Clients

Use clients to produce and consume messages

# Objectives

Run different producers and consumers against your new cluster.

Use API Keys, Service Users, ACLs and RBAC to secure access to the cluster and its resources.

# Labs

## Setup

* Create a couple of topics (skip the schema creation)
  + A topic called “test-topic” with 1 partition
  + A topic called “customer” with 6 partitions
    - Have a look at the advanced settings as well
* Create a Java client configuration on the “Clients/Add New Client” page
  + Create an API Key
  + Go directly to the step 4
  + Copy the completed configuration and store it in a local file,   
    for example “ccloud.properties”

## Standard Java Clients

The Confluent platform comes with a set of standard Java clients. This part of the lab will ensure you know how to use them.

* On your local machine (laptop), run the following Java clients[[1]](#footnote-0) in separate terminals to see the produced messages in the consumer. Use the “test-topic” or your own topics for this (do not use the “customer” topic).
  + kafka-console-producer
  + kafka-console-consumer
  + Remember: You will have to specify the --bootstrap-server again for each command
  + Use the “--help” option to find out how to pass the appropriate configuration for producers, consumers and kafka-topics.

## Application with schema

The Schema Registry in the Confluent Cloud uses a separate API Key. The following is an example of a client application that communicates with both the broker and the schema registry and therefore requires additional authentication.

* Run a Java application that uses the Schema Registry, for example
  + Download or clone <https://github.com/sknop/kafka_transactions>
  + Install Maven if you have not done so already (brew install maven on a Mac)
  + Compile the applications with “mvn clean package -DskipTests”
  + Create a schema API Key and API Secret
    - Navigate to the environment overview for the environment you created (“bootcamp”)
    - In the right-hand panel on the bottom, click on the link below Credentials to create a new key
  + Update your config file to add the schema registry

| # Required connection configs for Confluent Cloud Schema Registry  schema.registry.url=https://psrc-4xrp1.eu-central-1.aws.confluent.cloud  basic.auth.credentials.source=USER\_INFO  basic.auth.user.info=<SR-KEY>:<SR-SECRET> |
| --- |

* + Run (use option -h to see all options)

| java -cp target/kafka\_transactions-2.0-SNAPSHOT.jar \  producer.CustomerProducer -c ccloud.properties -v -m 10 |
| --- |

* + You can use consumer.CustomerConsumer to read the messages (they are Avro encoded, so the normal kafka-console-consumer is not helpful)

## Run the kafka-avro-console-consumer

The kafka-avro-console-consumer is an extension of the standard kafka-console-consumer with the Confluent package.

* Run the kafka-avro-console-consumer against your Confluent Cloud instance's “customer” topic. You will need to specify “--from-beginning” to see all messages. You also need a set of additional properties to talk to and authenticate against the schema registry. You can use the “--property” option multiple times. Values to set are
  + schema.registry.url
  + basic.auth.credentials.source=USER\_INFO
  + basic.auth.user.info
  + key-deserializer (The key is an Integer)
  + print.key=true
  + key.separator=":"

## C/C++ clients - kafkacat/kcat (Optional)

kafkacat (now named kcat) is an example of a C/C++ client based on the same library as the C#/Python/etc clients. On Mac it can be installed with “brew install kcat”

* Run kafkacat/kcat). Use the New Client page and the C/C++ language to create the configuration file, for example, bootcamp-kcat.cfg.
  + Use kcat to read from your topics.

| kcat -F bootcamp-kcat.cfg -C -t test-topic |
| --- |

* + You can use kcat also to read Avro-encoded records:

| kcat -F bootcamp-kcat.cfg -C -t customer -s key=i -s value=avro |
| --- |

* + - You will have to add the schema.registry.url with api-key and api-secret to your config-file
      * schema.registry.url=[https://api-key:api-secret@URL](https://api-key:api-secret@url)

## Confluent CLI

This section uses the Confluent CLI, which you need to download and install (See references for how).

* Use Confluent CLI to consume from your “test-topic”. There is a “CLI and Tools” section at the left bottom of your Confluent Cloud UI with useful hints, or you can use the Confluent documentation.
* Use the same CLI to consume from “customer”. Now you need to specify serializers since the data is stored in Avro. Try adding the options “--value-format avro --print-key --key-format integer --delimiter "-" to see if you can get it to work.

# References

<https://docs.confluent.io/platform/current/installation/installing_cp/zip-tar.html#install-cp-using-zip-and-tar-archives>

<https://www.confluent.io/en-gb/blog/using-apache-kafka-command-line-tools-confluent-cloud/>

<https://docs.confluent.io/confluent-cli/current/install.html#install-confluent-cli>

<https://docs.confluent.io/confluent-cli/current/beginner-cloud.html>

<https://developer.confluent.io/tutorials/kafka-console-consumer-producer-avro/confluent.html>

<https://docs.confluent.io/platform/current/tools/kafkacat-usage.html>   
(Note that you can download the configuration from your Confluent Cloud cluster / New Client page and use the option -F with that file, rather than passing all parameters in on the command line).

# Expected Outcomes

Access Confluent Cloud cluster and produce to and consume from topics using different clients.

Understand how API Keys are used for authentication.

# Check your understanding

This colour marks advanced questions.

* What is an API Key?
* How and why do the configurations for Java and C/C++ clients differ?
* Why are the API Keys for the Kafka cluster and the Schema Registry different?

# 

1. See references for the location of the client tar file, if not installed already [↑](#footnote-ref-0)