

Laboratory 7

Lab1:

1. Login into admin application <http://localhost:8080>

2. Login as

System: PostgreSQL

Server: postgres

Username: demo

Password: demo

Database: shop

The screenshot shows a web browser window with three tabs open. The active tab is 'Login - Adminer' at 'localhost:8080'. The browser's address bar also displays 'localhost:8080'. The Adminer interface has a sidebar on the left with the title 'Adminer 4.8.1' and a main 'Login' panel on the right. In the 'Login' panel, there is a form with the following fields:

System	PostgreSQL
Server	postgres
Username	demo
Password	****
Database	shop

Below the form are two buttons: 'Login' and 'Permanent login' (with a checked checkbox).

2. Create a database (SQL command)

Press "SQL command" button from left side.

```
CREATE TABLE IF NOT EXISTS products (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    price INT
);
```

← → ⌂ localhost:8080/?pgsql=postgres&username=demo&db=shop&ns=public&sql=CREATE%20TABLE%20IF%20NOT%20EXISTS%

Language: English PostgreSQL » postgres » shop » public » SQL command

Adminer 4.8.1

DB: shop Schema: public

SQL command Import
Export Create table

select customers
select customers-with-key
select products

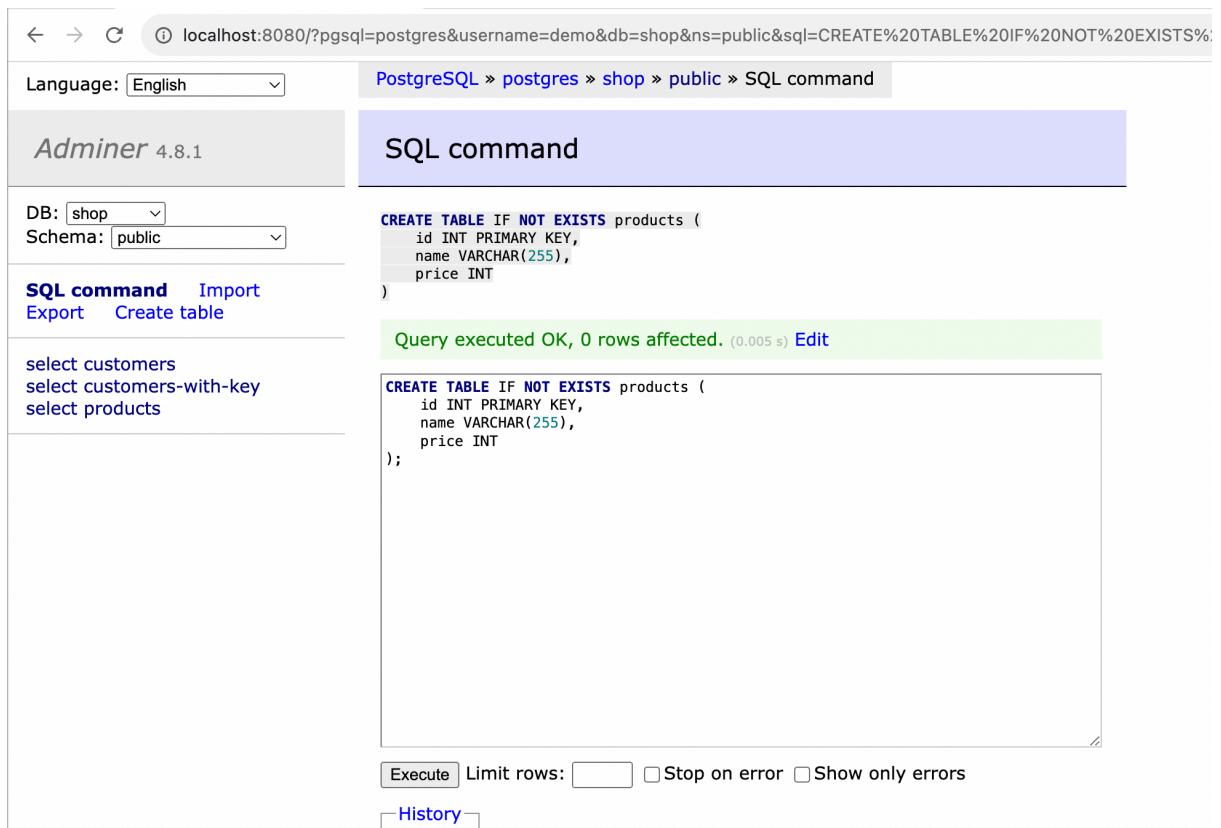
CREATE TABLE IF NOT EXISTS products (
 id INT PRIMARY KEY,
 name VARCHAR(255),
 price INT
)

Query executed OK, 0 rows affected. (0.005 s) Edit

CREATE TABLE IF NOT EXISTS products (
 id INT PRIMARY KEY,
 name VARCHAR(255),
 price INT
)

Execute Limit rows: Stop on error Show only errors

History



3. Enter SQL Command

```
insert into products values(1, 'cup', 1);
insert into products values(2, 'kettle', 5);
insert into products values(3, 'phone', 500);
insert into products values(4, 'tablet', 125);
```

← → ⌂ ⓘ localhost:8080/?pgsql=postgres&username=demo&db=shop&ns=public&sql=%20%20insert%20into%20products%20values(1%2C%2

Language: English

PostgreSQL » postgres » shop » public » SQL command

Adminer 4.8.1

DB: shop Schema: public

SQL command Import
Export Create table

select customers
select customers-with-key
select products

insert into products values(1, 'cup', 1)
Query executed OK, 1 row affected. (0.001 s) Edit

insert into products values(2, 'kettle', 5)
Query executed OK, 1 row affected. (0.001 s) Edit

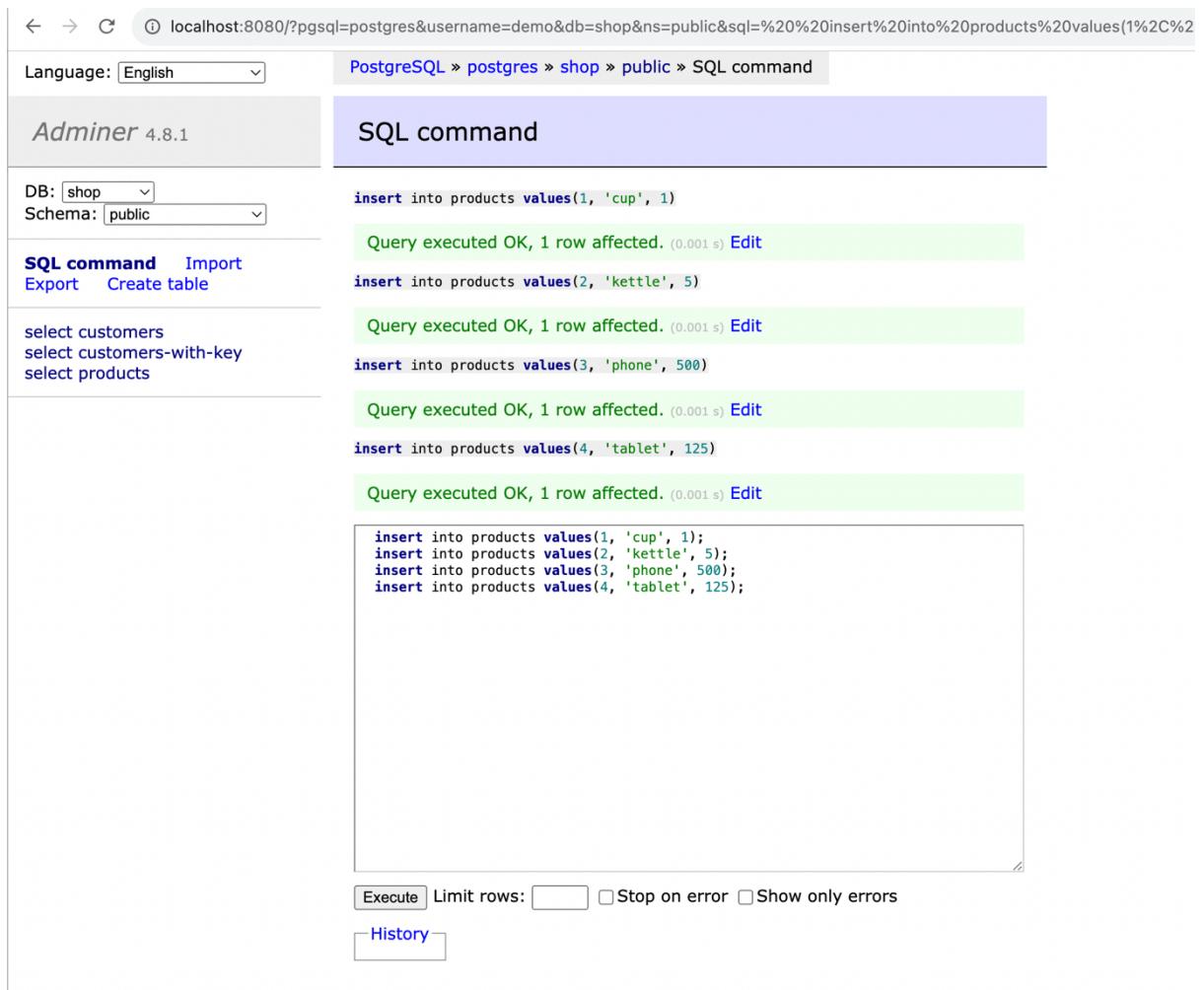
insert into products values(3, 'phone', 500)
Query executed OK, 1 row affected. (0.001 s) Edit

insert into products values(4, 'tablet', 125)
Query executed OK, 1 row affected. (0.001 s) Edit

insert into products values(1, 'cup', 1);
insert into products values(2, 'kettle', 5);
insert into products values(3, 'phone', 500);
insert into products values(4, 'tablet', 125);

Execute Limit rows: Stop on error Show only errors

History



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

The screenshot shows the Postman interface with a collection named "nike_cms". A POST request is being made to "http://localhost:8083/connectors" to create a connector named "jdbc-source-connector". The request body contains the following JSON configuration:

```

1  {
2     "name": "jdbc-source-connector",
3     "config": {
4         "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
5         "tasks.max": 1,
6         "topic.prefix": "connect-jdbc-1",
7         "connection.url": "jdbc:postgresql://postgres:5432/shop",
8         "mode": "incrementing",
9         "incrementing.column.name": "id",
10        "value.converter": "org.apache.kafka.connect.json.JsonConverter",
11        "value.converter.schemas.enable": "true",
12        "table.whitelist": "public.products",
13        "connection.user": "demo",
14        "connection.password": "demo"
15    }
16 }

```

The response status is 201 Created, and the time taken is 571 ms.

6. Check connectors through Control center or through REST

The screenshot shows the Postman interface with the same collection "nike_cms". A GET request is being made to "http://localhost:8083/connectors/jdbc-source-connector/status" to check the status of the connector. The response status is 200 OK, and the time taken is 27 ms. The response body is as follows:

```

1  {
2     "name": "jdbc-source-connector",
3     "connector": {
4         "state": "RUNNING",
5         "worker_id": "connect:8083"
6     },
7     "tasks": [
8         {
9             "id": 0,
10            "state": "RUNNING",
11            "worker_id": "connect:8083"
12        }
13    ],
14    "type": "source"
15 }

```

7. Check messages

Topics				Partitions	Replications		Consumer Groups
Name	Count	Size	Last Record	Total	Factor	In Sync	
connect-jdbc-1-products	≈ 8	2.382 KB	1 hour ago	1	1	1	
connect-jdbc-2-products	≈ 8	2.39 KB	1 hour ago	1	1	1	
customers	≈ 3	252 B	57 minutes ago	1	1	1	connect-jdbc-sink-connector Lag: 0
customers-with-key	≈ 3	259 B	49 minutes ago	1	1	1	connect-jdbc-sink-connector-2 Lag: 0
docker-connect-configs	≈ 22	9.824 KB	17 minutes ago	1	1	1	
docker-connect-offsets	≈ 4	628 B	1 hour ago	25	1	1	

8. Add more data

```
insert into products values(5, 'toaster', 15);
insert into products values(6, 'coffee maker', 25);
```

DETAIL: KEY (10)=(4) already exists.

```
insert into products values(5, 'toaster', 15)
```

Query executed OK, 1 row affected. (0.001 s) [Edit](#)

```
insert into products values(6, 'coffee maker', 25)
```

Query executed OK, 1 row affected. (0.001 s) [Edit](#)

Error in query: 1 2 3 4

```
insert into products values(1, 'cup', 1);
insert into products values(2, 'kettle', 5);
insert into products values(3, 'phone', 500);
insert into products values(4, 'tablet', 125);

insert into products values(5, 'toaster', 15);
insert into products values(6, 'coffee maker', 25);
```

[Execute](#) Limit rows: Stop on error Show only errors

...

9. Check messages

Lab2:

1. Add JDBC source connector

The screenshot shows the Postman application interface. In the left sidebar, there's a collection named "nike-cms" containing several requests. One request, "POST Create new topic", is highlighted. The main workspace shows a "New Request" dialog for a "POST" method to "http://localhost:8083/connectors". The "Body" tab is selected, displaying a JSON configuration for a JDBC source connector. The response status is "201 Created" with a response time of 145 ms and a size of 112 KB.

```

1 {
2   "name": "jdbc-source-connector-2",
3   "config": {
4     "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
5     "tasks.max": "1",
6     "topic.prefix": "connect-jdbc-2-",
7     "connection.url": "jdbc:postgresql://postgres:5432/shop",
8     "mode": "incrementing",
9     "incrementing.column.name": "id",
10    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
11    "value.converter.schemas.enable": "true",
12    "table.whitelist": "public.products",
13    "connection.user": "demo",
14    "connection.password": "demo",
15    "transforms.createkey.type": "org.apache.kafka.connect.transforms.ValueToKey",
16    "transforms.createkey.fields": "id",
17    "transforms.extractInt.type": "org.apache.kafka.connect.transforms.ExtractField$Key",
18    "transforms.extractInt.fields": "id",
19    "transforms.RenameField.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
20    "transforms.RenameField.renames": "name:code"
21  }
22

```

2. Stop connector

The screenshot shows the Postman application interface. In the left sidebar, there's a collection named "nike-cms" containing several requests. One request, "PUT Stop connector", is highlighted. The main workspace shows a "Stop connector" dialog for a "PUT" method to "localhost:8083/connectors/jdbc-source-connector/pause". The "Body" tab is selected, showing a single parameter "Key" with the value "Key". The response status is "202 Accepted" with a response time of 22 ms and a size of 114 B.

Key	Value	Description
Key		Description

3. Check status

The screenshot shows the Postman application interface. On the left, there's a sidebar with collections like 'nike-cms' and 'mqqt-lab7'. Under 'mqqt-lab7', there are several requests: 'Get clusters ids', 'Create new topic', 'Get all topics', 'Describe topic', 'Send data', 'Add consumer group', 'Create subscription', 'Get messages', 'New Request', 'New Request', 'Stop connector', and 'Check status'. The 'Check status' request is selected. The main panel shows the request details: method 'GET', URL 'localhost:8083/connectors/jdbc-source-connector/status', and a table for 'Query Params'. The response tab shows a JSON response with fields like 'name', 'connector', 'state', 'worker_id', 'tasks', and 'type'. The status bar at the bottom indicates 'Status: 200 OK'.

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

```
insert into products values(7, 'saucepan', 11);
insert into products values(8, 'iron', 7);
```

The screenshot shows a database query interface. It displays two successful insert operations:

```
insert into products values(7, 'saucepan', 11)
Query executed OK, 1 row affected. (0.001 s) Edit, Warnings
```

```
insert into products values(8, 'iron', 7)
Query executed OK, 1 row affected. (0.001 s) Edit, Warnings
```

Below these, there is an error message:

Error in query: 2 3 4 5 6 7

Finally, there is a large block of SQL code:

```
CREATE TABLE IF NOT EXISTS products (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    price INT
);

insert into products values(1, 'cup', 1);
insert into products values(2, 'kettle', 5);
insert into products values(3, 'phone', 500);
insert into products values(4, 'tablet', 125);

insert into products values(5, 'toaster', 15);
insert into products values(6, 'coffee maker', 25);

insert into products values(7, 'saucepan', 11);
insert into products values(8, 'iron', 7);
```

5. Check messages

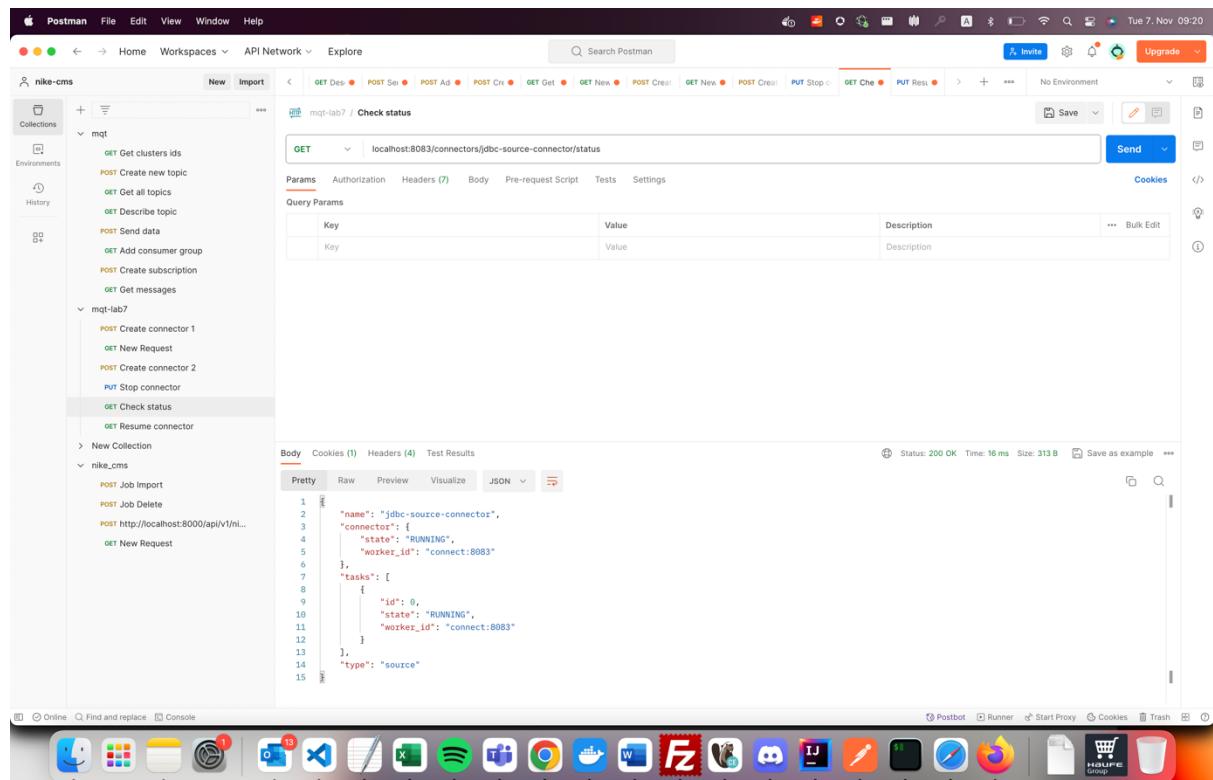
The screenshot shows the Apache Kafka UI interface. The main title bar says "Topics". Below it is a search bar with "Search" and "Keep search" options. A table lists four topics: "connect-jdbc-1-products", "connect-jdbc-2-products", "docker-connect-configs", and "docker-connect-offsets". Each topic row includes columns for Name, Count, Size, Last Record, Partitions, Factor, In Sync, and Consumer Groups. At the bottom right of the table is a "Create a topic" button.

Name	Count	Size	Last Record	Partitions	Factor	In Sync	Consumer Groups
connect-jdbc-1-products	≈ 6	1.729 KB	11 minutes ago	1	1	1	
connect-jdbc-2-products	≈ 8	2.39 KB	43 seconds ago	1	1	1	
docker-connect-configs	≈ 10	4.528 KB	5 minutes ago	1	1	1	
docker-connect-offsets	≈ 3	472 B	19 seconds ago	25	1	1	

6. Resume connector

The screenshot shows the Postman application interface. On the left, there's a sidebar with collections like "nike-cms" and "mqqt-lab7". Under "mqqt-lab7", there are several API endpoints listed under the "Resume connector" section. One endpoint is highlighted: "PUT localhost:8083/connectors/jdbc-source-connector/resume". The "Params" tab is selected, showing a "Key" parameter with a value of "Value". The "Body" tab shows a simple JSON body: {"key": "Value"}. The status bar at the bottom indicates a successful response: "Status: 202 Accepted Time: 30 ms Size: 114 B".

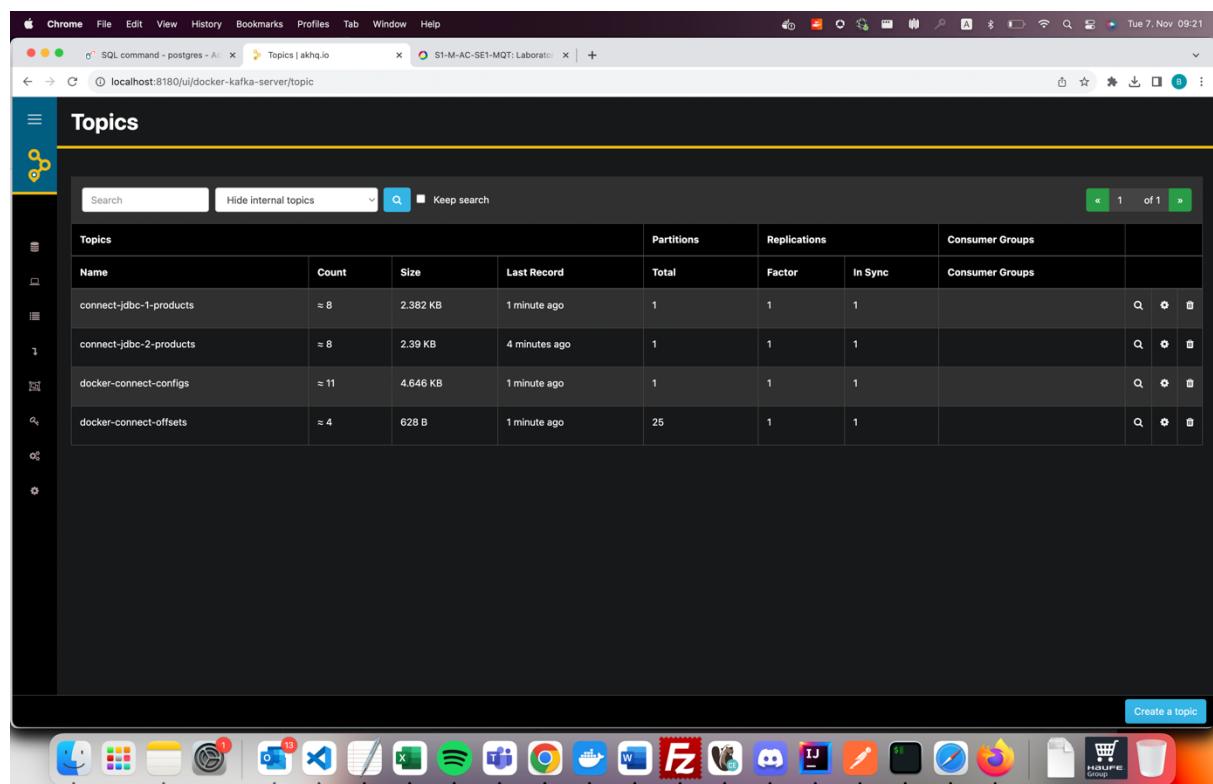
7. Check status



The screenshot shows the Postman application interface. On the left, there's a sidebar with collections like 'nike-cms' and 'mqt-lab'. The main area shows a request to 'localhost:8083/connectors/jdbc-source-connector/status'. The 'Body' tab displays the following JSON response:

```
1 "name": "jdbc-source-connector",
2   "connector": {
3     "state": "RUNNING",
4     "worker_id": "connect:8083"
5   },
6   "tasks": [
7     {
8       "id": 0,
9       "state": "RUNNING",
10      "worker_id": "connect:8083"
11    }
12  ],
13  "type": "source"
14
15
```

8. Check messages



The screenshot shows the Apache Kafka UI 'Topics' page. It lists the following topics:

Name	Count	Size	Last Record	Total	Factor	In Sync	Consumer Groups
connect-jdbc-1-products	≈ 8	2.382 KB	1 minute ago	1	1	1	
connect-jdbc-2-products	≈ 8	2.39 KB	4 minutes ago	1	1	1	
docker-connect-configs	≈ 11	4.646 KB	1 minute ago	1	1	1	
docker-connect-offsets	≈ 4	628 B	1 minute ago	25	1	1	

Lab3:

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

The screenshot shows the Postman application interface. The left sidebar has a tree view with collections: 'nike-cms' (selected), 'mqt', and 'mqt-lab7'. Under 'nike-cms', there are several API endpoints like 'Get clusters ids', 'Create new topic', etc. Under 'mqt-lab7', there are endpoints for creating connectors. The main workspace shows a POST request to 'http://localhost:8083/connectors' with the following JSON body:

```
POST /connectors
{
  "name": "jdbc-sink-connector",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
    "tasks.max": "1",
    "topics": "customers",
    "connection.url": "jdbc:postgresql://postgres:5432/shop",
    "connection.user": "demo",
    "connection.password": "demo",
    "connection.ds.pool.size": 5,
    "auto.create": "true",
    "insert.mode.databaselevel": true,
    "value.converter": "io.confluent.connect.avro.AvroConverter",
    "value.converter.schema.registry.url": "http://schema-registry:8081"
  }
}
```

The status bar at the bottom indicates 'Status: 201 Created'.

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"}
```

Docker Desktop Update to latest

Containers Images Volumes Dev Environments BETA

Add Extensions

schema-registry confluentinc/cp-schema-registry:7.0.1 RUNNING

New integrated terminal
To set your external terminal as the default, change your [preferences](#).

Open in external terminal

```
sh-4.4$ /usr/bin/kafka-avro-console-producer --bootstrap-server kafka:19092 --topic customers --property schema.registry.url=http://schema-registry:8081 --property value.schema='{"name": "id", "type": "int"}, {"name": "name", "type": "string"}'
[2023-11-07 07:25:02,027] INFO Registered KafkaType=kafka.Log4jController$KafkaLog4jController (org.apache.kafka.clients.producer.internals.ConsumerConfig)
[2023-11-07 07:25:03,469] INFO ProducerConfig values:
acks = 1
batch.size = 16384
bootstrap.servers = [kafka:19092]
buffer.memory = 33554432
client.dns.lookup = use_all_dns_ips
client.id = schema-registry
compression.type = none
connections.max.idle.ms = 40000
delays.max.total.ms = 120000
enable.idempotence = true
interceptor.classes = []
key.serializer = class org.apache.kafka.common.serialization.ByteArraySerializer
linger.ms = 1000
max.block.ms = 60000
max.in.flight.requests.per.connection = 5
max.lag.check.interval.ms = 1048576
metadata.max.age.ms = 300000
metadata.max.idle.ms = 300000
metrics.num.samples = 2
metrics.recording.level = INFO
metric.reporters = []
partitions.max.in.fetch = 1000
partitioner.class = class org.apache.kafka.clients.producer.internals.DefaultPartitioner
receive.buffer.bytes = 32768
reconnect.backoff.max.ms = 1000
reconnect.backoff.ms = 100
request.timeout.ms = 150
retries = 1
retry.backoff.ms = 100
sasl.client.callback.handler.class = null
sasl.jaas.config = null
sasl.kerberos.principal = /usr/bin/kinit
sasl.kerberos.min.time.before.relogin = 60000
sasl.kerberos.service.name = null
sasl.kerberos.ticket.renew.jitter = 0.05
sasl.kerberos.ticket.renew.window.factor = 0.8
sasl.login.callback.handler.class = null
sasl.login.class = null
sasl.login.refresh.max.period.seconds = 300
sasl.login.refresh.min.period.seconds = 60
sasl.login.refresh.window.factor = 0.8
sasl.login.refresh.window.jitter = 0.05
sasl.mechanism = GSSAPI
security.protocol = PLAINTEXT
security.providers = null
send.buffer.bytes = 162400
socket.connection.setup.timeout.max.ms = 30000
socket.connection.setup.timeout.ms = 10000
ssl.cipher.suites = null
ssl.enabled.protocols = [TLSv1.2, TLSv1.3]
ssl.endpoint.identification.algorithm = https
ssl.keystore.location = null
ssl.key.password = null
ssl.keymanager.algorithm = SunX509
```

RAM 4.43GB CPU 22.85% Not connected to Hub

Docker Desktop Update to latest

Containers Images Volumes Dev Environments BETA

Add Extensions

schema-registry confluentinc/cp-schema-registry:7.0.1 RUNNING

New integrated terminal
To set your external terminal as the default, change your [preferences](#).

Open in external terminal

```
sh-4.4$ /usr/bin/kafka-avro-console-producer --bootstrap-server kafka:19092 --topic customers --property schema.registry.url=http://schema-registry:8081 --property value.schema='{"name": "id", "type": "int"}, {"name": "name", "type": "string"}'
[2023-11-07 07:25:04,123] INFO Kafka version: 7.0.1-ce (org.apache.kafka.common.utils.AppInfoParser)
[2023-11-07 07:25:04,123] INFO Kafka commitId: 165b634a62c986 (org.apache.kafka.common.utils.AppInfoParser)
[2023-11-07 07:25:04,124] INFO Kafka startTimestamps: 1699341904067 (org.apache.kafka.common.utils.AppInfoParser)
[2023-11-07 07:25:04,124] INFO Kafka startOffset: 0 (Producer clientId=console-producer) Cluster ID: 8g9efxq0N2X0epN1CJ7Ow (org.apache.kafka.clients.Metadata)
{"id": 1, "name": "Jane Doe"}, {"id": 2, "name": "John Smith"}, {"id": 3, "name": "Ana Black"}
```

RAM 4.43GB CPU 25.09% Not connected to Hub

3. Check the table in <http://localhost:8080>

Language: English

PostgreSQL » postgres » shop » public » Select: customers

Adminer 4.8.1

DB: shop Schema: public

SQL command Import Export Create table

Select: customers

Select data Show structure Alter table New item

Select Search Sort Limit 50 Text length 100 Action Select

```
SELECT * FROM "customers" LIMIT 50 (0.000 s) Edit
```

<input type="checkbox"/> Modify	id	name
<input type="checkbox"/> edit	1	Jane Doe
<input type="checkbox"/> edit	2	John Smith
<input type="checkbox"/> edit	3	Ann Black

Whole result 3 rows Modify Save Selected (0) Edit Clone Delete Export (3)

Import

Lab4:

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

Postman

POST Ad POST Ctr GET Get GET New POST Creat GET New POST Creat PUT Stop GET Che PUT Rest POST Creat POST Ctr

mgt-lab7 / Create connector sink 2

POST http://localhost:8083/connectors

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

4     "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
5     "tasks.max": "1",
6     "topics": "customers-with-key",
7     "connection.url": "jdbc:postgresql://postgres:5432/shop",
8     "connection.user": "shop",
9     "connection.password": "demo",
10    "connection.ds.pool.size": 5,
11    "auto.create": "true",
12    "insert.mode.databaselevel": true,
13    "value.converter": "io.confluent.connect.avro.AvroConverter",
14    "value.converter.schema.registry.url": "http://schema-registry:8081",
15    "pk.mode": "record_value",
16    "pk.fields": "id"
17
18
19

```

Body Cookies (1) Headers (5) Test Results

Status: 201 Created Time: 113 ms Size: 788 B Save as example

Online Find and replace Console

2. In schema registry container produce avro messages

MERGE:

```
docker exec -ti schema-registry /usr/bin/kafka-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"}]}'
```

The screenshot shows the Docker Desktop interface with a terminal window open for the 'schema-registry' container. The terminal output displays Java stack traces and configuration logs related to Kafka's JSON deserialization and producer configuration.

```
Open in external terminal
clear
org.apache.kafka.common.errors.SerializationException: Error deserializing json clear to Avro of schema {"type": "record", "name": "customer", "fields": [{"name": "id", "type": "int"}, {"name": "name", "type": "string"}]}
    at io.confluent.kafka.formatter.AvroMessageHeader.readFrom(AvroMessageHeader.java:131)
    at io.confluent.kafka.formatter.SchemaMessageHeader.readFrom(SchemaMessageHeader.java:325)
    at kafka.tools.ConsoleProducer.main(ConsoleProducer.scala:51)
    at kafka.tools.ConsoleProducer$.main(ConsoleProducer.scala:51)
Caused by: com.fasterxml.jackson.core.JsonParseException: Unrecognized token 'clear': was expecting JSON String, Number, Array, Object or token 'null', 'true' or 'false'
    at [Source: (byte[])@1000000000000000; line: 1, column: 1]
    at com.fasterxml.jackson.core.JsonParser._constructError(JsonParser.java:2337)
    at com.fasterxml.jackson.core.base.ParserMinimalBase._reportError(ParserMinimalBase.java:170)
    at com.fasterxml.jackson.core.base.ParserMinimalBase._reportError(ParserMinimalBase.java:170)
    at com.fasterxml.jackson.core.json.ReaderBasedJsonParser.nextToken(ReaderBasedJsonParser.java:781)
    at com.fasterxml.jackson.core.json.ReaderBasedJsonParser.nextToken(ReaderBasedJsonParser.java:1949)
    at org.apache.avro.io.JsonDecoder.<init>(JsonDecoder.java:124)
    at org.apache.avro.io.JsonDecoder.<init>(JsonDecoder.java:46)
    at org.apache.avro.io.JsonDecoder.<init>(JsonDecoder.java:74)
    at org.apache.avro.io.DecoderFactory.jsonDecoder(DecoderFactory.java:264)
    at io.confluent.kafka.formatter.AvroMessageHeader.readFrom(AvroMessageHeader.java:214)
    at io.confluent.kafka.formatter.AvroMessageHeader.readFrom(AvroMessageHeader.java:124)
    ...
3 more
[2023-11-07 07:31:51.876] INFO [Producer clientId=console-producer] Closing the Kafka producer with timeoutMillis = 922372036854775807 ms. (org.apache.kafka.clients.producer.KafkaProducer)
[2023-11-07 07:31:51.946] INFO Metrics scheduler closed (org.apache.kafka.common.metrics.Metrics)
[2023-11-07 07:31:51.948] INFO Closing reporter org.apache.kafka.common.metrics.JmxReporter (org.apache.kafka.common.metrics.Metrics)
[2023-11-07 07:31:51.949] INFO [Producer clientId=console-producer] Shutting down (org.apache.kafka.clients.producer.KafkaProducer)
[2023-11-07 07:31:51.969] INFO App info kafka.producer for console-producer unregistered (org.apache.kafka.common.utils.AppInfoParser)
[2023-11-07 07:32:19.832] INFO Registered kafka:type=kafka.Log4jController Rmean (kafka.utils.Log4jControllerRegistration)
[2023-11-07 07:32:19.832] INFO ProducerConfig values:
acks = 1
batch.size = 16384
buffer.memory = 33554432
client.dns.lookup = use_all_dns_ips
client.id = console-producer
compression.type = none
connections.max.idle.ms = 540000
delays.max.lifetime.ms = 10000
enable.idempotence = true
interceptor.classes = []
key.serializer = class org.apache.kafka.common.serialization.ByteArraySerializer
linger.ms = 1000
max.block.ms = 60000
max.in.flight.requests.per.connection = 5
max.message.bytes = 1048576
metadata.max.age.ms = 300000
metadata.max.idle.ms = 300000
metrics.num.samples = 2
metrics.recording.level = INFO
metrics.sample.window.ms = 30000
partitions.max.in.class = class org.apache.kafka.clients.producer.internals.DefaultPartitioner
receive.buffer.bytes = 32768
reconnect.backoff.ms = 1000
reconnect.backoff.ms = 1000
request.timeout.ms = 1500
retries = 3
retry.backoff.ms = 100
[2023-11-07 07:32:19.841] INFO [Producer clientId=console-producer] Closing the Kafka producer with timeoutMillis = 922372036854775807 ms. (org.apache.kafka.clients.producer.KafkaProducer)
[2023-11-07 07:32:19.841] INFO Metrics scheduler closed (org.apache.kafka.common.metrics.Metrics)
[2023-11-07 07:32:19.842] INFO Closing reporter org.apache.kafka.common.metrics.JmxReporter (org.apache.kafka.common.metrics.Metrics)
[2023-11-07 07:32:19.843] INFO [Producer clientId=console-producer] Shutting down (org.apache.kafka.clients.producer.KafkaProducer)
[2023-11-07 07:32:19.843] INFO Registered kafka:type=kafka.Log4jController Rmean (kafka.utils.Log4jControllerRegistration)
```

RAM: 4.11GB CPU: 22.70% Not connected to Hub v4.12.0

```
{"id": 1, "name": "Janett Falow"}
{"id": 2, "name": "John Cache"}
{"id": 3, "name": "Black Beauty"}
```

```

metrics.sample.window.ms = 30000
partitioner.class = class org.apache.kafka.clients.producer.internals.DefaultPartitioner
receive.buffer.bytes = 32768
reconnect.backoff.ms = 1000
reconnect.backoff.ms = 50
request.timeout.ms = 1500
retry.backoff.ms = 100
ssl.client.callback.handler.class = null
ssl.jaas.config = null
ssl.keystore.location = /usr/bin/kinit
ssl.kerberos.min.time.before.relogin = 60000
ssl.kerberos.service.name = null
ssl.login.callback.handler.class = null
ssl.login.refresh.window.factor = 0.05
ssl.login.refresh.window.factor = 0.8
ssl.login.callback.handler.class = null
ssl.login.refresh.buffer.seconds = 300
ssl.login.refresh.min.period.seconds = 60
ssl.login.refresh.window.factor = 0.8
ssl.login.refresh.window.litter = 0.05
ssl.mechanism = GSSAPI
security.protocol = PLAINTEXT
secure.random.implementation = null
send.buffer.bytes = 102400
socket.connection.setup.timeout.max.ms = 30000
socket.receive.timeout.ms = 10000
ssl.cipher.suites = null
ssl.enabled.protocols = [TLSv1.2, TLSv1.3]
ssl.engine.factory.class = null
ssl.key.password = null
ssl.keymanager.algorithm = SunX509
ssl.keystores.truststore.chain = null
ssl.keystores.key = null
ssl.keystores.location = null
ssl.keystores.type = PKCS12
ssl.keystores.truststore.location = null
ssl.keystores.truststore.password = null
ssl.keystores.type = JKS
ssl.minimum.version = 1.2
ssl.secure.random.implementation = null
ssl.trustmanager.algorithm = PKIX
ssl.trustmanager.certificates = null
ssl.truststores.location = null
ssl.truststores.type = JKS
ssl.transactional.id = null
transactional.id = null
value.serializer = class org.apache.kafka.common.serialization.ByteArraySerializer
(org.apache.kafka.clients.producer.ProducerConfig)
[2023-11-07 07:13:21,822] INFO Kafka version: 7.0.1-ce (org.apache.kafka.common.utils.AppInfoParser)
[2023-11-07 07:13:21,823] INFO Kafka commitId: 1fd5b634a62c986 (org.apache.kafka.common.utils.AppInfoParser)
[2023-11-07 07:13:21,823] INFO Kafka startTimestampMs: 169934224171 (org.apache.kafka.common.utils.AppInfoParser)
[2023-11-07 07:13:23,861] INFO [Producer clientId=console-producer] Cluster ID: 8ghpeqgD2X0egMTC3YQw (org.apache.kafka.clients.Metadata)
{"id": 1, "name": "Janett Falow"}
{"id": 2, "name": "John Cache"}
{"id": 3, "name": "Black Beauty"}

```

RAM 4.07GB CPU 21.40% Not connected to Hub v4.12.0

3. Check the table in <http://localhost:8080>

Language: English

PostgreSQL » postgres » shop » public » Select: customers-with-key

Adminer 4.8.1

DB: shop Schema: public

Select data Show structure Alter table New item

Select Search Sort Limit 50 Text length 100 Action Select

SELECT * FROM "customers-with-key" LIMIT 50 (0.001 s) Edit

<input type="checkbox"/> Modify	id	name
<input type="checkbox"/> edit	1	Janett Falow
<input type="checkbox"/> edit	2	John Cache
<input type="checkbox"/> edit	3	Black Beauty

Whole result 3 rows Modify Save Selected (0) Edit Clone Delete Export (3)

Import