Lab 02 Producing Messages to Kafka

a. Kafka Producer (Java, C#, Python)

The goal of this lab is to create a simple producer. The producer application reads a file of latitude and longitude values and writes to the topic `driver-positions`. See an example file at `~/confluent-dev/challenge/java-producer/drivers/driver-1.csv`. For each entry in the file the application writes a Kafka record. When the application reaches the end of the file it loops back to the top. The record is created with the driver id as the key, and the comma separated latitude and longitude as the value. For example:

Key	Value
driver-1	47.5952,-122.3316

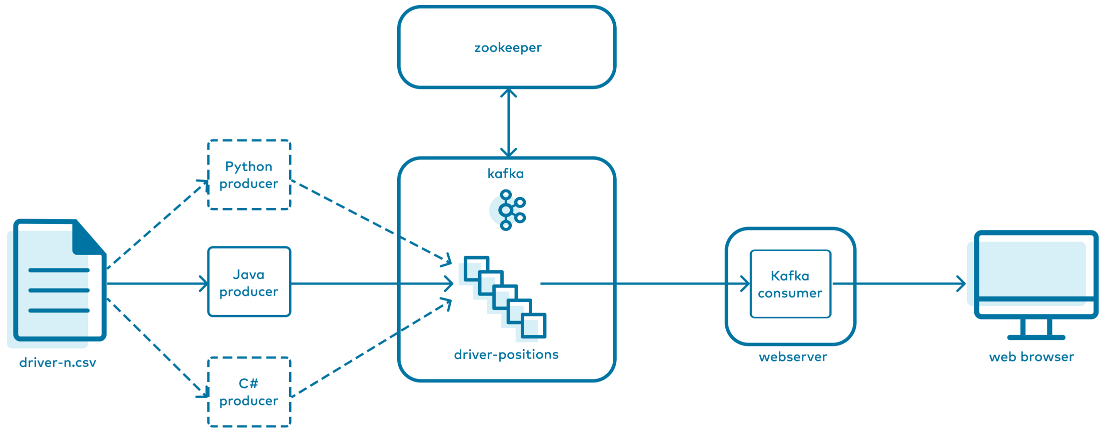
Prerequisites

1. Use the command in the table below to navigate to the project folder for your language. Click the associated hyperlink to open the API reference:

Language	Command	API Reference
Java	`cd ~/confluent-dev/challenge/java-producer`	Class KafkaProducer<K,V>
C#	`cd ~/confluent-dev/challenge/dotnet-producer`	Interface IProducer<TKey, TValue>
Python	`cd ~/confluent-dev/challenge/python-producer`	class confluent_kafka.Producer

2. Run the Kafka cluster:

```
$ docker-compose up -d zookeeper kafka control-center create-topics
webserver
```



The `create-topics` container creates the topics for all of the upcoming exercises and then exits. The `webserver` container is running a web application that is consuming from the `driver-positions` and displaying the position of each driver.

3. View the application at http://localhost:3001. You will see a driver appear on the map when you complete the code challenges in this exercise.

4. Run the `kafka-topics` command to see the new topics. All of the new topics are prefixed with `driver`:

```
$ kafka-topics --bootstrap-server kafka:9092 --describe | grep
'driver'
Topic: driver-positions-pyavro  PartitionCount: 3
ReplicationFactor: 1    Configs:
    Topic: driver-positions-pyavro  Partition: 0    Leader: 101
Replicas: 101   Isr: 101    Offline:
    Topic: driver-positions-pyavro  Partition: 1    Leader: 101
Replicas: 101   Isr: 101    Offline:
    Topic: driver-positions-pyavro  Partition: 2    Leader: 101
Replicas: 101   Isr: 101    Offline:
...
```

If any replicas were listed as offline, this would be an indication that the corresponding broker is also offline.

5. If you are completing the C# or Python exercise, install the dependencies.

   a. For C#:

      First, install dotnet running this command. You'll be prompted to enter the password: **training**

```
$ ~/confluent-dev/dotnet-install.sh
```

Then, run:

```
$ dotnet restore
```

b. For Python:

```
$ pip3 install -r requirements.txt
```

6. Open the project in Visual Studio Code. As always, make sure you open VS Code in the correct project folder as specified in step 1.

```
$ code .
```

Writing the Producer

1. Open the implementation file for your language of choice
   - Java `src/main/java/clients/Producer.java`
   - C#: `Program.cs`
   - Python: `main.py`.
2. Locate the `TODO` comments in your implementation file. Use the API reference for your language to attempt each challenge. Solutions are provided at the end of this lab and in the `~/confluent-dev/solution` folder.
3. At any time run the application by selecting the menu **Run** → **Start Debugging** in VS Code. As you complete the challenges try to produce a similar output from your application:

```
Starting Java producer.
Sent Key:driver-1 Value:47.618579,-122.355081
Sent Key:driver-1 Value:47.618577152452055,-122.35520620652974
Sent Key:driver-1 Value:47.61857902704408,-122.35507321130525
Sent Key:driver-1 Value:47.618579488930855,-122.35494018791431
Sent Key:driver-1 Value:47.61857995081763,-122.35480716452278
...
```

Are you having issues with VS code debugging?

Here are some common issues you can check for:

- Did you launch `code` from the correct directory? Ensure you've followed the `cd` command above to change to the project folder.
- Do you miss an earlier step? Quit VS Code, run the missing step, and relaunch VS Code.
- Java users: if VS Code is still having an issue after relaunching try cleaning your workspace. Click the cog wheel icon at the bottom left of VS Code, click **Command Palette**, search for and select **Java: Clean the Java language server workspace**, click **Restart and delete**.

4. When you have the application producing data, leave the application running, and return to the web application at http://localhost:3001.

If you are interested in the inner workings of the web application the source code is available at `~/confluent-dev/webserver`. The web application is a simple Node.js Express web application that contains a Kafka consumer reading from a topic and delivering the records via a Socket.IO websocket to the front end.

5. Use the `kafka-console-consumer` tool to view the data on the `driver-positions` topic:

```
$ kafka-console-consumer --bootstrap-server kafka:9092 \
     --topic driver-positions \
     --property print.key=true \
     --from-beginning
```

   Exit the consumer with `Ctrl+C`.

6. When you have completed the challenges, stop the debugger in VS Code.

Extra Challenges and Questions

1. The producer application takes the driver ID from an environment variable `DRIVER_ID`. From new terminal windows run the producer with several different driver IDs. Observe the web application at http://localhost:3001. When you have finished exit your producers with `Ctrl+C`.

   a. Java

   ```
   $ cd ~/confluent-dev/solution/java-producer && \
      DRIVER_ID=driver-3 ./gradlew run --console plain
   ```

   b. C#

   ```
   $ cd ~/confluent-dev/solution/dotnet-producer && \
      DRIVER_ID=driver-3 dotnet run
   ```

   c. Python

   ```
   $ cd ~/confluent-dev/solution/python-producer && \
      DRIVER_ID=driver-3 python3 main.py
   ```

2. Experiment with producer settings of `batch.size` (default: 16384) and `linger.ms` (default: 0 ms). How would you expect the producer to behave with the settings below? Observe the web application at http://localhost:3001.

   a. Java

   ```
   settings.put(ProducerConfig.BATCH_SIZE_CONFIG, 16384);
   settings.put(ProducerConfig.LINGER_MS_CONFIG, 5000);
   ```

   b. C#

   ```
   BatchNumMessages = 16384,
   LingerMs = 5000
   ```

   c. Python

```
'batch.num.messages': 16384,
'linger.ms': 5000
```

Java Solution

solution/java-producer/src/main/java/clients/Producer.java

```
// TODO: configure the location of the bootstrap server
settings.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka:9092");
```

```
// TODO: populate the message object
final ProducerRecord<String, String> record = new ProducerRecord
<>(KAFKA_TOPIC, key, value);
```

```
// TODO: write the lat/long position to a Kafka topic
// TODO: print the key and value in the callback lambda
producer.send(record, (md, e) -> {
  System.out.printf("Sent Key:%s Value:%s\n", key, value);
});
```

C# Solution

solution/dotnet-producer/Program.cs

```
// TODO: configure the location of the bootstrap server
BootstrapServers = "kafka:9092",
```

```
// TODO: populate the message object
var message = new Message<string, string> { Key = driverId, Value = line
};
```

```
// TODO: write the lat/long position to a Kafka topic
// TODO: configure handler as a callback to print the key and value
producer.Produce(KafkaTopic, message, handler);
```

Python Solution

solution/python-producer/main.py

```
#TODO: configure the location of the bootstrap server
'bootstrap.servers': 'kafka:9092',
#TODO: write the lat/long position to a Kafka topic
#TODO: configure delivery_report as a callback to print the key and
value
producer.produce(
    KAFKA_TOPIC,
    key=DRIVER_ID,
    value=line,
    on_delivery=delivery_report)
```

Extra Challenges and Questions Solutions

1. If you followed the steps successfully you will see multiple cars driving on the map.
2. With the supplied settings for `batch.size` and `linger.ms` the consumer is batching up records. The amount of batched data never exceeds the `batch.size`, so the batch is sent to the broker when the `linger.ms` of seconds is met. From the producer documentation:

> linger.ms: ...This setting gives the upper bound on the delay for batching: once we get batch.size worth of records for a partition it will be sent immediately regardless of this setting, however if we have fewer than this many bytes accumulated for this partition we will 'linger' for the specified time waiting for more records to show up...

STOP HERE. THIS IS THE END OF THE EXERCISE.