**Working with Kafka CLI**

ssh student@35.246.154.45, ssh student@34.107.117.239 or ssh student@35.246.196.7

Passw: OroJul21

To see the Kafka version:

/opt/kafka/bin/kafka-broker-api-versions.sh --bootstrap-server localhost:9092 --version

Internal IP’s for the brokers: 10.156.0.12 (Zookeeper also on this node) , 10.156.0.13, 10.156.0.14

**Type in the command prompt: /tmp/kafka/bin/kafka-topics.sh (help with all the possible commands and options will be displayed)**

**Lets see all the topics created so far**

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --list

OR The same can be achieved with the below command, since all brokers are aware of the cluster metadata and can route the request to Zookeeper.

/opt/kafka/bin/kafka-topics.sh --list --bootstrap-server localhost:9092

**And now let’s create our own Topic: name, partitions number, replication factor**

**Please see** [**here**](https://docs.confluent.io/current/installation/configuration/topic-configs.html) **all topics configs param.**

**# Create a topic**

**kafka/bin/kafka-topics.sh --create \**

**--zookeeper 10.156.0.12:2181 \**

**--replication-factor 1 --partitions 13 \**

**--topic my-topic**

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --create --partitions 3 --replication-factor 4 --topic exercitii **// will generate error given that we have only 3 brokers (you cannot create a topic with a replication factor greater than the number of brokers)**

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --topic exercitii --create --partitions 4 --replication-factor 3

OR the same can be achieved with the below command, since all brokers are aware of the cluster metadata and can route the request to Zookeeper.

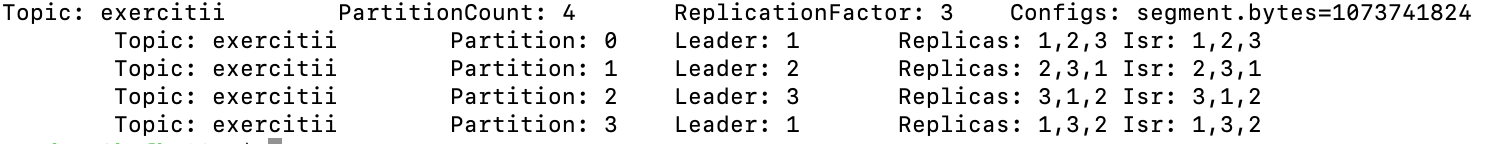
/opt/kafka/bin/kafka-topics.sh --topic exercitii --create --partitions 3 --replication-factor 3 --bootstrap-server localhost:9092

**Let’s see if our topic was created**

/opt/kafka/bin/kafka-topics.sh --list --bootstrap-server localhost:9092

**Let’s see details of a topic**

/opt/kafka/bin/kafka-topics.sh --bootstrap-server localhost:9092 --topic exercitii --describe



**Let’s create another topic with a RF<=3**

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --create --partitions 4 --replication-factor 2 --topic exercitii1

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --list **// to see that the topic was created**

**We can see details of all the topics:**

/opt/kafka/bin/kafka-topics.sh --describe --bootstrap-server localhost:9092

**Topics have lots of options that can be set at creation/altered later. See: /tmp/kafka/bin/kafka-topics.sh (in the --config section you will find these options).**

**Let’s create a topic with min.insync.replicas= 2**

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --create --partitions 3 --replication-factor 3 --config min.insync.replicas=2 --topic exercitii2

**// Note**: default value for min.insync.replicas:1

If you use --describe now you will see the min.insync.replicas value - since its something different than the default

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --topic exercitii2 --describe

And let's change the partition number for the topic (**please note** that you can increase the number of partitions for an existing topic but you cannot decrease it. Be aware: changing the number of partitions will change the keys distributions inside the partitions.)

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --alter --topic exercitii3 --partitions 4

**Let’s delete a topic**

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --delete --topic exercitii1

**Delete.topic.enable is set to true, thus the topic will be deleted (per broker config file: server.properties in /tmp/kafka/config directory). By default true starting with Kafka 0.10.0**

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --list

---------------------------------------------**--STOP HERE** ----------------------------------------------------------

**Producer Console**

Type: /opt/kafka/bin/kafka-console-producer.sh

//broker list and topic are required fields

**Let’s start a producer in order to send messages for topic exercitii and then start typing messages. Use enter for a new message in the produce console and CTRL+C for exiting the console**

/opt/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic exercitii

This is a kafka course

My first message

Awesome

:-)

CTRL+C

**Let’s start sending messages in a non existing topic.**

/opt/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic new\_exercitii

**WARN Error while fetching metadata with correlation id 1 : {new-topic=LEADER\_NOT\_AVAILABLE} (org.apache.kafka.clients.NetworkClient)**

**(by default auto.create.topics.enable = true)**

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --list

/opt/kafka/bin/kafka-topics.sh --zookeeper 10.156.0.12:2181 --topic new\_exercitii --describe **// the result will be a topic with 1 partition and 1 RF ( default settings)**

**Note: This default can be changed in the server.properties file, num\_partitions from 1 to 3 (or whatever the choice is).**

**Let’s write into the topic exercitii3 ( with set ISR 2): - we can write with default ack=1 (one node - the partition leader acknowledges the write only)**

/opt/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic exercitii2

**Or the producer can send data with ack all (meaning that in this case min 2 nodes need to ack the write before the whole operation is successful):**

/opt/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --request-required-acks all --topic exercitii2

----------------------------------------------**-STOP HERE** ----------------------------------------------------------

**Kafka Console Consumer**

/opt/kafka/bin/kafka-console-consumer.sh

//bootstrap server = broker and topic are required fields

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii

**//will display no messages (by default - if the consumer does not have an established offset to consume from - it will start reading from the latest offset, so in this case will process nothing. We have to indicate from-beginning or reset the offset to earliest) . There is a config parameter auto.offset.rest = earliest/latest possible values. For Kafka consumers the default value is the latest.**

**Open a new CLI tab and connect to the server again and start typing new messages in the exercitii topic (through producer console) - the new messages will appear immediately in the consumer console.**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii

**CTRL+C to exit the current consumer in the first window.**

**So, how do we read all the messages from the producer?**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii --from-beginning

**//Will show all messages. If you continue typing in the second window you will see the messages in real time**

**We can print the key and the timestamp as well. Unfortunately, the kafka console consumer doesn’t display partitions and offset as well (each Kafka message has besides the key(optional) and value, timestamp, partition, offset, topic info included).**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii --from-beginning --property print.timestamp=true

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii --from-beginning --property print.key=true

----------------------------------------------**-STOP HERE** ----------------------------------------------------------

[**Consumer Group mode**](https://docs.cloudera.com/documentation/enterprise/latest/topics/kafka_admin_cli.html)

**Lets see all consumers also from the console**

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list

**Important note:** When you’re working from the console and using kafka-console-consumer without group.id (like we did before), a new group.id is generated using: console-consumer-${new Random().nextInt(100000)}. So unless you use the same group.id afterwards, it would be as if you create a new consumer group each time.

**Now let’s create an explicit consumer from new group 1234 that reads from topic exercitii.**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii --group 1234

**If I continue typing in the producer all the messages will be visible.**

Note:

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii --group 1234 --property print.timestamp=true **// you can see the timestamp of the messages. Please see here a list of the** [**not so common CLI commands**](https://idarlington.github.io/2018/Not-so-common-Kafka-cli-commands/) **of the console.**

**Let’s open a new consumer (new SSH window) we will see that the new messages we type in producer console for first-topic will be split between the 2 consumers**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii --group 1234

**Now, we close (use CTRL+C, don’t just close the window) one of the consumers. We start typing more messages in the first-topic producer and we will see that the messages are sent to the existing active consumer.**

**Lets see all consumers open for this group**

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group 1234

**Now, we close (use CTRL+C, don’t just close the window) another of the exercitii consumer consoles.**

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group 1234

**If I go back and type some more in the producer console and I produce 4 more messages and I don’t consume them ( so no consumer is open).**

Consumer group '1234' has no active members.

GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID

1234 exercitii 0 12 13 **1**  - - -

1234 exercitii 1 11 13  **2**  - - -

1234 exercitii 2 12 13  **1**  - - -

**And if we open a new consumer, only the 4 new values will be displayed.**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii --group 1234

**We can see now in another tab which are the live consumers in the 1234 group:**

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group 1234 --members

**Note:** we can continue typing new data in our topic; all messages will be visible to the only consumer active in the 1234 group . A new consumer in a new group will start reading from active offset without any relation to the group 1234.

**-----------------------------END HERE -------------------------------**

**By default a new consumer in a new group will read from the latest offset in the topic’s partitions, but there are ways to define a different offset:**

**Let’s say we want to reset the offsets to the earliest and read all messages from beginning:**

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --reset-offsets --to-earliest --execute --group 1234 --topic exercitii

Error: Assignments can only be reset if the group '1234' is inactive, but the current state is Stable.

**Let’s close the consumer CTRL+C and run again**

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --group 1234 --topic exercitii --reset-offsets --to-earliest --execute

GROUP TOPIC PARTITION NEW-OFFSET

1234 exercitii 0 0

1234 exercitii 2 0

1234 exercitii 1 0

**If we open a new consumer in the group all messages will be displayed:**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii --group 1234

**We can reset offsets to a particular offset - so that all consumers start reading from that particular offset on:**

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --group 1234 --topic exercitii --reset-offsets --to-offset 9 --execute

**Now, when we open a new consumer in the group only part of the messages will be displayed:**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic exercitii --group 1234

**Lets see what is the offset situation on the 1234 group:**

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group 1234

Open the producer console and type some more data in exercitii topic. Consume the data and let’s see the offsets situation:

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server 10.156.0.27:9092 --describe --group 1234

And now let's try to move the offsets per partition with 4 offsets backwards

/opt/kafka/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --group 1234 --topic exercitii --reset-offsets --shift-by -4 --execute

Producer Console - sending Keys together with messages

Class exercise: Create a new topic (e.g. topic-keys) , 1 partitions , replication 1. List topics to verify it has been created.

**Let’s introduce messages with keys and values in Kafka**

/opt/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --property "parse.key=true" --property "key.separator=:" --topic topic-keys

**We intro:**

**1:a**

**2:b**

**3:c**

**4:d**

**1:b**

**3:f**

**2:c**

**5:d**

**Let’s open a consumer and read the data - you can see the order has been preserved at topic level:**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --property print.key=true --property key.separator=: --from-beginning --topic topic-keys

Class exercise: Create a new topic (e.g. topic-keys2), with 3 partitions , replication 1. List topics to verify it has been created.

/opt/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic topic-keys2 --property "parse.key=true" --property "key.separator=:"

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic topic-keys2 --property print.key=true --property key.separator=- --from-beginning  **//see order of the data is different.**