Hands-on Lab: Kafka Python Client

Estimated time needed: **30** minutes

Objectives

After completing this lab, you will be able to:

- Use `kafka-python` to interact with Kafka server in Python
- Send and receive messages through the Kafka-python client

About Skills Network Cloud IDE

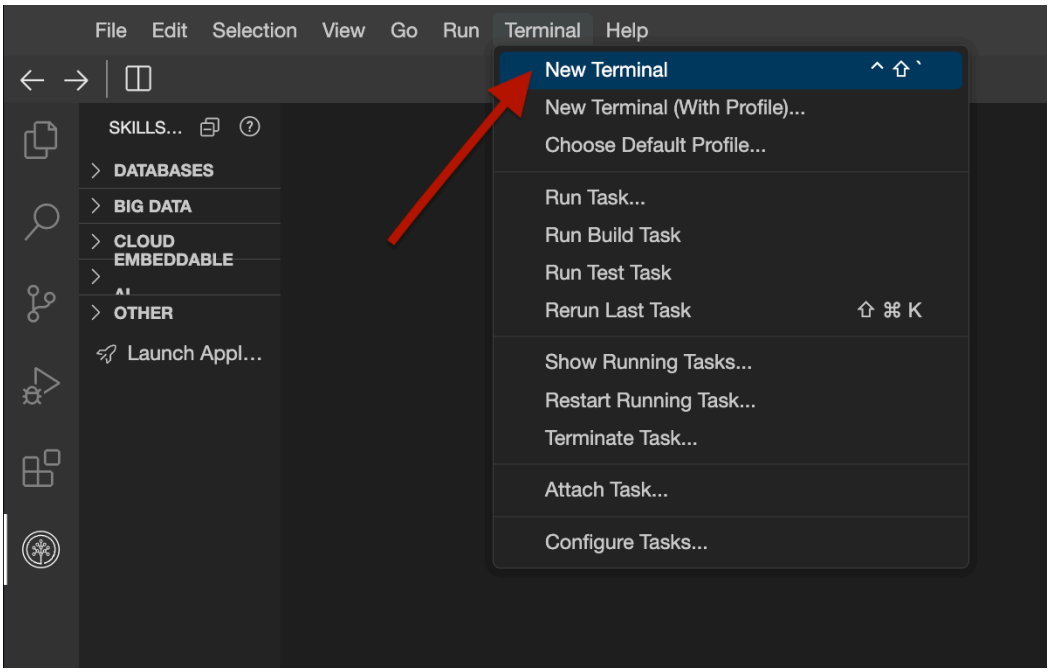
Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands on labs for course and project-related labs. Theia is an open-source IDE (Integrated Development Environment) that can be run on desktop or on the cloud. To complete this lab, we will be using the Cloud IDE based on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

Important notice about this lab environment

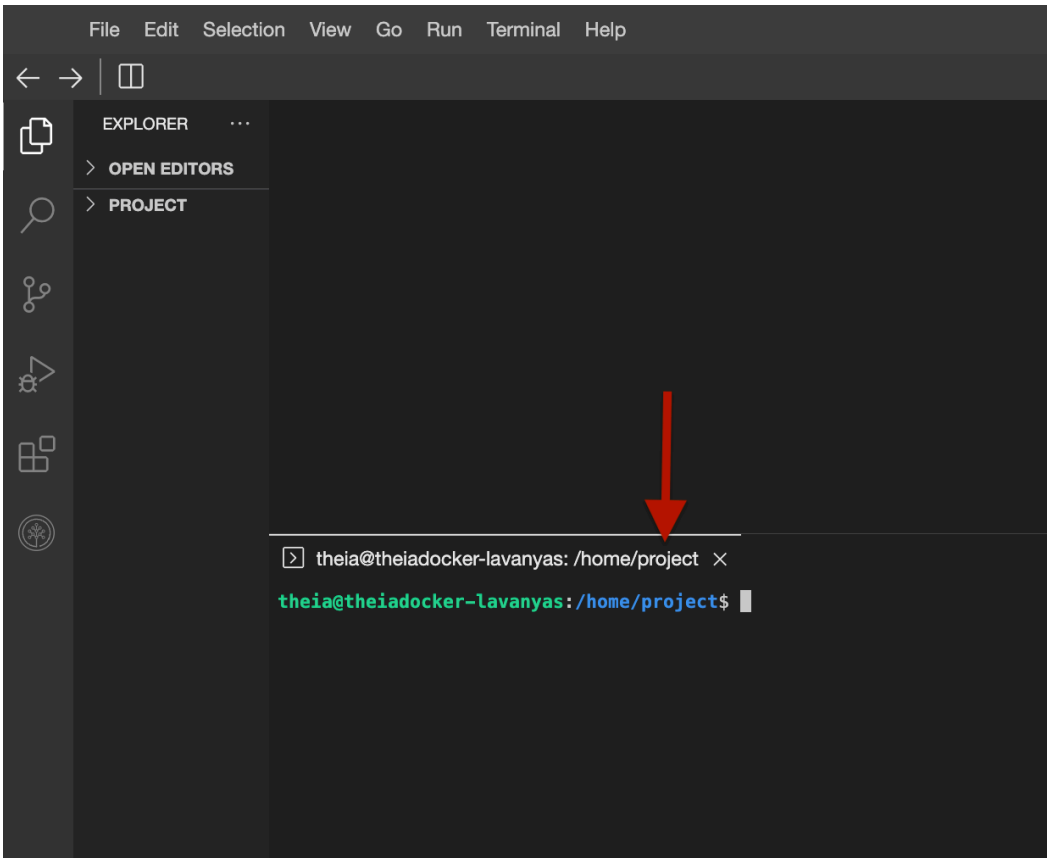
Please be aware that sessions for this lab environment are not persistent. A new environment is created for you every time you connect to this lab. Any data you may have saved in an earlier session will get lost. To avoid losing your data, please plan to complete these labs in a single session.

Exercise 1: Download and extract Kafka

1. Open a new terminal by clicking the menu bar and selecting **Terminal->New Terminal**.



This will open a new terminal at the bottom of the screen.



Run the commands below on the newly opened terminal.

Note: You can copy the code by clicking the little copy button on the bottom right of the code snippet below and then paste it, wherever you wish.

2. Download Kafka by running the command below:

```
wget https://downloads.apache.org/kafka/3.8.0/kafka_2.13-3.8.0.tgz
```

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

```
tar -xzf kafka_2.13-3.8.0.tgz
```

This creates a new directory `kafka_2.13-3.8.0` in the current directory.

Exercise 2: Configure KRaft and start server

1. Change to the `kafka_2.13-3.8.0` directory.

```
cd kafka_2.13-3.8.0
```

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

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

Note: This cluster ID 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 it has started when you see an output contains messages that confirm the Kafka Server started successfully.

```
[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: Create a topic in the admin.py file

Install kafka-python

1. Open a new terminal and navigate to the `kafka_2.13-3.8.0` directory.

```
cd kafka_2.13-3.8.0
```

2. Install the `kafka-python` package by running the following command.

```
pip3 install kafka-python
```

3. Create a file named `admin.py` by running the following command.

```
touch admin.py
```

4. Click the button below to open the file in edit mode and paste the following content in the file and save it.

Open **admin.py** in IDE

```
from kafka.admin import KafkaAdminClient,NewTopic
admin_client = KafkaAdminClient(bootstrap_servers="localhost:9092", client_id='test')
topic_list = []
new_topic = NewTopic(name="bankbranch", num_partitions= 2, replication_factor=1)
topic_list.append(new_topic)
admin_client.create_topics(new_topics=topic_list)
```

Note: We are creating a topic "bankbranch" through this code.

Exercise 4: Create the producer.py file

You need a producer to send messages to Kafka. You will find the code for the producer in the producer.py file.

1. Create a file named `producer.py` by running the following command.

```
touch producer.py
```

2. Click the button below to open the file in edit mode and paste the following content in the file and save it.

Open **producer.py** in IDE

```
from kafka import KafkaProducer
import json
producer = KafkaProducer(value_serializer=lambda v: json.dumps(v).encode('utf-8'))
producer.send("bankbranch", {'atmid':1, 'transid':100})
producer.send("bankbranch", {'atmid':2, 'transid':101})
producer.flush()
producer.close()
```

In the above code, the producer is sending across two messages through this code. These messages will be received by the consumer.

Exercise 5: Create the consumer.py file

You need a consumer to read messages from Kafka. The code for consumer will be written in the `consumer.py` file.

1. Create a file named `consumer.py` by running the following command.

```
touch consumer.py
```

2. Click the button below to open the file in edit mode and paste the following content in the file and save it.

Open **consumer.py** in IDE

```
from kafka import KafkaConsumer
consumer = KafkaConsumer('bankbranch',
                         group_id=None,
                          bootstrap_servers=['localhost:9092'],
                          auto_offset_reset = 'earliest')
print("Hello")
print(consumer)
for msg in consumer:
    print(msg.value.decode("utf-8"))
```

Exercise 6: Execute the three Python files

1. Execute `admin.py` and `producer.py` using the following commands in terminal:

```
python3 admin.py
python3 producer.py
```

2. Open a new terminal and execute the following commands to run consumer.py:

```
cd kafka_2.13-3.8.0
python3 consumer.py
```

3. Your consumer should print the messages sent by the producer as follows:

```
theia@theiadocker-lavanyas:/home/project$ python3 consumer.py
Hello
<kafka.consumer.group.KafkaConsumer object at 0x7f4e8be6bc10>
{"atmid": 1, "transid": 100}
{"atmid": 2, "transid": 101}
```

Practice Exercise

1. Create a new producer from `bankbranch` in a file named `new_producer.py` which will take user input as long as the user wants and accept user input for the ATM number they want to transact with (1 or 2) and stream the transaction.

2. Observe the consumer getting the events streamed by the producer in real time.

▼ Click here for the **solution**.

```
from kafka import KafkaProducer
import json
producer = KafkaProducer(value_serializer=lambda v: json.dumps(v).encode('utf-8'))
transid = 102
while True:
        user_input = input("Do you want to add a transaction? (press 'n' to stop): ")
        if user_input.lower() == 'n':
            print("Stopping the transactions")
            break
        else:
            atm_choice = input("Which ATM you want to transact in? 1 or 2 ")
            if (atm_choice == '1' or atm_choice == '2'):
                producer.send("bankbranch", {'atmid':int(atm_choice), 'transid':transid})
                producer.flush()
                transid = transid + 1
            else:
                print('Invalid ATM number')
                continue
producer.close()
```

Authors

Ramesh Sana Reddy
Lavanya T S
Shreya Khurana