Hands-on Lab: Build a Streaming ETL Pipeline using Kafka

Estimated time needed: **45** minutes.

Project scenario

You are a data engineer at a data analytics consulting company. You have been assigned to a project that aims to de-congest the national highways by analyzing the road traffic data from different toll plazas. As a vehicle passes a toll plaza, the vehicle's data like `vehicle_id`,`vehicle_type`,`toll_plaza_id`, and timestamp are streamed to Kafka. Your job is to create a data pipe line that collects the streaming data and loads it into a database.

Objectives

In this assignment, you will create a streaming data pipe by performing these steps:

- Start a MySQL database server
- Create a table to hold the toll data
- Start the Kafka server
- Install the Kafka Python driver
- Install the MySQL Python driver
- Create a topic named toll in Kafka
- Download streaming data generator program
- Customize the generator program to steam to toll topic
- Download and customize streaming data consumer
- Customize the consumer program to write into a MySQL database table
- Verify that streamed data is being collected in the database table

Note about screenshots

Throughout this lab, you will be prompted to take screenshots and save them on your device. You will need to upload the screenshots for peer review. You can use various free screen grabbing tools or your operating system's shortcut keys (Alt + PrintScreen in Windows, for example) to capture the required screenshots. You can save the screenshots with the `.jpg` or `.png` extension.

About Skills Network Cloud IDE

Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands-on labs for course and project-related labs. Theia is an open-source IDE (Integrated Development Environment) that can be run on a desktop or on the cloud. To complete this lab, you will be using the Cloud IDE based on Theia, running in a Docker container.

Important notice about this lab environment

Please be aware that sessions for this lab environment are not persistent. A new environment is created for you every time you connect to this lab. Any data you may have saved in an earlier session will get lost. To avoid losing your data, please plan to complete these labs in a single session.

Exercise 1: Download and extract Kafka

1. Download Kafka by running the command below.

   ```
   wget https://archive.apache.org/dist/kafka/3.7.0/kafka_2.12-3.7.0.tgz
   ```

2. Extract Kafka from the zip file by running the command below.

   ```
   tar -xzf kafka_2.12-3.7.0.tgz
   ```

   Note: This command creates a directory named `kafka_2.12-3.7.0` in the current directory.

Exercise 2: Configure KRaft and start server

1. Change to the `kafka_2.12-3.7.0` directory.

   ```
   cd kafka_2.12-3.7.0
   ```

2. Generate a cluster UUID that will uniquely identify the Kafka cluster.

   ```
   KAFKA_CLUSTER_ID="$(bin/kafka-storage.sh random-uuid)"
   ```

   Note: The new cluster id generated will be used by the KRaft controller.

3. KRaft requires the log directories to be configured. Run the following command to configure the log directories passing the cluster id.

   ```
   bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c config/kraft/server.properties
   ```

4. Now that KRaft is configured, you can start the Kafka server by running the following command.

   ```
   bin/kafka-server-start.sh config/kraft/server.properties
   ```

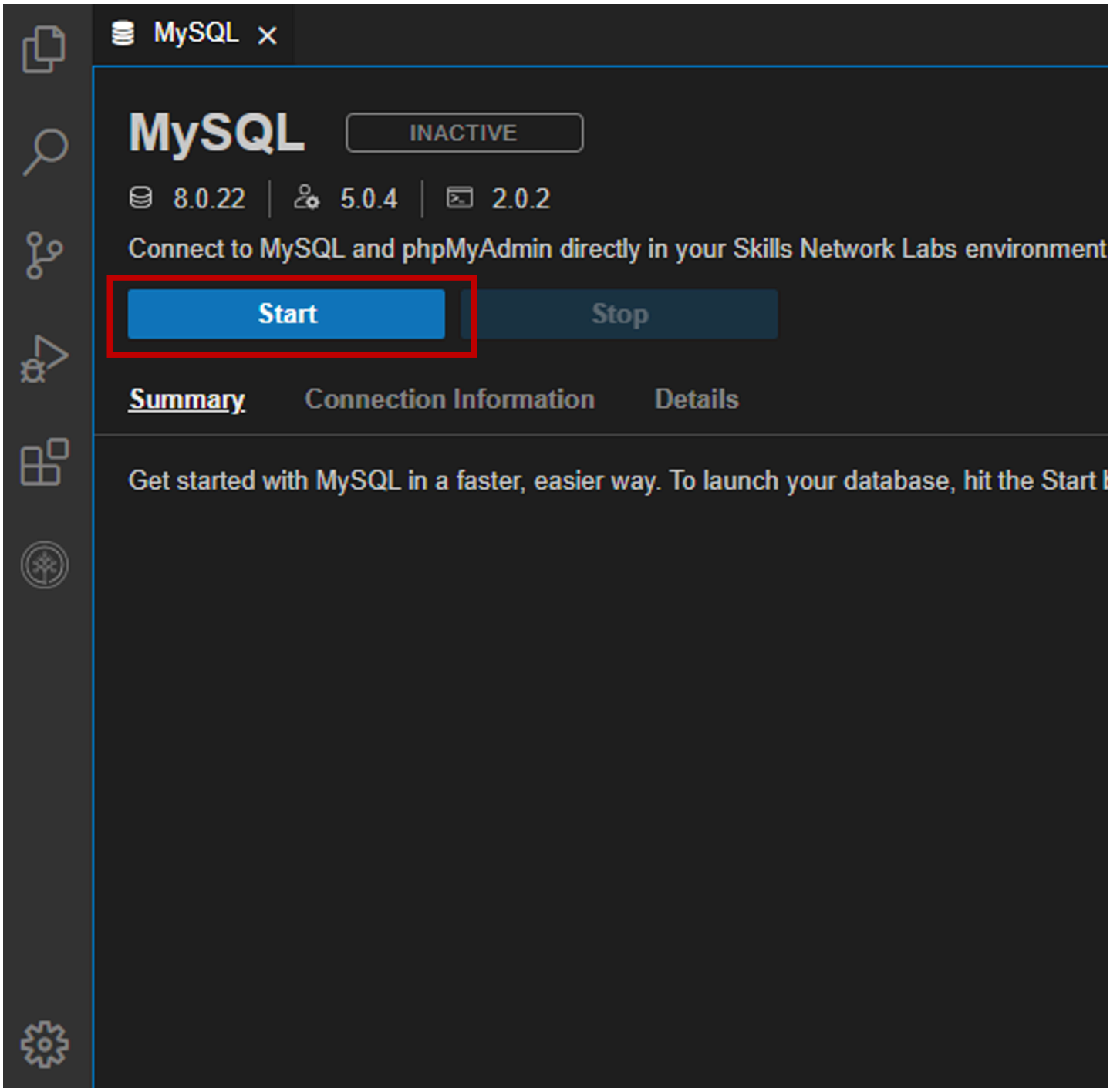
   Note: You can be sure that the Kafka server started there is information generated that the server started successfully along with some additional messages, such as log loaded.

```
[2024-06-12 02:19:51,129] INFO [BrokerServer id=1] Transition from STARTING to STARTED (kafka.server.BrokerSer
ver)
[2024-06-12 02:19:51,130] INFO Kafka version: 3.7.0 (org.apache.kafka.common.utils.AppInfoParser)
[2024-06-12 02:19:51,135] INFO Kafka commitId: 2ae524ed625438c5 (org.apache.kafka.common.utils.AppInfoParser)
[2024-06-12 02:19:51,135] INFO Kafka startTimeMs: 1718173191129 (org.apache.kafka.common.utils.AppInfoParser)
[2024-06-12 02:19:51,137] INFO [KafkaRaftServer nodeId=1] Kafka Server started (kafka.server.KafkaRaftServer)
[2024-06-12 02:20:25,678] INFO [ReplicaFetcherManager on broker 1] Removed fetcher for partitions Set(bankbran
ch-1, bankbranch-0) (kafka.server.ReplicaFetcherManager)
[2024-06-12 02:20:25,718] INFO [LogLoader partition=bankbranch-1, dir=/tmp/kraft-combined-logs] Loading produc
er state till offset 0 with message format version 2 (kafka.log.UnifiedLog$)
[2024-06-12 02:20:25,722] INFO Created log for partition bankbranch-1 in /tmp/kraft-combined-logs/bankbranch-1
 with properties {} (kafka.log.LogManager)
[2024-06-12 02:20:25,725] INFO [Partition bankbranch-1 broker=1] No checkpointed highwatermark is found for pa
rtition bankbranch-1 (kafka.cluster.Partition)
[2024-06-12 02:20:25,727] INFO [Partition bankbranch-1 broker=1] Log loaded for partition bankbranch-1 with in
itial high watermark 0 (kafka.cluster.Partition)
[2024-06-12 02:20:25,745] INFO [LogLoader partition=bankbranch-0, dir=/tmp/kraft-combined-logs] Loading produc
er state till offset 0 with message format version 2 (kafka.log.UnifiedLog$)
[2024-06-12 02:20:25,746] INFO Created log for partition bankbranch-0 in /tmp/kraft-combined-logs/bankbranch-0
 with properties {} (kafka.log.LogManager)
```

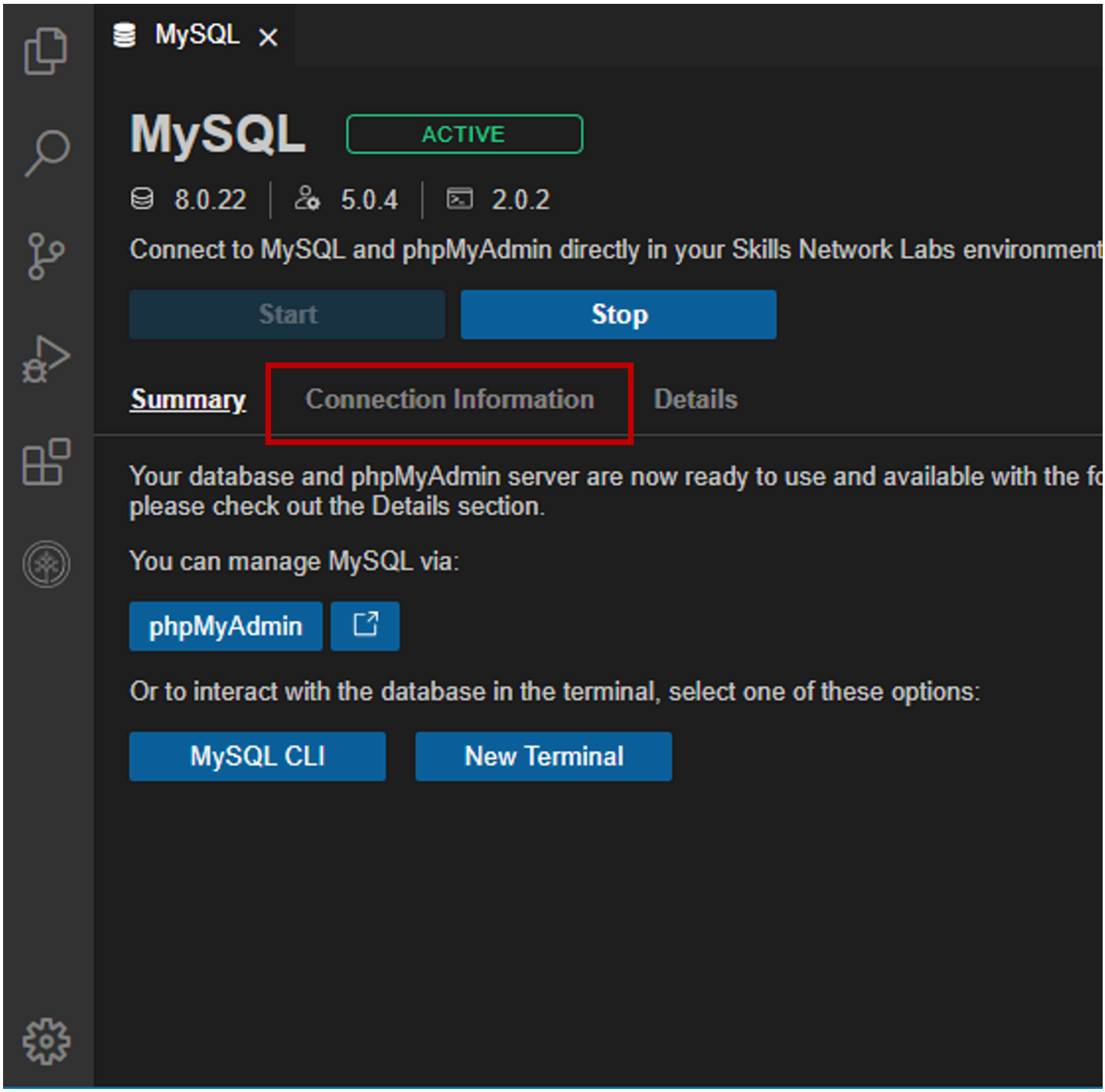
Exercise 3: Start MySQL server and setup the database

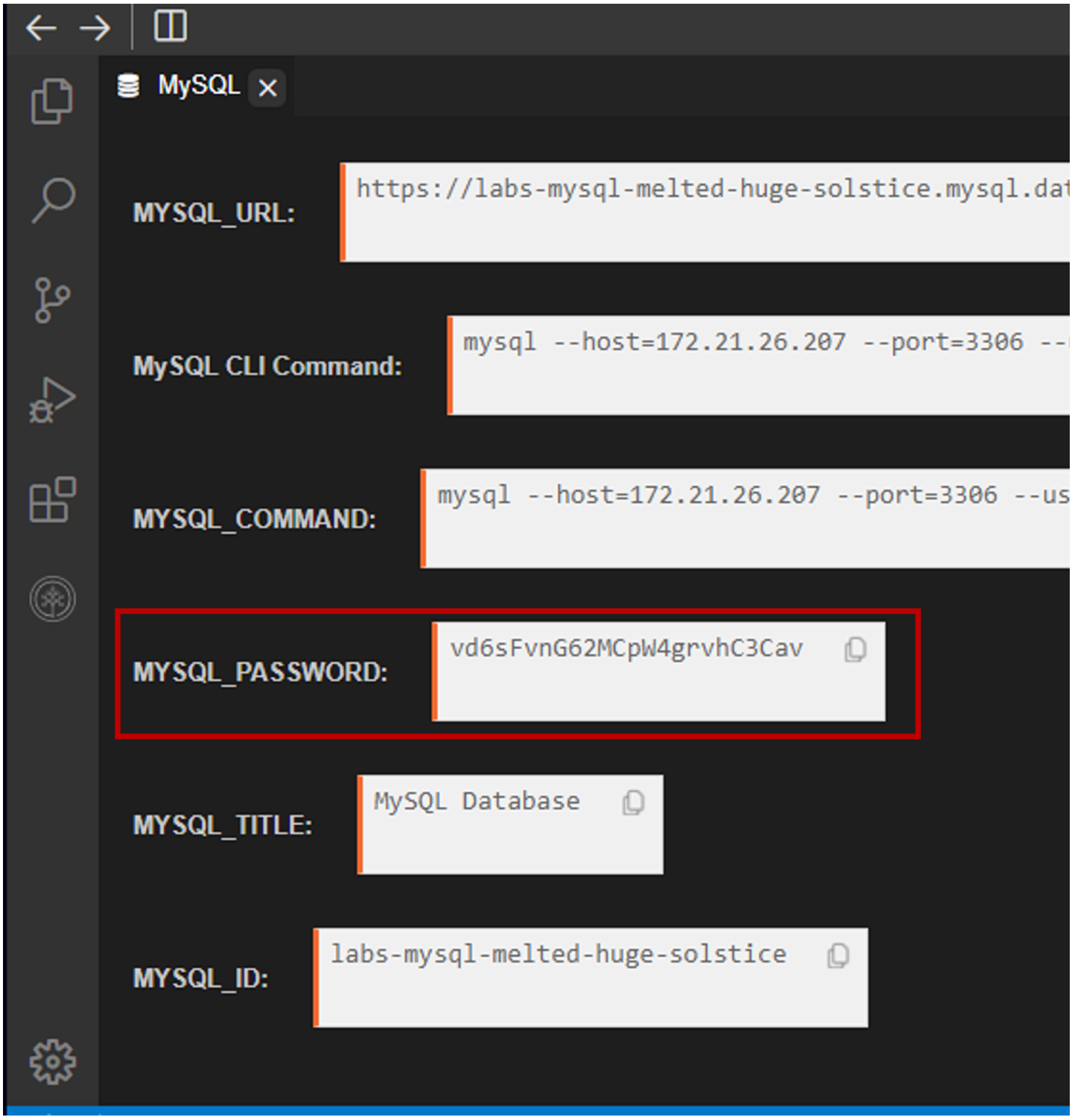
Open MySQL Page in IDE

1. On the launching page, click the **Start** button.



2. Once the MySQL server started, select the **Connection Information** tab. From that, copy the password.


MySQL
MySQL
ACTIVE
8.0.22
5.0.4
2.0.2
Connect to MySQL and phpMyAdmin directly in your Skills Network Labs environment
Start
Stop
Summary
Connection Information
Details
Your database and phpMyAdmin server are now ready to use and available with the fo
please check out the Details section.
You can manage MySQL via:
phpMyAdmin
Or to interact with the database in the terminal, select one of these options:
MySQL CLI
New Terminal



3. Connect to the MySQL server using the command below in the terminal. Make sure you use the password given to you when the MySQL server starts. Please make a note of the password because you will need it later.

```
mysql --host=mysql --port=3306 --user=root --password=Replace your password
```

4. Create a database named `tolldata`.

At the **mysql>** prompt, run the command below to create the database.

```
create database tolldata;
```

5. Create a table named `livetolldata` with the schema to store the data generated by the traffic simulator.

Run the following command to create the table:

```
use tolldata;
create table livetolldata(timestamp datetime,vehicle_id int,vehicle_type char(15),toll_plaza_id smallint);
```

Note: This is the table where you will store all streamed data that comes from Kafka. Each row is a record of when a vehicle has passed through a certain toll plaza along with its type and anonymized id.

6. Disconnect from the MySQL server.

```
exit
```

Exercise 4: Install the Python packages

1. Install the Python module `kafka-python`. This Python module will help you to communicate with kafka server. It can used to send and receive messages from Kafka.

```
pip3 install kafka-python
```

2. Install the Python module `mysql-connector-python` using the `pip` command.

```
pip3 install mysql-connector-python==8.0.31
```

This Python module will help you to interact with MySQL server.

Exercise 5: Create data pipeline for toll data

1. Create a Kafka topic named `toll`.
2. Download the `toll_traffic_generator.py` from the url given below using **wget**.

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Final%20Assignment/toll_traffic_generator.py
```

3. Open the code using the editor using the “Menu –> File –>Open” option.

4. Open the `toll_traffic_generator.py` and set the topic to `toll`.

5. Run the `toll_traffic_generator.py`.

   ```
   python3 toll_traffic_generator.py
   ```

6. Download the `streaming-data-reader.py` from the URL below using **wget**.

   ```
   wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/vVxmU5uatDowvAIKRZrFjg/streaming-data-reader.py
   ```

7. Open the `streaming-data-reader.py` and modify the following details so that the program can connect to your MySQL server.

   ```
   TOPIC

   DATABASE

   USERNAME

   PASSWORD
   ```

8. Run the `streaming-data-reader.py`.

   ```
   python3 streaming-data-reader.py
   ```

9. If you completed all the steps correctly, the streaming toll data will get stored in the table `livetolldata`. As a last step in this lab, open mysql CLI and list the top 10 rows in the table `livetolldata`.

Authors

Ramesh Sannareddy
Lavanya T S

Other Contributors

Rav Ahuja