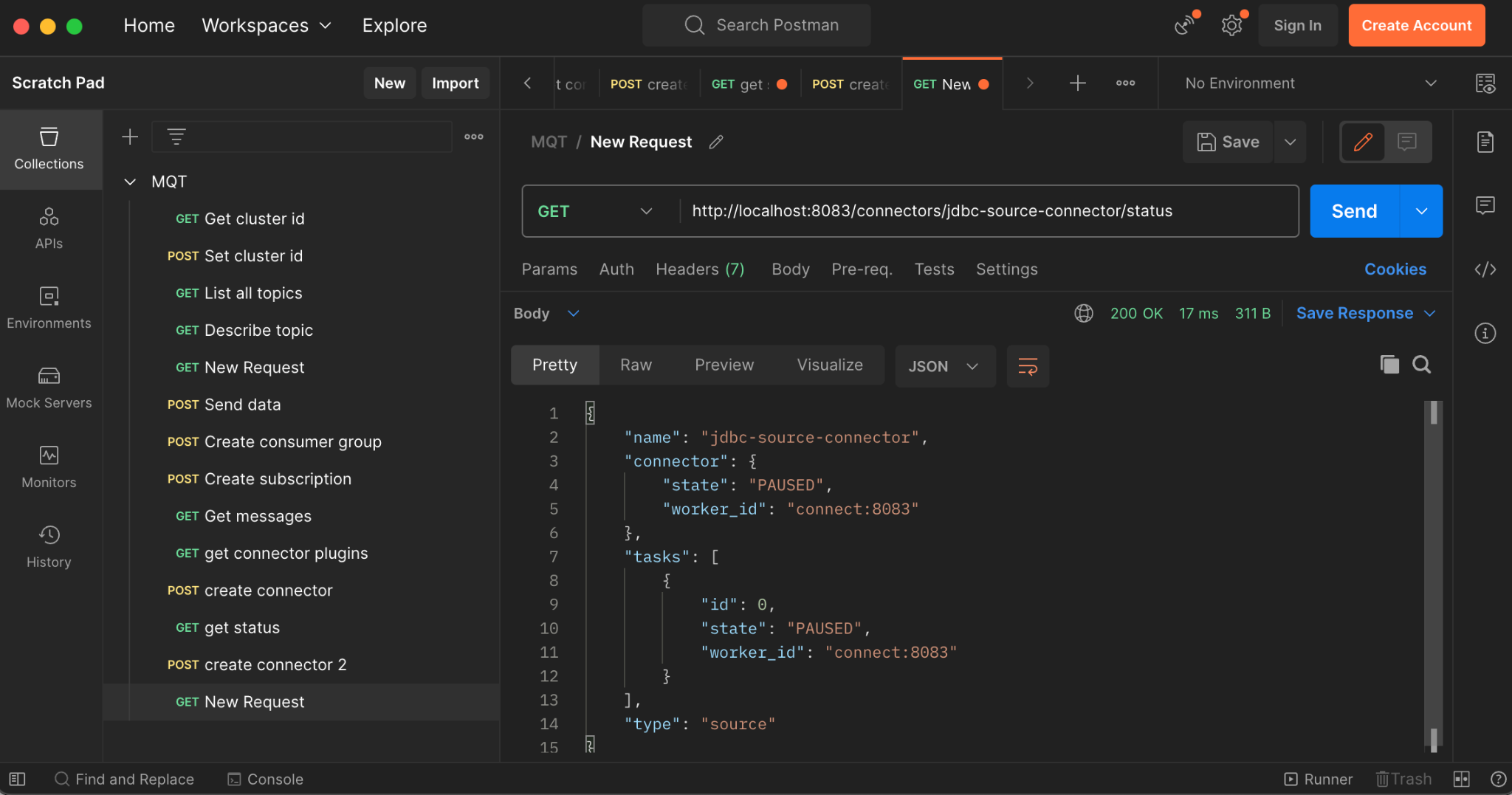
**1. Add JDBC source connector**



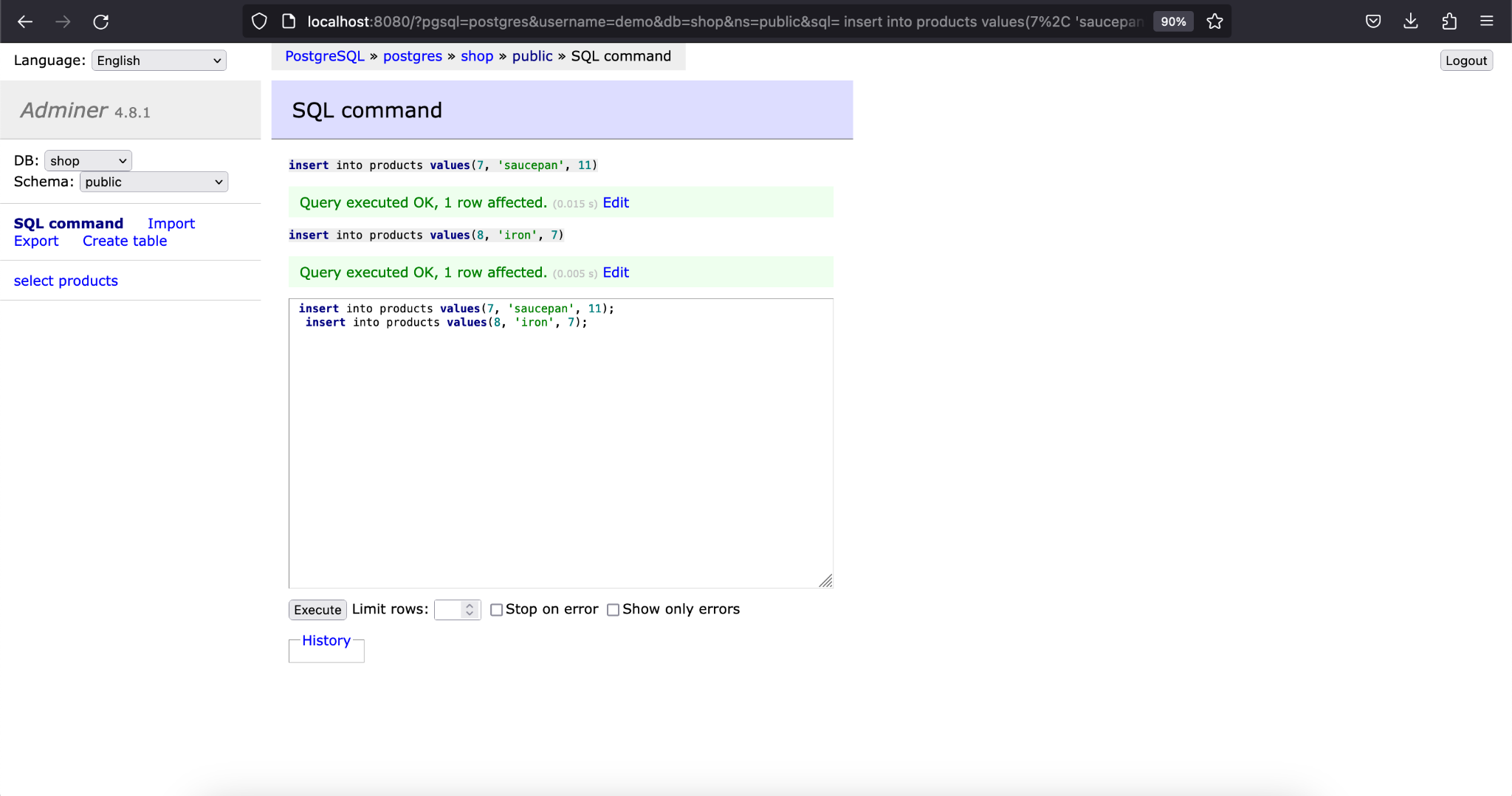
**2. Stop connector**



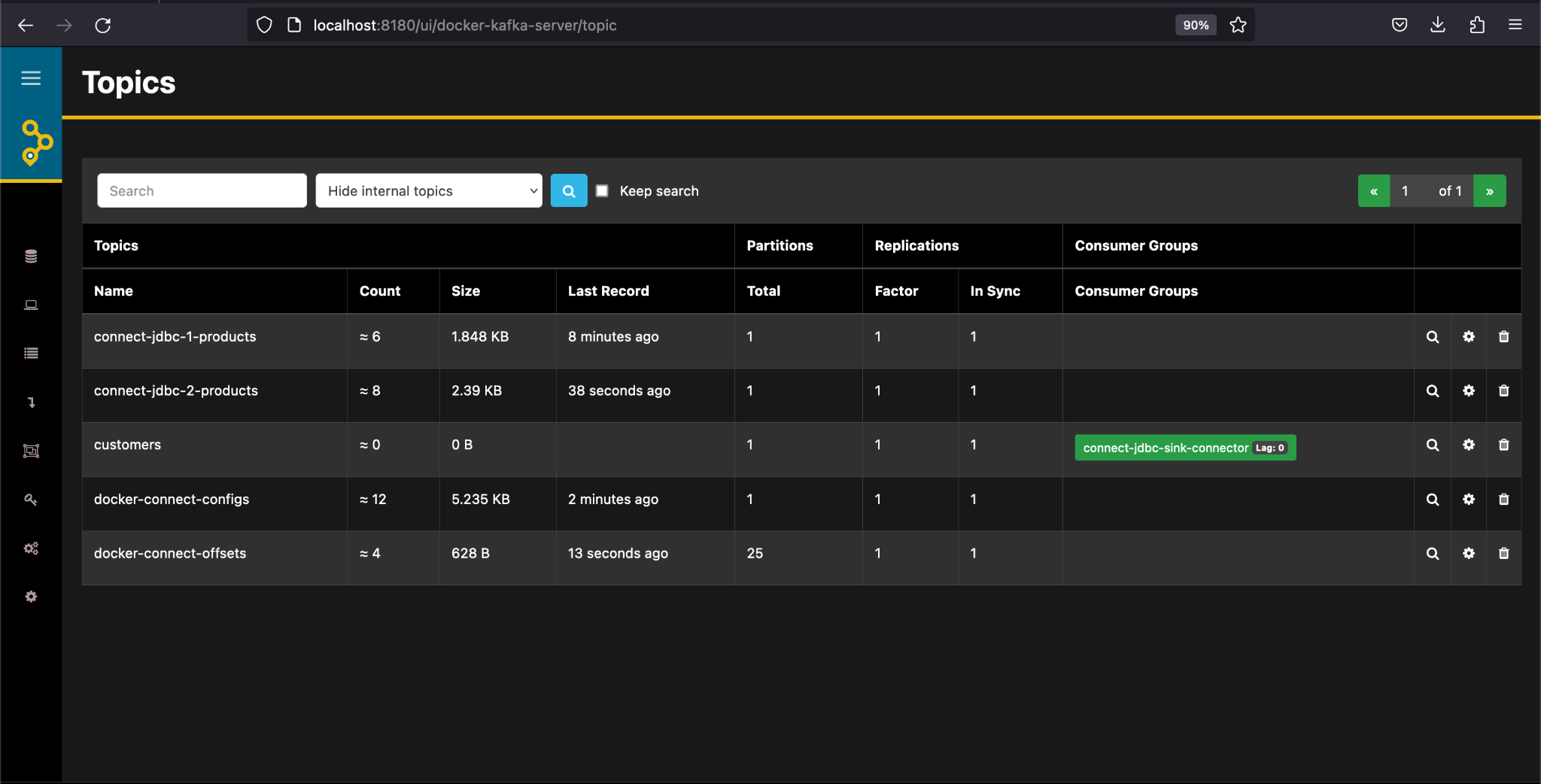
**3. Check status**



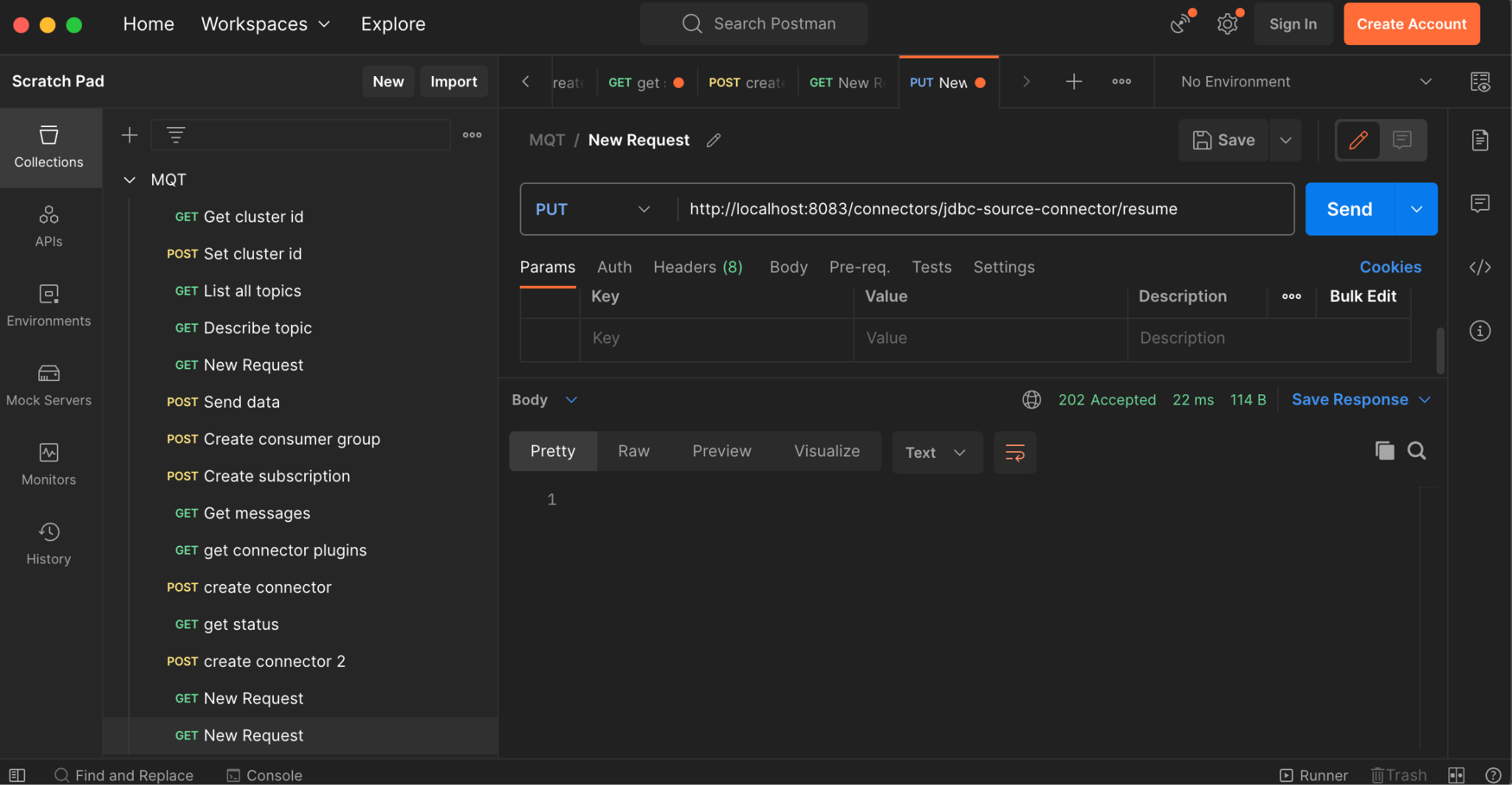
**4. Add new data (http://localhost:8080 login as demo|demo, use SQL Command)**



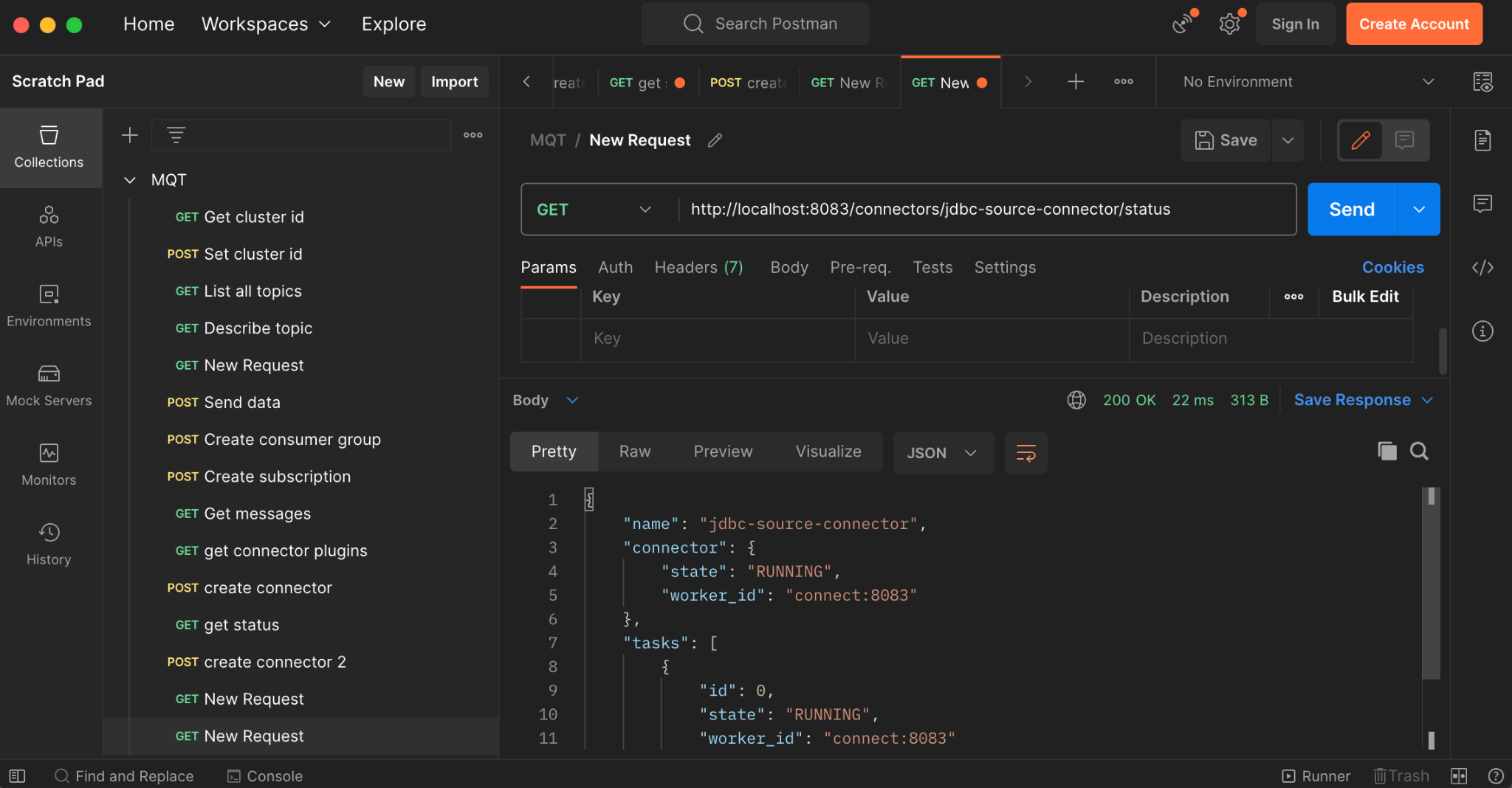
**5. Check messages**



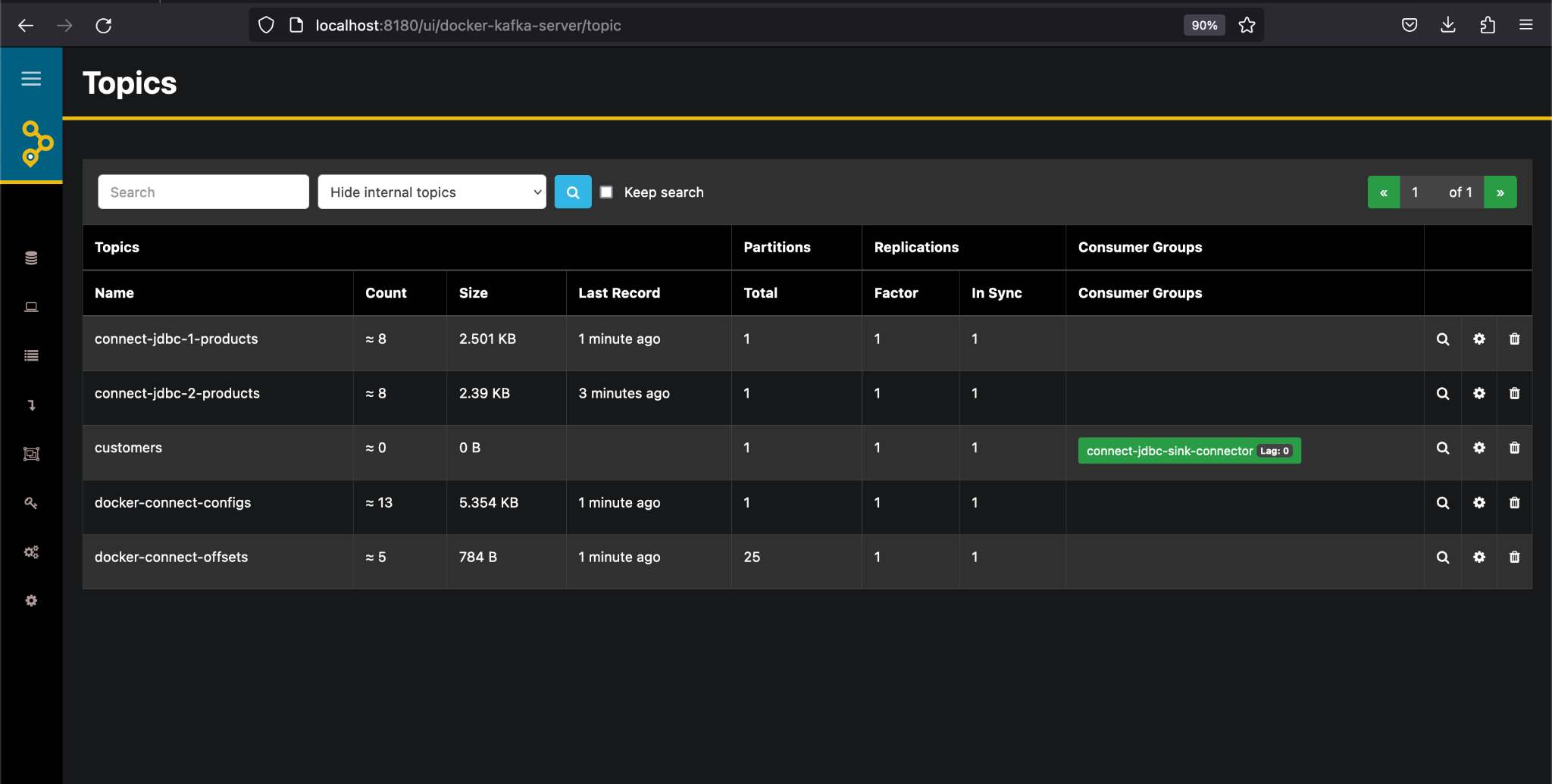
**6. Resume connector**



**7. Check status**

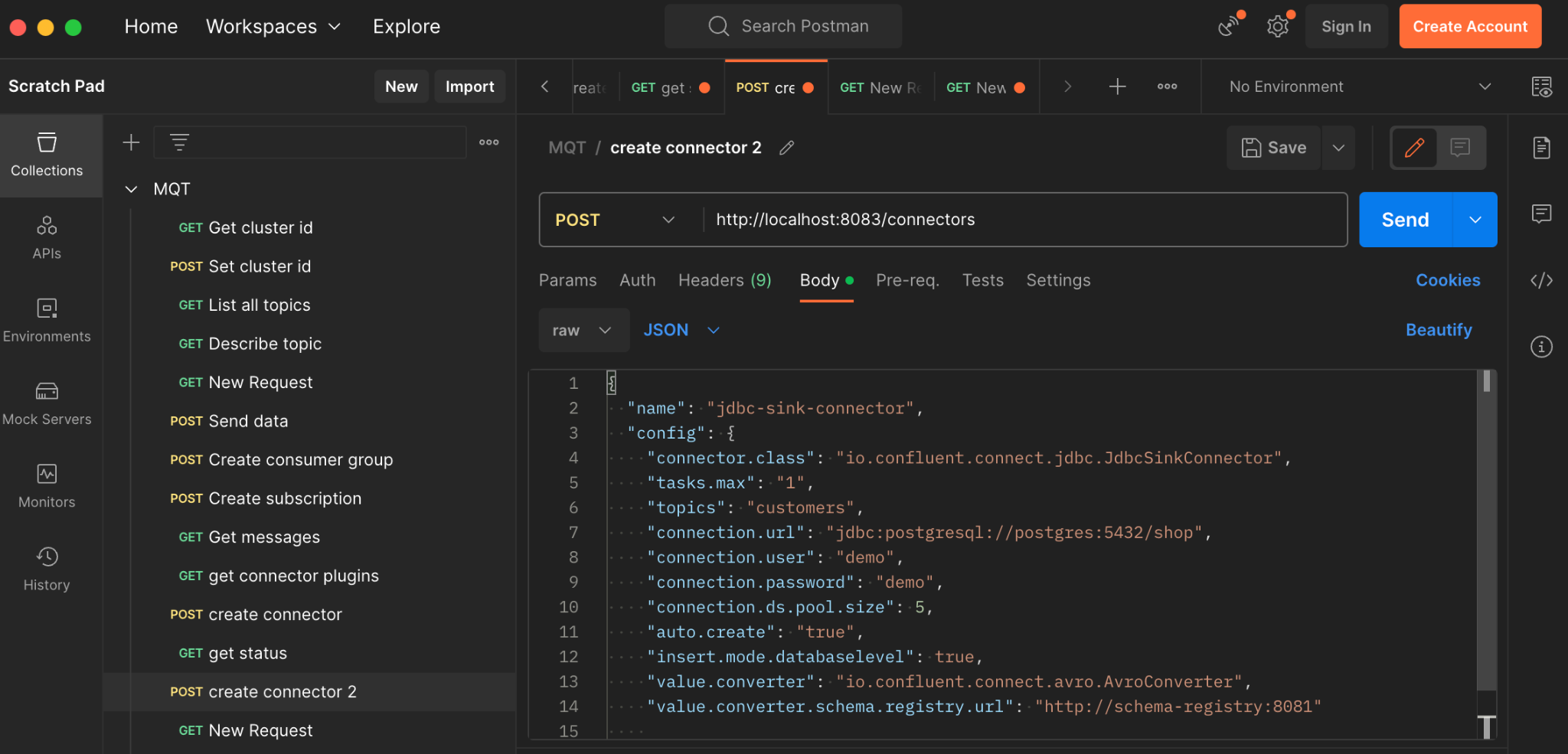


**8. Check messages**



**Lab 3:**

**1. In Control Center create JDBC sink connector. The configuration is in file postgres-sink-connector-1.json**



**2. In schema registry container produce avro messages**

docker exec -ti schema-registry /usr/bin/kafka-avro-console-producer --bootstrap-server kafka:19092 --topic customers --property schema.registry.url=http://schema-registry:8081 --property value.schema="{\"name\":\"lau\",\"type\":\"record\",\"name\":\"customer\",\"fields\":[{\"name\":\"id\",\"type\":\"int\"},{\"name\":\"name\",\"type\":\"string\"}]}"

the command can also be executed from Docker Desktop, schema-registry terminal:

/usr/bin/kafka-avro-console-producer --bootstrap-server kafka:19092 --topic customers2 --property schema.registry.url=http://schema-registry:8081 --property value.schema="{\"name\":\"lau\",\"type\":\"record\",\"name\":\"customer\",\"fields\":[{\"name\":\"id\",\"type\":\"int\"},{\"name\":\"name\",\"type\":\"string\"}]}"

add in the producer console next lines:

{"id": 1, "name": "Jane Doe"}

{"id": 2, "name": "John Smith"}

{"id": 3, "name": "Ann Black"}

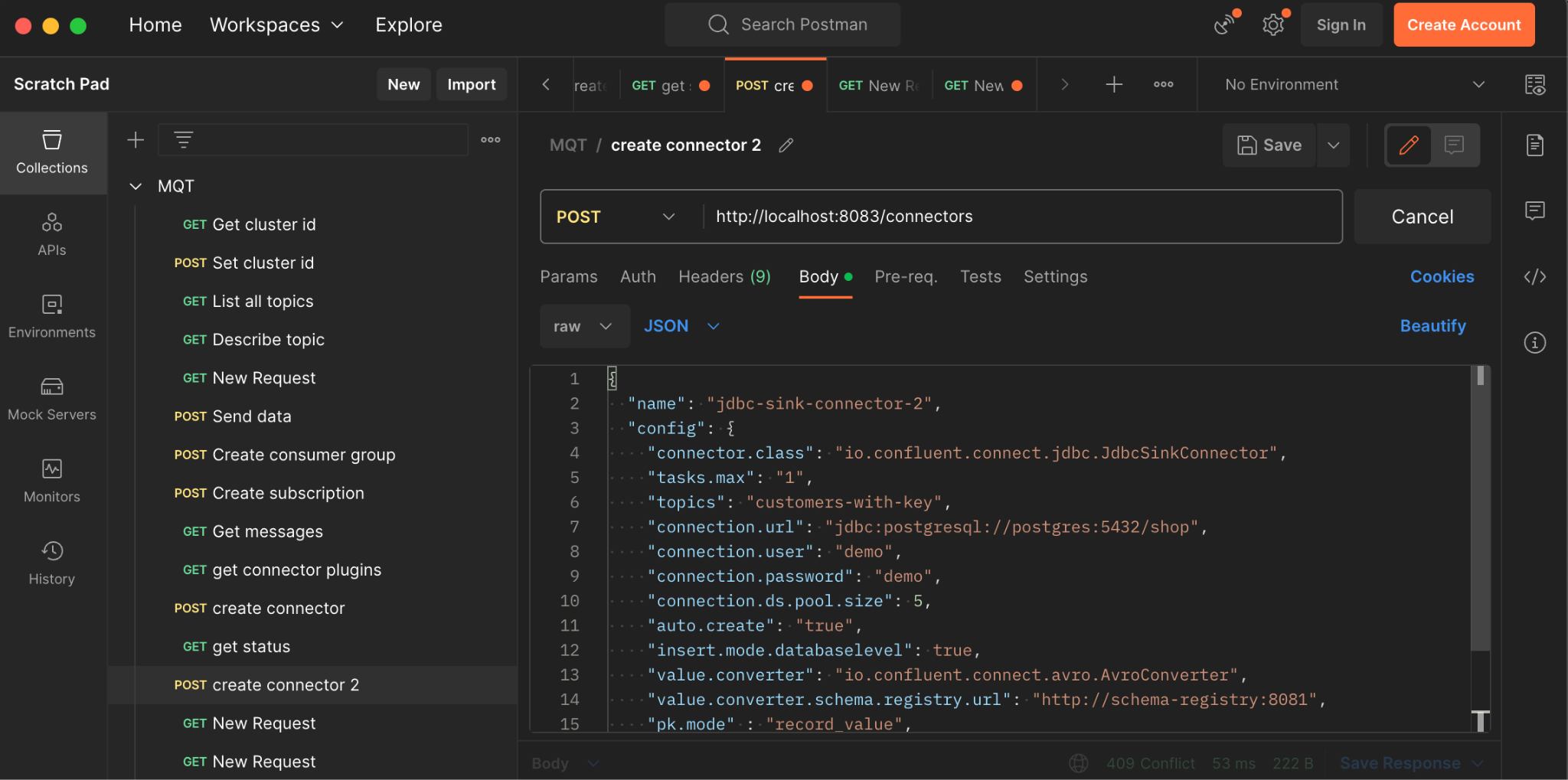
**3. Check the table in http://localhost:8080**

A screenshot of a computer

Description automatically generated

**Lab 4:**

**1. In Control Center create JDBC sink connector. The configuration is in file postgres-sink-connector-2.json**

****

**2. In schema registry container produce avro messages**

MERGE:

docker exec -ti schema-registry /usr/bin/kafka-avro-console-producer --bootstrap-server kafka:19092 --topic customers-with-key --property schema.registry.url=http://schema-registry:8081 --property value.schema="{\"type\":\"record\",\"name\":\"customer\",\"fields\":[{\"name\":\"id\",\"type\":\"int\"},{\"name\":\"name\",\"type\":\"string\"}]}"

-- fara \" --> NU merge

docker exec -ti schema-registry /usr/bin/kafka-avro-console-producer --bootstrap-server kafka:19092 --topic customers-with-key --property schema.registry.url=http://schema-registry:8081 --property value.schema="{"type":"record","name":"customer","fields":[{"name":"id","type":"int"},{"name":"name","type":"string"}]}"

{"id": 1, "name": "Janett Falow"}

{"id": 2, "name": "John Cache"}

{"id": 3, "name": "Black Beauty"}

**3. Check the table in http://localhost:8080**

A screenshot of a computer

Description automatically generated