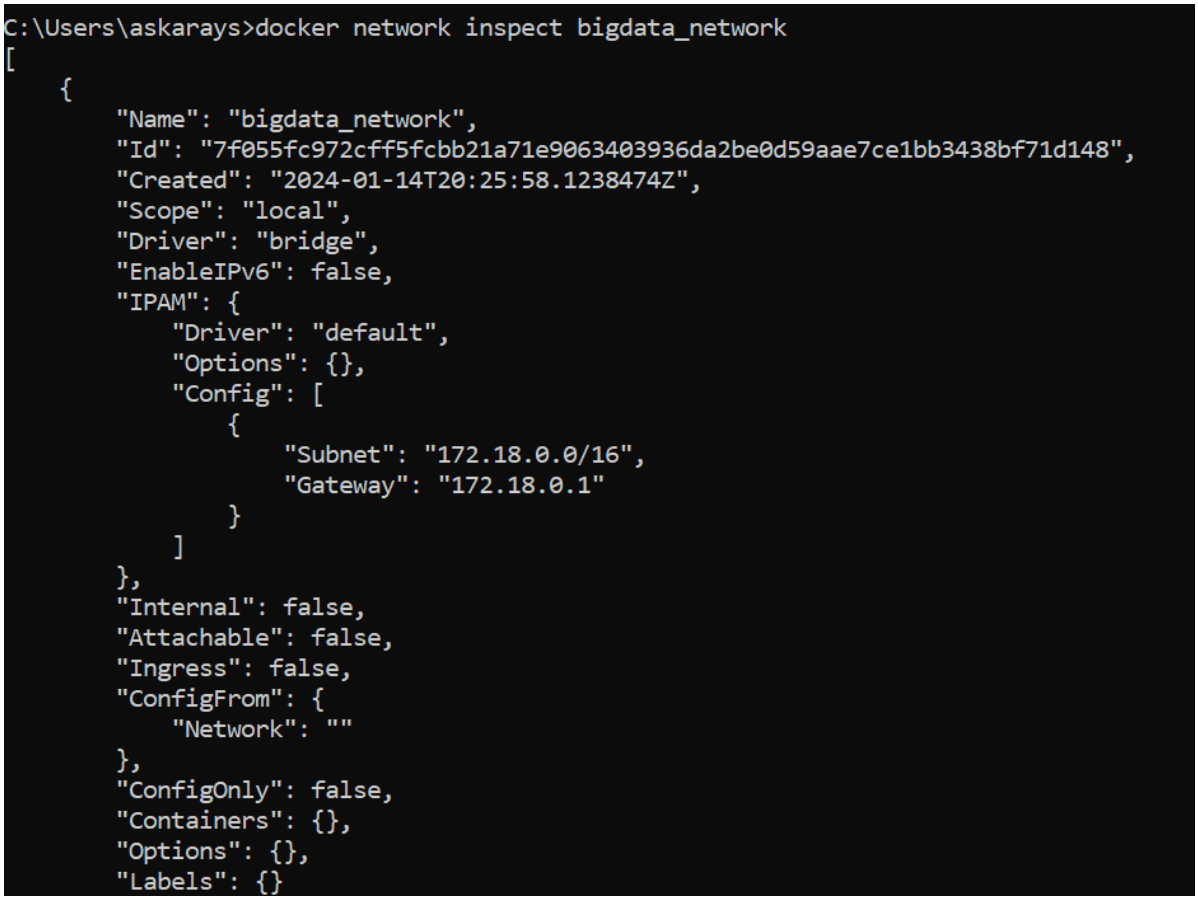
Kafka

Askarova Aisha

▼ **CONNECT TO THE THE KAFKA ENVIRONMENT**

First I create a virtual docker network

Then create the kafka docker container and start



C:\Users\askarays>docker network inspect bigdata_network
[
{
"Name": "bigdata_network",
"Id": "7f055fc972cff5fcbb21a71e9063403936da2be0d59aae7ce1bb3438bf71d148",
"Created": "2024-01-14T20:25:58.1238474Z",
"Scope": "local",

"Driver": "bridge",
"EnableIPv6": false,
"IPAM": {
"Driver": "default",
"Options": {},
"Config": [
{
"Subnet": "172.18.0.0/16",
"Gateway": "172.18.0.1"
}
]
},
"Internal": false,
"Attachable": false,
"Ingress": false,
"ConfigFrom": {
"Network": ""
},
"ConfigOnly": false,
"Containers": {},
"Options": {},
"Labels": {}
}
]

```
C:\Users\askarays>docker create --name kafka --network bigdata_network -p 2181:2181 -p 9092:9092 -p 8083:8083 -e ADVERT
SED_HOST=172.18.0.1 -it ofrir119/kafka:2.4.0
Unable to find image 'ofrir119/kafka:2.4.0' locally
2.4.0: Pulling from ofrir119/kafka
8e402f1a9c57: Pull complete
230022b1752e: Pull complete
59d760897013: Pull complete
1ac8479ef1a8: Pull complete
08ef9a9acf6e: Pull complete
6009319f03be: Pull complete
062044acb3f4: Pull complete
42238460f1ed: Pull complete
bf81c5bba825: Pull complete
32b4146a7660: Pull complete
d526d9ab574a: Pull complete
cef973c02ae6: Pull complete
Digest: sha256:53026538a98cc8c94518df61e7b7f2bcc7ee681c91d430e18f0dd84dd62bbd8f
Status: Downloaded newer image for ofrir119/kafka:2.4.0
6d48f1e1fee768ac8271ea2eb50cc2b91506c2d0dcc1fc9e45bc10e51aa46365
```

- Open a BASH session in the practice environment. Be sure to include the "-l" flag to load scripts.

  docker exec -it kafka /bin/bash

```
C:\Users\askarays>docker exec -it kafka /bin/bash
bash-4.4# ls /opt/kafka/bin
```

1. **CREATING KAFKA TOPICS (10)**

   - View the Kafka related scripts in the Kafka "Bin" directory

```
bash-4.4# ls /opt/kafka_2.12-2.4.0/bin
connect-distributed.sh              kafka-delete-records.sh             kafka-server-stop.sh
connect-mirror-maker.sh             kafka-dump-log.sh                   kafka-streams-application-reset.sh
connect-standalone.sh               kafka-leader-election.sh            kafka-topics.sh
kafka-acls.sh                       kafka-log-dirs.sh                   kafka-verifiable-consumer.sh
kafka-broker-api-versions.sh        kafka-mirror-maker.sh               kafka-verifiable-producer.sh
kafka-configs.sh                    kafka-preferred-replica-election.sh trogdor.sh
kafka-console-consumer.sh           kafka-producer-perf-test.sh         windows
kafka-console-producer.sh           kafka-reassign-partitions.sh        zookeeper-security-migration.sh
kafka-consumer-groups.sh            kafka-replica-verification.sh       zookeeper-server-start.sh
kafka-consumer-perf-test.sh         kafka-run-class.sh                  zookeeper-server-stop.sh
kafka-delegation-tokens.sh          kafka-server-start.sh               zookeeper-shell.sh
bash-4.4# echo $KAFKA_HOME
/opt/kafka_2.12-2.4.0
```

   Hint: View the PATH variable contents to see the location of the directory. You should see ~25 scripts

   - Check the following:
   - The value of KAFKA_HOME in your environment

```
bash-4.4# echo $KAFKA_HOME
/opt/kafka_2.12-2.4.0
```

   - If there are any configured topics currently in Kafka
     No
   - Create a new Kafka topic named **kafka-tst-01**, which has one partition and runs on one node.
   - Check again and make sure your topic was created.

```
bash-4.4# $KAFKA_HOME/bin/kafka-topics.sh --list --zookeeper localhost:2181
bash-4.4# /opt/kafka_2.12-2.4.0/bin/kafka-topics.sh --list --zookeeper localhost:2181
bash-4.4# $KAFKA_HOME/bin/kafka-topics.sh --create --zookeeper localhost:2181 --topic kafka-tst-01 --partitions 1 --replication-factor 1
Created topic kafka-tst-01.
bash-4.4# $KAFKA_HOME/bin/kafka-topics.sh --list --zookeeper localhost:2181
kafka-tst-01
```

2. **WRITING AND READING KAFKA TOPICS (15)**

- Write a command which reads data from a Kafka topic and writes it to the standard output. Think, which script should you use for this task?
  I use
  kafka-console-consumer.sh script to read data from a Kafka topic.

```
$KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic kafka-tst-01--from-beginning
```

```
bash-4.4# $KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic kafka-tst-01--from-beginning
[2024-01-15 08:34:20,359] WARN [Consumer clientId=consumer-console-consumer-42032-1, groupId=console-consumer-42032] Error while fetching
metadata with correlation id 2 : {kafka-tst-01--from-beginning=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
```

Note: Your screen should "hang" waiting for incoming messages. You can always break using Ctrl+c, but do not do this yet.

- Open a second terminal window and connect to the Docker there too.

```
C:\Users\askarays>docker exec -it kafka /bin/bash
bash-4.4#
```

- Write a command which reads data from the standard input and writes it to your Kafka topic. Think, which script should you use for this task?
  Use kafka-console-producer.sh script

```
bash-4.4# $KAFKA_HOME/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic kafka-tst-01
>
>
```

Note: You should get the ">" prompt, waiting for you to enter your text.

- Write the text "Hello world! This is Kafka!" in the producer side (standard input, Session 2)

```
bash-4.4# $KAFKA_HOME/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic kafka-tst-01
>Hello world! This is Kafka!
>
```

- Ensure the consumer receives the messages (Standard output, Session 1).

```
bash-4.4# $KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic kafka-tst-01
Hello world! This is Kafka!
```

- Produce a few more messages: "Event 1", "Event 2".. and see how they are consumed. Think if you can have another producer writing to the same topic.

```
bash-4.4# $KAFKA_HOME/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic kafka-tst-01
>Hello world! This is Kafka!
>Event 1
>Event 2
```

```
bash-4.4# $KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic kafka-tst-01
Hello world! This is Kafka!
Event 1
Event 2
```

- Check if your assumption is correct: Open the third session and create the second producer writing to the same topic.

```
Командная строка - docker  exec -it kafka /bin/bash
Microsoft Windows [Version 10.0.19044.1288]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\askarays>
C:\Users\askarays>docker exec -it kafka /bin/bash
bash-4.4# $KAFKA_HOME/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic kafka-tst-01
>Hello world from second producer
>
```

- Produce some events and make sure they are consumed by the consumer session.

```
bash-4.4# $KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic kafka-tst-01
Hello world! This is Kafka!
Event 1
Event 2
Hello world from second producer
```

- Think if you can have another consumer reading from the same topic.
- Check if your assumption is correct: Open the 4th session and create the second consumer reading from the same topic.

- Answer the questions: Which consumer will now receive new events? Just one of them? Both of them? Round robin?

  Both of them

- Produce an event and check if your assumption is correct.
- Think if you can create a new consumer that will read all events generated since the beginning of the topic.

- Create such a consumer and check if your assumption is correct.
- Delete the Kafka topic you created. You should get the message: "Topic kafka-tst-01 is marked for deletion..."

`$KAFKA_HOME/bin/kafka-topics.sh --delete --bootstrap-server localhost:9092 --topic kafka-tst-01`

```
bash-4.4# $KAFKA_HOME/bin/kafka-topics.sh --delete --bootstrap-server localhost:9092 --topic kafka-tst-01
bash-4.4#
```

```
C:\Users\askarays>docker exec -it kafka /bin/bash
bash-4.4# kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic kafka-tst-01
Hello world 2
Hello world 1
[2024-01-15 09:19:49,219] WARN [Consumer clientId=consumer-console-consumer-71429-1, groupId=console-consumer-71429] Received u
nknown topic or partition error in fetch for partition kafka-tst-01-0 (org.apache.kafka.clients.consumer.internals.Fetcher)
[2024-01-15 09:19:49,223] WARN [Consumer clientId=consumer-console-consumer-71429-1, groupId=console-consumer-71429] Error whil
e fetching metadata with correlation id 1144 : {kafka-tst-01=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2024-01-15 09:19:49,337] WARN [Consumer clientId=consumer-console-consumer-71429-1, groupId=console-consumer-71429] Error whil
e fetching metadata with correlation id 1146 : {kafka-tst-01=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2024-01-15 09:19:49,338] WARN [Consumer clientId=consumer-console-consumer-71429-1, groupId=console-consumer-71429] The follow
ing subscribed topics are not assigned to any members: [kafka-tst-01]  (org.apache.kafka.clients.consumer.internals.ConsumerCoo
rdinator)
```

- Make sure the topic was deleted.

```
bash-4.4# $KAFKA_HOME/bin/kafka-topics.sh --delete --bootstrap-server localhost:9092 --topic kafka-tst-01
bash-4.4# $KAFKA_HOME/bin/kafka-topics.sh --list --bootstrap-server localhost:9092
__consumer_offsets
bash-4.4#
```

`$KAFKA_HOME/bin/kafka-topics.sh --list --bootstrap-server localhost:9092`

3. **USING KAFKA CONNECT TO READ FROM A SOURCE FILE (15)**

- Create the following directories in the system for using them in the Kafka practice:

mkdir -p /kafka/{confFiles,srcFiles,sinkFiles}

- Verify the directories were created:

ls -l /kafka/

```
bash-4.4# mkdir -p /kafka/{confFiles,srcFiles,sinkFiles}
bash-4.4# ls -l /kafka/
total 12
drwxr-xr-x    2 root     root          4096 Jan 15 09:28 confFiles
drwxr-xr-x    2 root     root          4096 Jan 15 09:28 sinkFiles
drwxr-xr-x    2 root     root          4096 Jan 15 09:28 srcFiles
bash-4.4# vi /kafka/confFiles/connect-file-source.properties
```

- Use the "VI" editor to create a file name **connect-file-source.properties** in the **confFiles** directory, with the following contents:
- Fill up the missing values

#---------- #Content of connect-file-source.properties

#----------

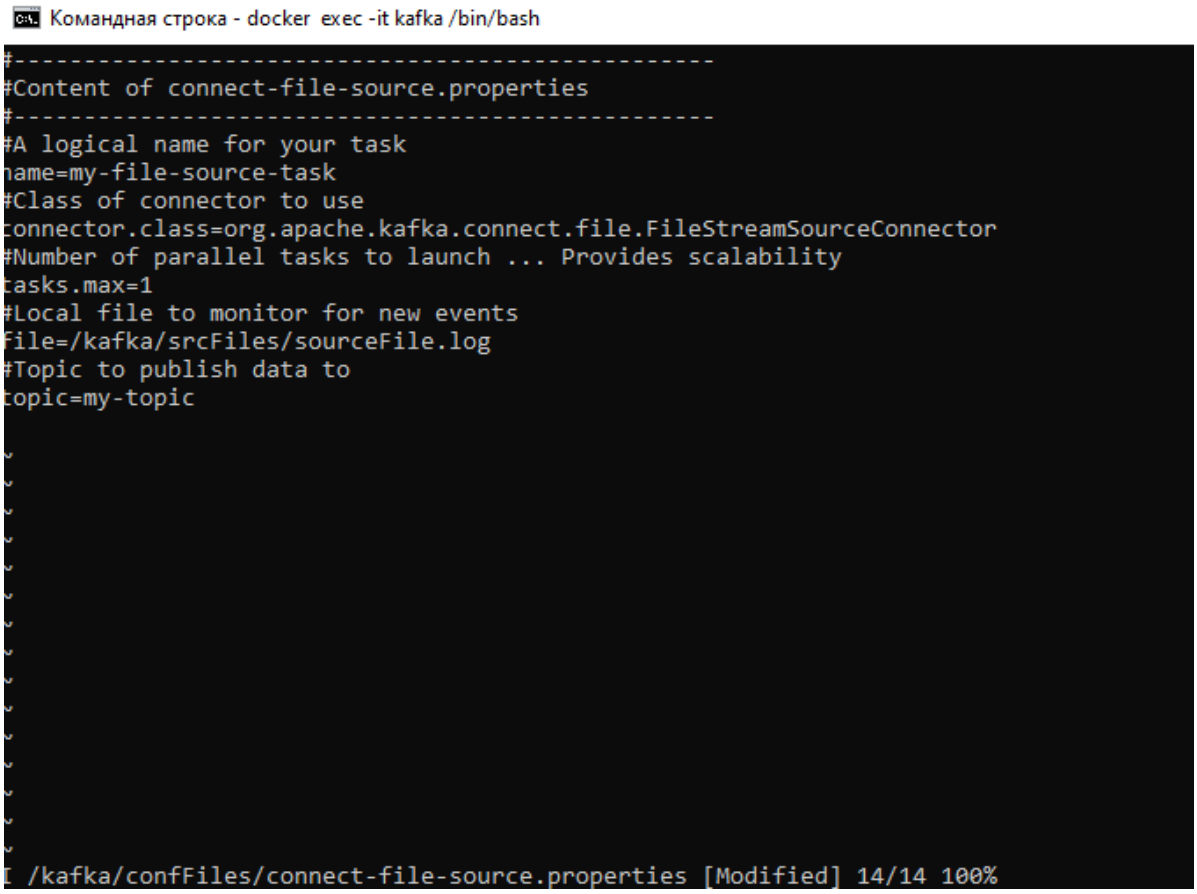
#A logical name for your task name=kafka-file-source-task

#Class of connector to use connector.class=<*Value*>

#Number of parallel tasks to launch – Provides scalability tasks.max=1

#Local file to monitor for new events file=/kafka/srcFiles/sourceFile.log

#Topic to publish data to topic=<*Value*>



- Exit and save the file (ESC + :wq)
- Insert a few records to the source file using the following script:

```
$ echo 'Event 1 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/sourceFile.log $ sleep 1
```

```
$ echo 'Event 2 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/sourceFile.log $ sleep 1
```

```
$ echo 'Event 3 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/sourceFile.log
```

```
bash-4.4# vi /kafka/confFiles/connect-file-source.properties
bash-4.4# echo 'Event 1 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/sourceFile.log
bash-4.4# sleep 1
bash-4.4# echo 'Event 2 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/sourceFile.log
bash-4.4# sleep 1
bash-4.4# echo 'Event 3 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/sourceFile.log
```

- Verify the contents of the source file you created

`cat /kafka/srcFiles/sourceFile.log`

```
bash-4.4# cat /kafka/srcFiles/sourceFile.log
Event 1 |  6d48f1e1fee7  |  Mon Jan 15 09:34:25 UTC 2024
Event 2 |  6d48f1e1fee7  |  Mon Jan 15 09:34:26 UTC 2024
Event 3 |  6d48f1e1fee7  |  Mon Jan 15 09:34:27 UTC 2024
```

- Issue the Kafka Connect Standalone script with the required parameters to read the created source file. See the guidelines document for a few pointers if you require assistance.

`$KAFKA_HOME/bin/connect-standalone.sh $KAFKA_HOME/config/connect-standalone.properties /kafka/confFiles/connect-file-source.properties`

Note: In this practice task, you only read from a file (source) and do not write to a file (sink). So, address only the relevant parameters.

- You can ignore any messages you receive when running the script, as it throws a lot of WARNING and INFO messages. As long as the script does not crash, everything is fine.
- Create a consumer that reads data from the topic you are using and write them to the screen.

`$KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic my-topic --from-beginning`

```
bash-4.4# $KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic my-topic --from-beginning
{"schema":{"type":"string","optional":false},"payload":"Event 1 |  6d48f1e1fee7  |  Mon Jan 15 09:34:25 UTC 2024"}
{"schema":{"type":"string","optional":false},"payload":"Event 2 |  6d48f1e1fee7  |  Mon Jan 15 09:34:26 UTC 2024"}
{"schema":{"type":"string","optional":false},"payload":"Event 3 |  6d48f1e1fee7  |  Mon Jan 15 09:34:27 UTC 2024"}
^C
```

Note: Specify a flag to ensure you get all messages from the beginning of the queue (all rows in the file source, including rows that were sent earlier to the topic).

- Write a few more messages to the file source:

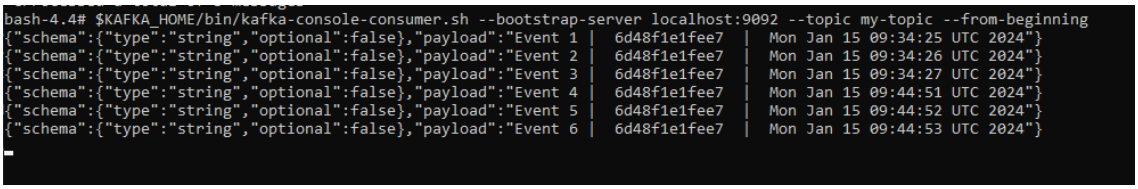
$ echo 'Event 1 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/sourceFile.log $ sleep 1

$ echo 'Event 2 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/sourceFile.log

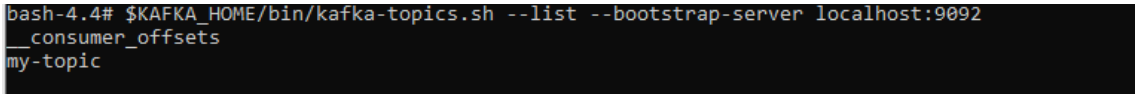
$ sleep 1

$ echo 'Event 3 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/sourceFile.log

- Make sure you received them on the consumer's side.



- List current Kafka topics and make sure you see the topic you specified.



- Stop both processes (producer and consumer) but drop the topic.

  do not

4. **USING KAFKA CONNECT TO WRITE TO A DESTINATION FILE (20)**

   Use the "VI" editor to overwrite the file you created in the previous practice named **connect-file-source.properties**, with the following contents.

   - Fill up the missing values

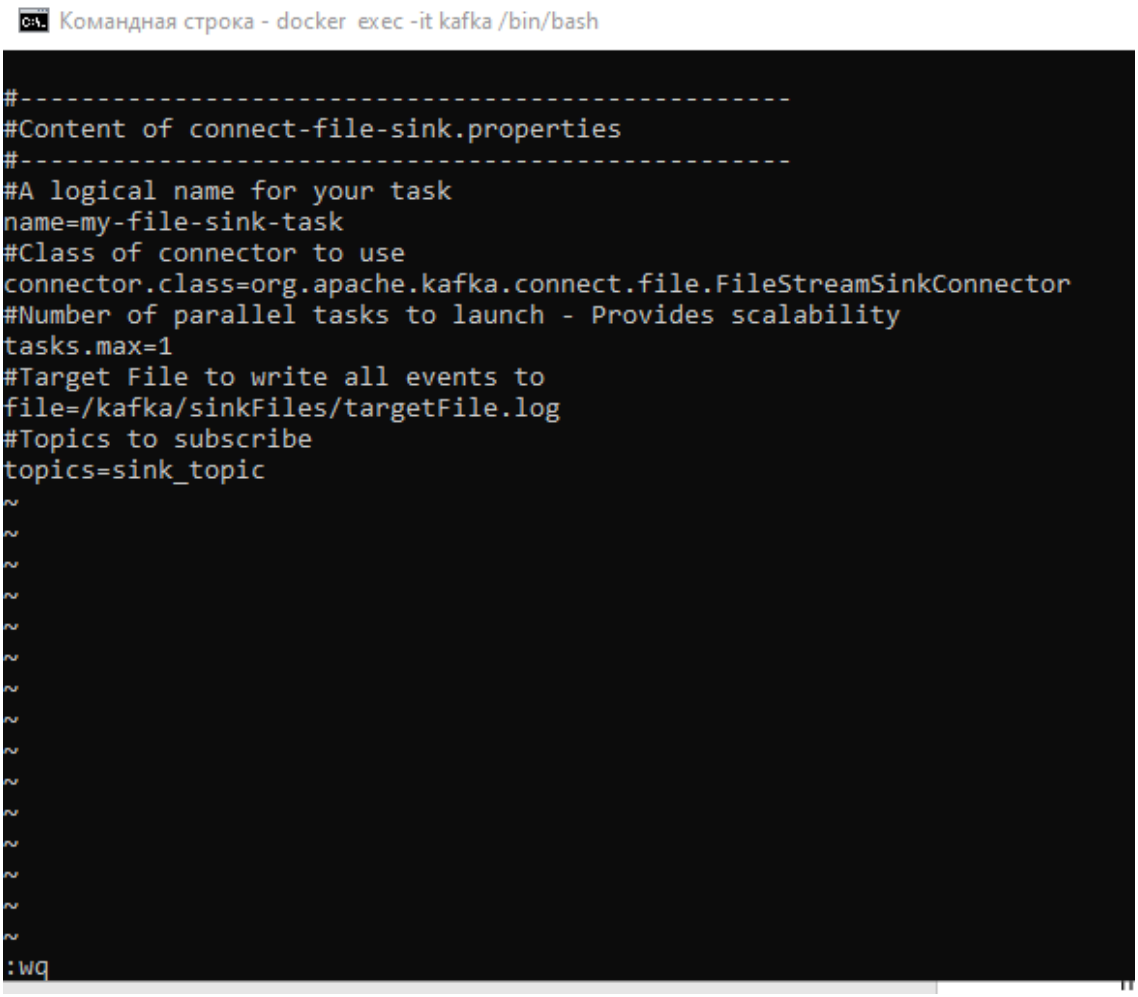
```
#---------------------------------------------------
#Content of connect-file-source.properties
#---------------------------------------------------
#A logical name for your task
name=kafka-file-source-task-2
#Class of connector to use
connector.class=org.apache.kafka.connect.file.FileStreamSourceConnector
#Number of parallel tasks to launch – Provides scalability
tasks.max=1
#Local file to monitor for new events
file=/kafka/srcFiles/newSourceFile.log
#Topic to publish data to (use a new topic name)
topic=new_kafka_topic
```

```
#---------------------------------------------------
#Content of connect-file-source.properties
#---------------------------------------------------
#A logical name for your task
name=kafka-file-source-task-2
#Class of connector to use
connector.class=org.apache.kafka.connect.file.FileStreamSourceConnector
#Number of parallel tasks to launch ... Provides scalability
tasks.max=1
#Local file to monitor for new events
file=/kafka/srcFiles/newSourceFile.log
#Topic to publish data to (use a new topic name)
topic=new_kafka_topic

~
~
~
~
```

- Exit and save the file (ESC + :wq)
- Use "VI" editor to create a file named **connect-file-sink.properties**, with the following contents. Fill up the missing values

```
#---------------------------------------------------
#Content of connect-file-sink.properties
#---------------------------------------------------
#A logical name for your task
name=my-file-sink-task
#Class of connector to use
connector.class=org.apache.kafka.connect.file.FileStreamSinkConnector
#Number of parallel tasks to launch - Provides scalability
tasks.max=1
#Target File to write all events to
file=/kafka/sinkFiles/targetFile.log
#Topics to subscribe
topics=new_kafka_topic
```



- Write a few messages to the new source file:

$ echo 'Event 1 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log $ sleep 1

$ echo 'Event 2 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log $ sleep 1

$ echo 'Event 3 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log

```
bash-4.4# echo 'Event 1 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log
bash-4.4# sleep 1
bash-4.4# echo 'Event 2 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log
bash-4.4# sleep 1
bash-4.4# echo 'Event 3 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log
bash-4.4#
```

- Ensure the contents of the source file you created:

$ cat /kafka/srcFiles/newSourceFile.log

```
bash-4.4# cat /kafka/srcFiles/newSourceFile.log
Event 1 |  6d48f1e1fee7  |  Mon Jan 15 10:18:40 UTC 2024
Event 2 |  6d48f1e1fee7  |  Mon Jan 15 10:18:41 UTC 2024
Event 3 |  6d48f1e1fee7  |  Mon Jan 15 10:18:42 UTC 2024
bash-4.4#
```

- Create the target file manually:

  $ touch /kafka/sinkFiles/targetFile.log

- Issue the Kafka Connect Standalone script with the required parameters to read the source file you created and write to the target (sink) file. See the guidelines document for a few pointers if you require assistance.

  ```
  $KAFKA_HOME/bin/connect-standalone.sh $KAFKA_HOME/config/connect-standalone.properties /kafka/confFiles/connect-file-source.properties /kafka/confFiles/connect-file-sink.properties
  ```

Note: In this practice task, you read from a file (source) and write to a file (sink). So, address both relevant parameters.

- You can ignore any messages you receive when running the script, as it throws a lot of WARNING and INFO messages. As long as the script does not crash, everything is fine.
- Ensure the messages arrived at the file target destination. (Use "***cat***")

```
bash-4.4# cat /kafka/sinkFiles/targetFile.log
Event 1 |  6d48f1e1fee7  |  Mon Jan 15 10:18:40 UTC 2024
Event 2 |  6d48f1e1fee7  |  Mon Jan 15 10:18:41 UTC 2024
Event 3 |  6d48f1e1fee7  |  Mon Jan 15 10:18:42 UTC 2024
```

- Create a consumer that reads data from the topic you are using and writes them to the screen.

```
bash-4.4# $KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic new_kafka_topic --from-beginning
{"schema":{"type":"string","optional":false},"payload":"Event 1 |  6d48f1e1fee7  |  Mon Jan 15 10:18:40 UTC 2024"}
{"schema":{"type":"string","optional":false},"payload":"Event 2 |  6d48f1e1fee7  |  Mon Jan 15 10:18:41 UTC 2024"}
{"schema":{"type":"string","optional":false},"payload":"Event 3 |  6d48f1e1fee7  |  Mon Jan 15 10:18:42 UTC 2024"}
```

Note: Specify a flag to ensure you get all messages from the beginning of the queue (all rows in the file source, including rows that were sent earlier to the topic).

- List current Kafka topics and make sure you see the topic you specified.

```
bash-4.4# $KAFKA_HOME/bin/kafka-topics.sh --list --bootstrap-server localhost:9092
__consumer_offsets
my-topic
new_kafka
new_kafka_topic
sink_topic
bash-4.4# _
```

- Insert a few more rows into the source file:

$ echo 'Event 1 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log $ sleep 1

$ echo 'Event 2 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log $ sleep 1

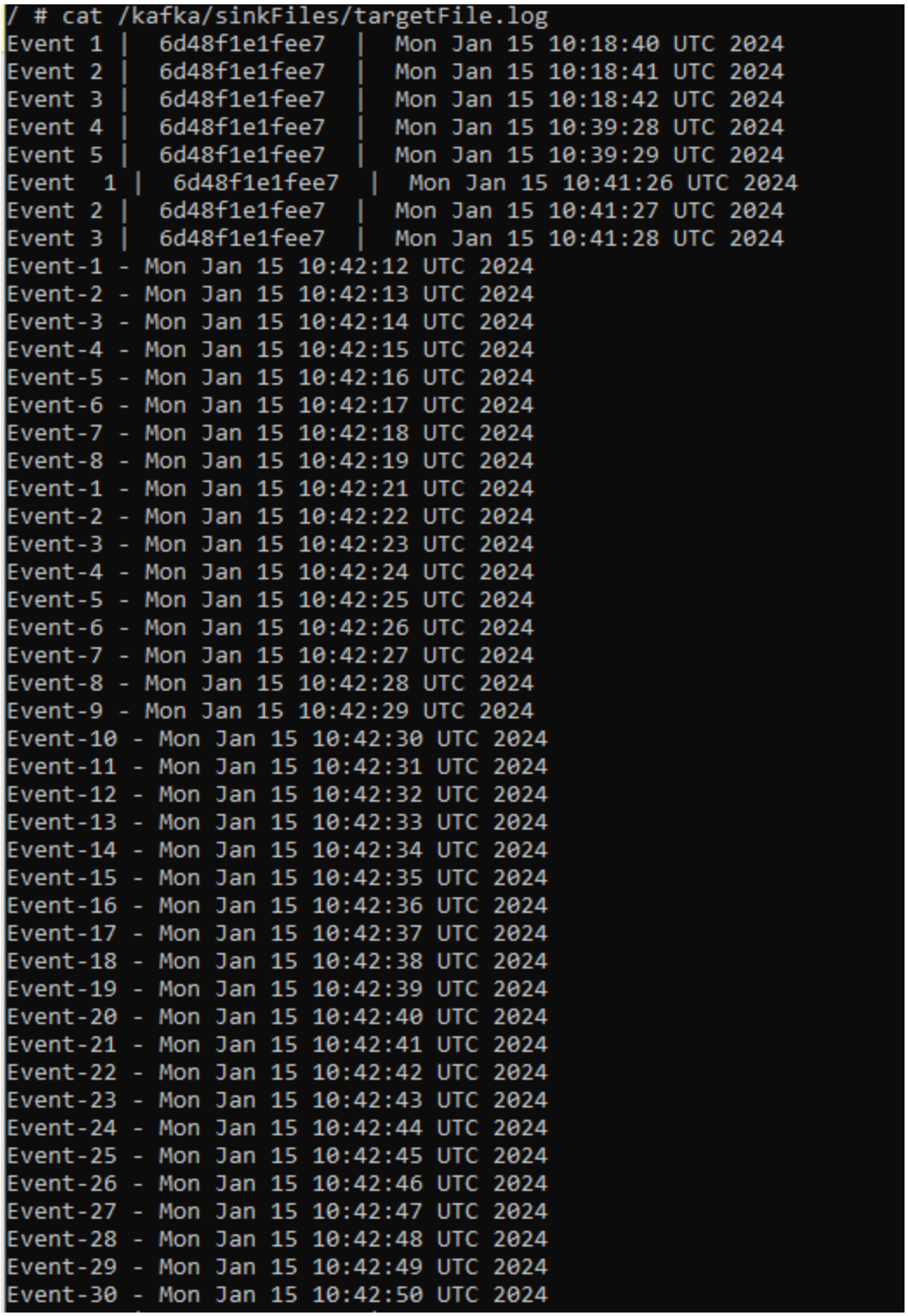
$ echo 'Event 3 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log

```
echo 'Event  1 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log
sleep 1
echo 'Event 2 | ' $(hostname) ' | ' $(date) >> /kafka/srcFiles/newSourceFile.log
sleep 1
echo 'Event 3 | ' $(hostname) ' | ' $(date) >>/kafka/srcFiles/newSourceFile.log
```

- Run the following script that creates new messages in the file for 30 seconds:

$ for v in {1..30}; do echo Event-$v '-' `date` >> /kafka/srcFiles/newSourceFile.log && sleep 1; done

- On another terminal window, ensure the messages arrive at the file target destination. (Use "***cat***")

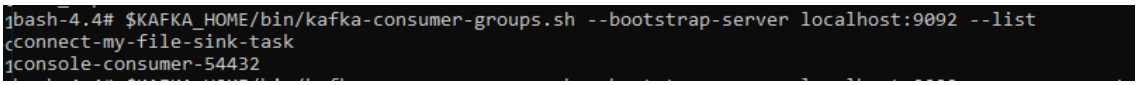


- Leave all processes active (producer and both consumers).

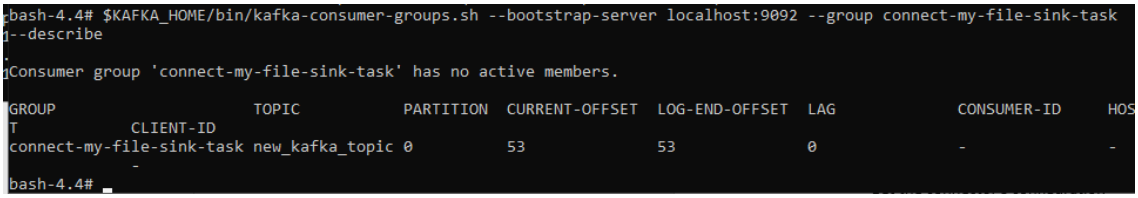
5. **KAFKA ADMINISTRATION (20)**

This task builds upon the previous one: "Using Kafka Connect to read from a source file," and assumes the consumers and producer processes are still active.

- List the Kafka consumer groups.

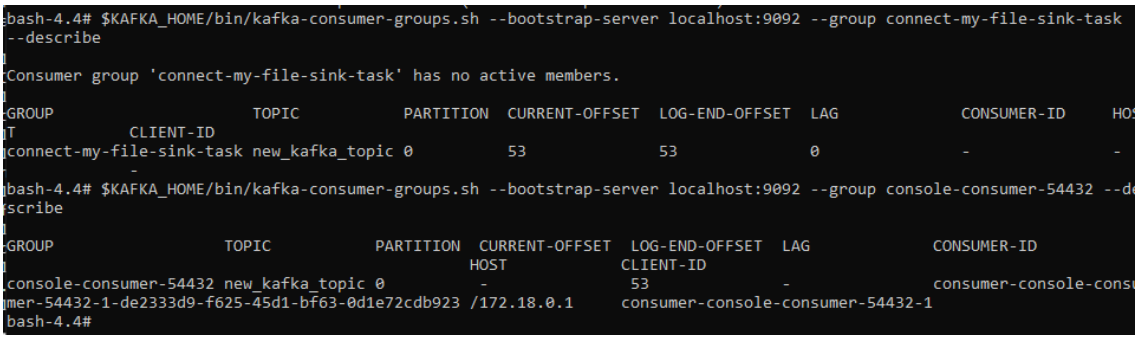


- Check out which groups and consumers you see.
- Check the Kafka consumers' offsets.



- Try to run with different consumer groups (change the group name in the group parameter).

`$KAFKA_HOME/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --group console-consumer-54432 --describe`



- Add "curl" to the environment:

  apk add curl

```
bash-4.4# apk add curl
fetch http://dl-cdn.alpinelinux.org/alpine/v3.9/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.9/community/x86_64/APKINDEX.tar.gz
(1/4) Installing nghttp2-libs (1.35.1-r2)
(2/4) Installing libssh2 (1.9.0-r1)
(3/4) Installing libcurl (7.64.0-r5)
(4/4) Installing curl (7.64.0-r5)
Executing busybox-1.29.3-r10.trigger
OK: 140 MiB in 68 packages
bash-4.4# _
```

- Use the Kafka REST API to get the following information from Kafka Connect (port 8083). (Use the **tool** to get more readable results. See the guidelines for more info.)

```
bash-4.4# curl localhost:8083/connectors | python -m json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0curl: (7) Failed to connect to localhost port 8083: Connection refused
No JSON object could be decoded
```

- Current running connectors

```
bash-4.4# curl http://localhost:8083/connectors | python -m json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     2  100     2    0     0    666      0 --:--:-- --:--:-- --:--:--   666
[]
```

- Active tasks for one of the running connectors
- Status for one of the running connectors
- Get the connector's configuration
- View the documentation using the Guidelines document link, and perform a few more queries.
- Use the information you found above to reset the file-sink consumer (screen output) to the earliest point possible for the topic it is consuming.
- See if you can change the offset while it is active.
- Stop it if required and change the offset.
- Activate it again and ensure it consumed all records from the top of the file.
- Stop and restart it without changing the offset and see that no new records have been consumed, as the offset did not change.
- Use the following command to check the rowcount:

cat /kafka/sinkFiles/targetFile.log | wc -l

```
bash-4.4# cat /kafka/sinkFiles/targetFile.log | wc -l
53
```

6. **USING KAFKA CONNECT TO READ FROM A DATABASE (MYSQL) (20)**

```
C:\Users\askarays>docker run -d --name mysql --network bigdata_network -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
bce031bc522d: Pull complete
cf7e9f463619: Pull complete
105f403783c7: Pull complete
878e53a613d8: Pull complete
2a362044e79f: Pull complete
6e4df4f73cfe: Pull complete
69263d634755: Pull complete
fe5e85549202: Pull complete
5c02229ce6f1: Pull complete
7320aa32bf42: Pull complete
Digest: sha256:4ef30b2c11a3366d7bb9ad95c70c0782ae435df52d046553ed931621ea36ffa5
Status: Downloaded newer image for mysql:latest
f212e9ae415aecab6e2b6ce1b5122ab30064e9c172a7f4260adaa06a757144e9
```

- Add and run the MySQL container as described in the installation guide.
- Ensure it is up and running using the **docker stats**
- Open a new terminal window and connect to the MySQL container:

docker exec -it mysql /bin/bash

Inside the MySQL Container:

§ Connect to MySQL, and run the following script:

mysql -uroot -proot

- Run the following script:

```
drop database if exists srcdb; create database srcdb;
use srcdb;
create table src_events( event_id int primary key, event_timestamp timestamp not null);
```

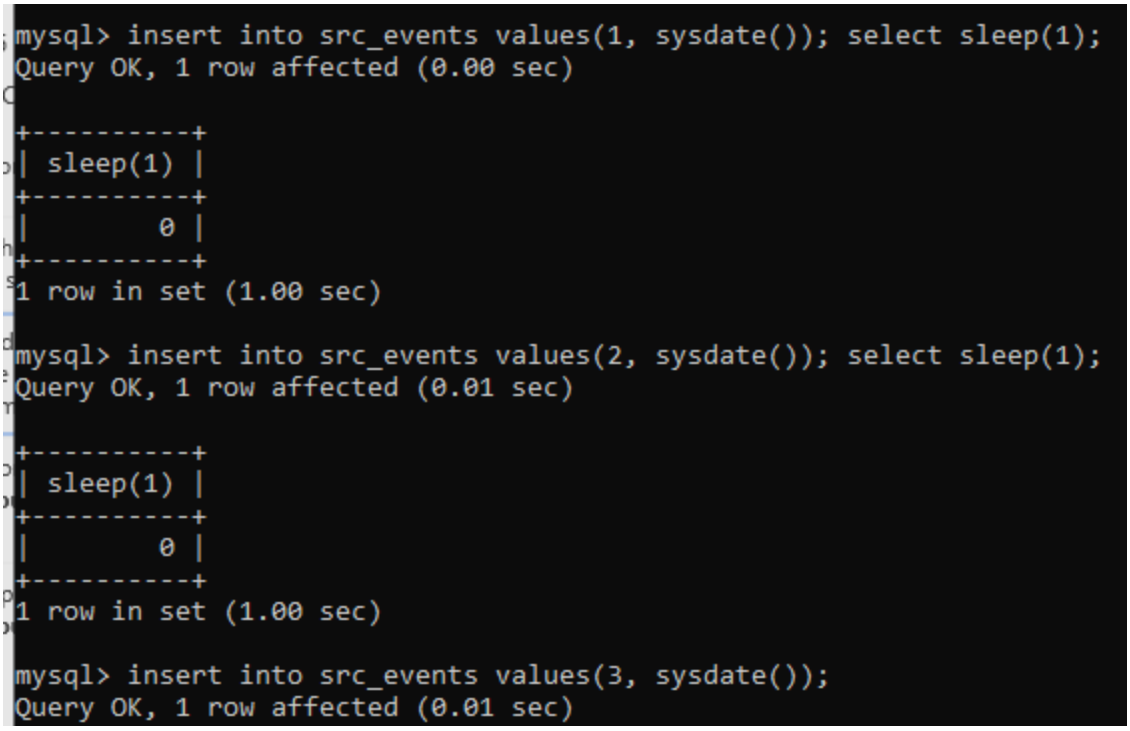
```
mysql> create database srcdb;
Query OK, 1 row affected (0.00 sec)

mysql> use srcdb;
Database changed
mysql> create table src_events(
    ->     event_id int primary key,
    ->     event_timestamp timestamp not null
    -> );
Query OK, 0 rows affected (0.02 sec)
```

insert into src_events values(1, sysdate()); select sleep(1);

insert into src_events values(2, sysdate()); select sleep(1);

insert into src_events values(3, sysdate());



```
mysql> insert into src_events values(1, sysdate()); select sleep(1);
Query OK, 1 row affected (0.00 sec)

+----------+
| sleep(1) |
+----------+
|        0 |
+----------+
1 row in set (1.00 sec)

mysql> insert into src_events values(2, sysdate()); select sleep(1);
Query OK, 1 row affected (0.01 sec)

+----------+
| sleep(1) |
+----------+
|        0 |
+----------+
1 row in set (1.00 sec)

mysql> insert into src_events values(3, sysdate());
Query OK, 1 row affected (0.01 sec)
```

create table web_logins(login_time timestamp, login_count int

);

insert into web_logins values(sysdate(), 0);

```
mysql> create table web_logins(
    ->     login_time timestamp,
    ->     login_count int
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> insert into web_logins values(sysdate(), 0);
Query OK, 1 row affected (0.00 sec)
```

exit;

```
mysql> exit;
Bye
```

- Verify the tables and rows were created successfully:

mysql -uroot -proot srcdb -e 'select * from src_events'

- Minimize the terminal window connected to the MySQL container, and continue with the Kafka container connection.

```
bash-4.4# mysql -uroot -proot srcdb -e 'select * from src_events'
mysql: [Warning] Using a password on the command line interface can be insecure.
+----------+---------------------+
| event_id | event_timestamp     |
+----------+---------------------+
|        1 | 2024-01-15 11:37:15 |
|        2 | 2024-01-15 11:37:16 |
|        3 | 2024-01-15 11:37:17 |
+----------+---------------------+
bash-4.4#
```

Inside the Kafka Container

o Ensure the directory names in the Kafka practice directory:

ls -l /kafka/

```
:\Users\askarays>docker exec -it kafka /bin/bash
ash-4.4# ls -l /kafka/
otal 12
rwxr-xr-x    2 root     root          4096 Jan 15 10:18 confFiles
rwxr-xr-x    2 root     root          4096 Jan 15 10:19 sinkFiles
rwxr-xr-x    2 root     root          4096 Jan 15 10:18 srcFiles
```

o Use the "VI" editor to create a file name connect-jdbc-source.properties in the confFiles directory with the following contents.

- Fill up the missing values
- Notes:
  - Get the information from the table named "src_events"
  - We want to identify new rows according to values in the "event_id" column.
  - MySQL database name is "srcdb"
  - MySQL username and password are root:root
- You can refer to the Kafka Practice Guidelines documents for assistance and hints.

#---------- #Content of connect-jdbc-source.properties

#----------

#Name of the connector name=Kafka-jdbc-source-task-1

#Connector class to be used.

connector.class=<*Value*>

#JDBC connector URL for mysql. make sure the mysql driver is in classpath. connection.url=jdbc:mysql://<*Gateway-IP*>:3306/<*db-name*>?user=<*username*>&password=<*password*>&allowPublicKeyRetrieval=true

#List of tables to publish. you can also use blacklists table.whitelist=<*Value*>

#No. of parallel tasks - Ideally one per table tasks.max=1

#How frequently to poll from the database for new records poll.interval.ms=2000

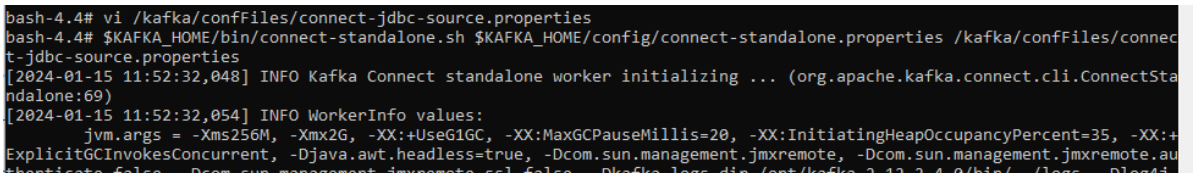
#mode - incrementing or timestamp+incrementing mode=<*Value*>

incrementing.column.name=<*Value*>

#Topic name to be created - This will create a topic with the prefix you specify, with the table name appended topic.prefix=<*Value*>

```
name=Kafka-jdbc-source-task-1
connector.class=io.confluent.connect.jdbc.JdbcSourceConnector
connection.url=jdbc:mysql://172.18.0.1:3306/srcdb?
user=root&password=root&allowPublicKeyRetrieval=true
table.whitelist=src_events
tasks.max=1
poll.interval.ms=2000
mode=incrementing
incrementing.column.name=event_id
topic.prefix=my_topic
```

- Exit and save the file (ESC + :wq)
- Issue the Kafka Connect Standalone script with the required parameters to read the source table from MySQL using the created properties file.



- See the guidelines document for a few pointers if you require assistance.

Note: In this practice, you only read from a table (source) and do not write to any table or file (sink). So, address only the relevant parameters.

- As long as the script does not crash, everything is fine.
- In case of an error, **Address is already in use** refer to the troubleshooting guide.
- Create a consumer that reads data from the topic you are using and write them to the screen.

Note: Specify a flag to ensure you get all messages from the beginning of the queue (all rows in the file source, including rows that were sent earlier to the topic).

- On the MySQL container, insert a few more records and verify they are ingested by Kafka and consumed by the consumer process:

```
bash-4.4# mysql -uroot -proot srcdb -e 'select * from src_events'
mysql: [Warning] Using a password on the command line interface can be insecure.
+----------+---------------------+
| event_id | event_timestamp     |
+----------+---------------------+
|        1 | 2024-01-15 11:37:15 |
|        2 | 2024-01-15 11:37:16 |
|        3 | 2024-01-15 11:37:17 |
+----------+---------------------+
bash-4.4# mysql -uroot -proot srcdb -e 'insert into src_events select max(event_id)+1, sysdate() from src_events'
mysql: [Warning] Using a password on the command line interface can be insecure.
bash-4.4# mysql -uroot -proot srcdb -e 'select * from src_events'
mysql: [Warning] Using a password on the command line interface can be insecure.
+----------+---------------------+
| event_id | event_timestamp     |
+----------+---------------------+
|        1 | 2024-01-15 11:37:15 |
|        2 | 2024-01-15 11:37:16 |
|        3 | 2024-01-15 11:37:17 |
|        4 | 2024-01-15 11:55:09 |
```

$ mysql -uroot -proot srcdb -e 'insert into src_events select max(event_id)+1, sysdate() from src_events'

$ mysql -uroot -proot srcdb -e 'select * from src_events'

- Verify you received them on the consumer side.

```
bash-4.4# $KAFKA_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic my-topic --from-beginning
{"schema":{"type":"string","optional":false},"payload":"Event 1 |  6d48f1e1fee7  |  Mon Jan 15 09:34:25 UTC 2024"}
{"schema":{"type":"string","optional":false},"payload":"Event 2 |  6d48f1e1fee7  |  Mon Jan 15 09:34:26 UTC 2024"}
{"schema":{"type":"string","optional":false},"payload":"Event 3 |  6d48f1e1fee7  |  Mon Jan 15 09:34:27 UTC 2024"}
{"schema":{"type":"string","optional":false},"payload":"Event 4 |  6d48f1e1fee7  |  Mon Jan 15 09:44:51 UTC 2024"}
{"schema":{"type":"string","optional":false},"payload":"Event 5 |  6d48f1e1fee7  |  Mon Jan 15 09:44:52 UTC 2024"}
{"schema":{"type":"string","optional":false},"payload":"Event 6 |  6d48f1e1fee7  |  Mon Jan 15 09:44:53 UTC 2024"}
```

- List current Kafka topics and verify you see the topic you have specified.

```
bash-4.4# $KAFKA_HOME/bin/kafka-topics.sh --list --zookeeper localhost:2181
__consumer_offsets
connect-configs
connect-offsets
connect-status
my-file-source-topic
my-topic
new_kafka
new_kafka_topic
sink_topic
```

- Stop both processes (producer and consumer) and drop the topic.

```
bash-4.4# $KAFKA_HOME/bin/kafka-topics.sh --delete --zookeeper localhost:2181 --topic my-file-source-topic
Topic my-file-source-topic is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
bash-4.4#
```