# Apache Kafka hands-on demo

This article is a continuation of a previous article in this series called "Apache Kafka in a nutshell" written by Vajo Lukic. After we have covered basic theory of Kafka, now is the time to get our hands on Kafka and try it out.

By following simple instructions below, you will be able to learn in no time how to create and change Kafka topics, as well how to write to them and read from them.

## Prerequisites

- Docker
- Kafka docker image "Landoop fast-data-dev"
- Java 8
- Some java IDE (Intellij, Eclipse...)
- Maven

In this demo we are going to run Kafka in a Docker container. If for some reason you cannot use Docker, you can install Kafka locally. Just follow instructions in the section "Kafka local installation".

## Get Docker

If you do not have Docker already, register and create your docker id. Then go and get Docker from here: Docker download.

### Docker for Mac

Install Docker for Mac or Install docker for Mac using Homebrew.

### Docker for Linux

Install Docker on Ubuntu

### Docker for Windows

Docker requires Windows 10 64bit **Pro, Enterprise or Education**. If you do not have "Pro" version of Windows, then install "Docker Toolbox" instead (see the next section).

Install Docker Desktop for Windows

### Docker Toolbox

This is for older versions of Mac and Windows or for other versions than "Pro".

**Important!!!** When you run Kafka image in Docker Toollbox, your "localhost" IP address is: 192.168.99.100

Install Docker Toolbox

## Get Docker Kafka image and run it

We will use **Landoop fast-data-dev** image which contains entire Kafka distribution with Apache Kafka, Kafka Connect, Zookeeper, Confluent Schema Registry and REST Proxy. More details about the image can be found here Landoop fast-data-dev.

If you have installed Docker, then just type :

```
docker run --rm --net=host landoop/fast-data-dev
```

After you have get Docker image and run it, to open GUI and go into fast-data-dev environment visit : `http://localhost:3030`

### For Mac and Windows

**!!! Important !!!**

On Mac and on widows `--net=host` parameter might not work. In that case, you should get an image and start it in Docker with another command:

```
docker run --rm -p 3030:3030 -p 9092:9092 -p 8081:8081 -p 8083:8083 -p
8082:8082 -p 2181:2181 landoop/fast-data-dev
```

or alternatively:

```
docker run --rm -it -p 2181:2181 -p 3030:3030 -p 8081:8081 -p 8082:8082
-p 8083:8083 -p 9092:9092 -e ZK_PORT=2181 -e WEB_PORT=3030 -e
REGISTRY_PORT=8081 -e REST_PORT=8082 -e CONNECT_PORT=8083 -e
BROKER_PORT=9092 -e ADV_HOST=127.0.0.1 landoop/fast-data-dev
```

Then to go into fast-data-dev environment use this link : `http://localhost:3030`

Here is the explanation of this problem and the solution Zookeeper would not start on Mac.

For Mac especially, you might also want to increase memory for Docker : how to increase docker-machine memory for Mac

## Kafka CLI hands-on

Here is a series of exercises to try some common Kafka operations by using CLI (command line interface).

To make it possible to run Kafka commands there are three options you can choose from :

1. You can "bash" into running Landoop Docker image:
   - First run this command in a terminal to find image name : `docker ps`
   - Then "bash" into image by typing:
     `docker exec -it <image name> /bin/bash`
   - And you are set to go and run Kafka commands like this for example : `kafka-topics --zookeeper 127.0.0.1:2181 --list` - notice that you do not need **".sh"** at the end if you run commands like this.
2. Run Kafka commands from your terminal through Docker. Here is one example :

   ```
   docker exec -it <image name> bash /opt/landoop/kafka/bin/kafka-topics --zookeeper 127.0.0.1:2181 --list
   ```

3. Or consider installing Kafka client. Find the instructions how to do it in this section below : "**Kafka local installation**".

### Create a new Kafka topic

Open your favorite terminal application and type in following commands:

- List all current topics :
  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --list
  ```
- Create a new topic :
  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic first_topic --create
  ```
- Add partitions and try again :
  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic first_topic --create --partitions 3
  ```
- Add replication factor and try again :
  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic first_topic --create --partitions 3 --replication-factor 2
  ```
- Adjust replication factor and try again :
  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic first_topic --create --partitions 3 --replication-factor 1
  ```

- Describe the topic :
  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic first_topic --describe
  ```
- Delete the topic :
  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic first_topic –delete
  ```

## Producing messages to a topic

- Open new editor window and start Kafka console producer for a topic named "first_topic":

  ```
  kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic first_topic
  ```

- Write some text to a topic : hello there, awesome demo, learning Kafka, just another message ...

- Producing to a **non existing** topic:

  ```
  kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic new_topic
  ```
  Write some tesxt to it : hello world!

- This results in creation of a "new_topic":

  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --list
  ```

- Let's describe it:

  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic new_topic --describe
  ```
  - Increase default number of paritions to 3 : - `edit config/server.properties` - `num.partitions=3`

- Produce to a non existing topic again:

  ```
  kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic new_topic_2
  ```

- Type some message : "hello again!", "more paritions", "please!"...

- This time our topic has 3 partitions :

  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic new_topic_2 –describe
  ```

### Consuming messages from a topic

- Open new editor tab in the same window and start Kafka console consumer to listen to messages from "first_topic":

  ```
  kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topic
  ```

- Continue typing messages in producer tab window, and observe how they are appearing in consumer window.
- To consume all messages from beginning, open a new tab and start a new consumer there :

  ```
  kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topic --from-beginning
  ```

## Consuming in Consumer groups

- Open new terminal window and start new consumer in a group named "my-first-application" :

  ```
  kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topic --group my-first-application
  ```

- Open new tab in the same window and start another consuer in the same group :

  ```
  kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topic --group my-first-application
  ```

- Open another tab and start a producer :

  ```
  kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic first_topic
  ```

- Start producing messages and observer how they get separated by two consumers who are part of the same consumer group. - Open new tab and start a single consumer in second consumer group :

  ```
  kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topic --group my-second-application --from-beginning
  ```

- This consumer has read all messages from the beginning.

### Describing Consumer groups

- List consumer groups :

```
kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list
```

- Describe one group :

```
kafka-consumer-groups.sh --bootstrap-server localhost:9092 --
describe --group my-first-application
```

- Describe another group :

```
kafka-consumer-groups.sh --bootstrap-server localhost:9092 --
describe --group my-second-application
```

## Resetting offsets

- Take a look at the documentation :

```
kafka-consumer-groups.sh
```

- Reset the offsets to the beginning of each partition :

```
kafka-consumer-groups.sh --bootstrap-server localhost:9092 --
group my-first-application --reset-offsets --to-earliest --
execute --topic first_topic
```

- Consume from where the offsets have been reset :

```
kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --
topic first_topic --group my-first-application
```

- Describe the group again :

```
kafka-consumer-groups.sh --bootstrap-server localhost:9092 --
describe --group my-first-application
```

- Experiment by shifting offsets by 2 (forward) :

```
kafka-consumer-groups.sh --bootstrap-server localhost:9092 --
group my-first-application --reset-offsets --shift-by 2 --execute
--topic first_topic
```

- Experiment by shifting offsets by 2 (backward) :

```
kafka-consumer-groups.sh --bootstrap-server localhost:9092 --
group my-first-application --reset-offsets --shift-by -2 --
execute --topic first_topic
```

- Consume messages again :

```
kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --
topic first_topic --group my-first-application
```

### Log Compaction

- Open a new terminal window and type following commands. - Let's first create new topic with log compaction policy :

  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --create --topic
  employee-salary --partitions 1 --replication-factor 1 --config
  cleanup.policy=compact --config min.cleanable.dirty.ratio=0.001 -
  -config segment.ms=5000
  ```

- Describe the topic :

  ```
  kafka-topics.sh --zookeeper 127.0.0.1:2181 --describe --topic
  employee-salary
  ```

- Start a consumer in a new tab :

  ```
  kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --
  topic employee-salary --from-beginning --property print.key=true
  --property key.separator=,
  ```

- Start a producer and create some messages :

  ```
  kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic
  employee-salary --property parse.key=true --property
  key.separator=,
  ```

- Example of messages: Mats,salary: 1000, Peter,salary: 2000, John,salary: 1500, Mats,salary: 4000, John,salary: 7000

Stop a consumer and start it again to see the effects of log compaction.

## Kafka local installation

If for some reason docker is not an option or you want to try Kafka locally, here are quick instructions how to do that in different environments.

### Mac OS X local installation

- Install brew if needed : `/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`
- Download and Setup Java 8 JDK:
- `brew tap caskroom/versions`
- `brew cask install java8`
- Download & Extract the Kafka binaries from : `https://kafka.apache.org/downloads`

- Install Kafka commands using brew : `brew install kafka`
- Try Kafka commands using `kafka-topics (for example)`
- Edit Zookeeper & Kafka configs using a text editor
- `zookeeper.properties: dataDir=/your/path/to/data/zookeeper`
- `server.properties: log.dirs=/your/path/to/data/kafka`

Start Zookeeper in one terminal window: `zookeeper-server-start config/zookeeper.properties`

Start Kafka in another terminal window: `kafka-server-start config/server.properties`

## Linux local installation

- Download and Setup Java 8 JDK: `sudo apt install openjdk-8-jdk`
- Download & Extract the Kafka binaries from `https://kafka.apache.org/downloads`
- Try Kafka commands using `bin/kafka-topics.sh (for example)`
- Edit PATH to include Kafka (in ~/.bashrc for example) `PATH="$PATH:/your/path/to/your/kafka/bin"`
- Edit Zookeeper & Kafka configs using a text editor
- `zookeeper.properties: dataDir=/your/path/to/data/zookeeper`
- `server.properties: log.dirs=/your/path/to/data/kafka`

Start Zookeeper in one terminal window: `zookeeper-server-start.sh config/zookeeper.properties`

Start Kafka in another terminal window: `kafka-server-start.sh config/server.properties`

## Windows local installation

- Download and Setup Java 8 JDK
- Download the Kafka binaries from `https://kafka.apache.org/downloads`
- Extract Kafka at the root of `C:\`
- Setup Kafka bins in the Environment variables section by editing Path

- Try Kafka commands using `kafka-topics.bat` (for example)
- Edit Zookeeper & Kafka configs using NotePad++ `https://notepad-plus-plus.org/download/`
- zookeeper.properties: `dataDir=C:/kafka_2.12-2.0.0/data/zookeeper` (yes the slashes are inversed)
- server.properties: `log.dirs=C:/kafka_2.12-2.0.0/data/kafka` (yes the slashes are inversed)

Start Zookeeper in one command line: `zookeeper-server-start.bat config\zookeeper.properties`

Start Kafka in another command line: `kafka-server-start.bat config\server.properties`

## References

*This demo has been inspired by series of Kafka courses from Stephane Maarek at Udemy.com*