Getting started with Kafka Lab

Let's do a simple lab showing how to use producers and consumers from the Kafka command line.

These files should be setup on your virtual box image. You do the work for this lab in the directory `~/kafka-training/lab1`. You can find the latest versions of the instructions for Lab1 here.

If you prefer to run the examples on another OS, e.g., OSX, please refer to the Kafka course notes for instructions on how to download labs and run them on OSX.

Note: later versions will likely work, but this was example was done with 1.0.0.0. The Kafka 1.0.0.0 just came out in November 2017. The course was recently upgraded to 1.1.0.

If you are using the Virtual Box image of Linux, we unzipped the Kafka download and put it in `~/kafka-training/`, and then renamed the Kafka install folder to `kafka`. Please do the same if you decide to install Kafka yourself.

You should be using the VirtualBox image.

Next, we are going to run *ZooKeeper* and then run *Kafka Server/Broker*. We will use some Kafka command line utilities, to create Kafka topics, send messages via a producer and consume messages from the command line.

You do the work for this lab in the directory `~/kafka-training/lab1`.

Run ZooKeeper for Kafka

Kafka relies on ZooKeeper. To keep things simple, we will use a single ZooKeeper node.

Kafka provides a startup script for ZooKeeper called `zookeeper-server-start.sh` which is located at `~/kafka-training/kafka/bin/zookeeper-server-start.sh`.

The Kafka distribution also provide a ZooKeeper config file which is setup to run single node.

To run ZooKeeper, we create this script in `kafka-training` and run it.

~/kafka-training/run-zookeeper.sh

```
#!/usr/bin/env bash
cd ~/kafka-training

kafka/bin/zookeeper-server-start.sh \
   kafka/config/zookeeper.properties
```

Run run-zookeeper.sh

```
~/kafka-training
$ ./run-zookeeper.sh
```

Wait about 30 seconds or so for ZooKeeper to startup.

Run Kafka Server

Kafka also provides a startup script for the Kafka server called `kafka-server-start.sh` which is located at `~/kafka-training/kafka/bin/kafka-server-start.sh`.

The Kafka distribution also provides a Kafka config file which is setup to run Kafka single node, and points to ZooKeeper running on `localhost:2181`.

To run Kafka, we created the script `run-kafka.sh` in `kafka-training`. Please review it and then run it in another terminal window.

~/kafka-training/run-kafka.sh

```
#!/usr/bin/env bash
cd ~/kafka-training

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

- ***ACTION*** Run the script.

Run run-kafka.sh

```
~/kafka-training
$ ./run-kafka.sh
```

Wait about 30 seconds or so for Kafka to startup.

Now let's create the topic that we will send records on.

Create Kafka Topic

Kafka also provides a utility to work with topics called `kafka-topics.sh` which is located at `~/kafka-training/kafka/bin/kafka-topics.sh`.

You will use this tool to create a topic called `my-topic` with a replication factor of 1 since we only have one server. We will use thirteen partitions for `my-topic`, which means we could have up to 13 Kafka consumers.

To run Kafka, finish creating this script in `kafka-training\lab1`, and run it in another terminal window.

~/kafka-training/lab1/create-topic.sh

```
#!/usr/bin/env bash

cd ~/kafka-training

# Create a topic
kafka/bin/kafka-topics.sh --create \
  --zookeeper localhost:2181 \
```

```
  --replication-factor 1 --partitions 13 \
  --topic my-topic
```

- ***ACTION*** Edit the file ~/kafka-training/lab1/create-topic.sh so that it creates a topic called my-topic.
- ***ACTION*** Run `create-topic.sh` from a new terminal window.

Run create-topic.sh from ~/kafka-training/lab1

```
~/kafka-training/lab1

$ ./create-topic.sh

Created topic "my-topic".
```

Notice we created a topic called `my-topic`.

List Topics

You can see which topics that Kafka is managing using `kafka-topics.sh` as follows.

Finish creating the script in `~/kafka-training/lab1/list-topics.sh` and run it.

~/kafka-training/lab1/list-topics.sh

```
#!/usr/bin/env bash

cd ~/kafka-training

# List existing topics
kafka/bin/kafka-topics.sh --list \
    --zookeeper localhost:2181
```

Notice that we have to specify the location of the ZooKeeper cluster node which is running on `localhost` port `2181`.

- ***ACTION*** Edit the file ~/kafka-training/lab1/list-topic.sh so that it lists all of the topics in Kafka.
- ***ACTION*** Run `list-topic.sh` from a new terminal window.

Run list-topics.sh run from ~/kafka-training/lab1

```
~/kafka-training/lab1
$ ./list-topics.sh
__consumer_offsets
_schemas
my-example-topic
my-example-topic2
```

```
my-topic
new-employees
```

You can see the topic `my-topic` in the list of topics.

Run Kafka Producer Console

The Kafka distribution provides a command utility to send messages from the command line. It start up a terminal window where everything you type is sent to the Kafka topic.

Kafka provides the utility `kafka-console-producer.sh` which is located at `~/kafka-training/kafka/bin/kafka-console-producer.sh` to send messages to a topic on the command line.

Finish creating the script in `~/kafka-training/lab1/start-producer-console.sh` and run it.

~/kafka-training/lab1/start-producer-console.sh

```
#!/usr/bin/env bash
cd ~/kafka-training

kafka/bin/kafka-console-producer.sh \
    --broker-list localhost:9092 \
    --topic my-topic
```

Notice that we specify the Kafka node which is running at `localhost:9092`.

- ***ACTION*** Edit the file ~/kafka-training/lab1/start-producer-console.sh so that it starts the Kafka producer.
- ***ACTION*** Run `start-producer-console.sh` from a new terminal window.

Run start-producer-console.sh and send at least four messages

```
~/kafka-training/lab1
$ ./start-producer-console.sh
This is message 1
This is message 2
This is message 3
Message 4
Message 5
```

In order to see these messages, we will need to run the consumer console.

Run Kafka Consumer Console

The Kafka distribution provides a command utility to see messages from the command line. It displays the messages in various modes.

Kafka provides the utility `kafka-console-consumer.sh` which is located at `~/kafka-training/kafka/bin/kafka-console-producer.sh` to receive messages from a topic on the command line.

Finish creating the script in `~/kafka-training/lab1/start-consumer-console.sh` and run it.

~/kafka-training/lab1/start-consumer-console.sh

```
#!/usr/bin/env bash
cd ~/kafka-training

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

Notice that we specify the Kafka node which is running at `localhost:9092` like we did before, but we also specify to read all of the messages from `my-topic` from the beginning `--from-beginning`.

- ***ACTION*** Edit the file ~/kafka-training/lab1/start-consumer-console.sh so that it starts the Kafka console consumer.
- ***ACTION*** Run `start-consumer-console.sh` from a new terminal window.

Run start-consumer-console.sh in another terminal

```
~/kafka-training/lab1
$ ./start-consumer-console.sh
Message 4
This is message 2
This is message 1
This is message 3
Message 5
Message 6
Message 7
```

Notice that the messages are not coming in order. This is because we only have one consumer so it is reading the messages from all 13 partitions. Order is only guaranteed within a partition.

Review of using Kafka from the command line

What server do you run first?

You need to run ZooKeeper than Kafka.

What tool do you use to create a topic?

kafka-topics.sh

What tool do you use to see topics?

kafka-topics.sh

What tool did we use to send messages on the command line?

kafka-console-producer.sh

What tool did we use to view messages in a topic?

kafka-console-consumer.sh

Why were the messages coming out of order?

The messages were being sharded among 13 partitions.

How could we get the messages to come in order from the consumer?

We could use only one partition or start up 13 consumers.

More about Kafka

To learn about Kafka see Kafka architecture, Kafka topic architecture and Kafka producer architecture.

Related content

- What is Kafka?
- Kafka Architecture
- Kafka Topic Architecture
- Kafka Consumer Architecture
- Kafka Producer Architecture
- Kafka Architecture and low level design
- Kafka and Schema Registry
- Kafka and Avro
- Kafka Ecosystem
- Kafka vs. JMS
- Kafka versus Kinesis
- Kafka Tutorial: Using Kafka from the command line
- Kafka Tutorial: Kafka Broker Failover and Consumer Failover
- Kafka Tutorial
- Kafka Tutorial: Writing a Kafka Producer example in Java
- Kafka Tutorial: Writing a Kafka Consumer example in Java
- Kafka Architecture: Log Compaction

About Cloudurable

We hope you enjoyed this article. Please provide feedback. Cloudurable provides Kafka training, Kafka consulting, Kafka support and helps setting up Kafka clusters in AWS.