Tutorial – Deploying Kakfa service in localhost

Instituto Superior Técnico
Universidade de Lisboa
Integração Empresarial / Enterprise Integration

The goal of this document is to show how to operate the basics of a local Kafka service: installation, starting, accessing, and testing.

Local installation is considered, where it requires the previous installation of docker in the computer.

The following contents is presented in this document.

Contents

A. Create a linux Docker container (for windows)

The goal of this section is to create a linux environment to later install the Kafka server.

A.1. Install docker from https://hub.docker.com/editions/community/docker-ce-desktop-windows
Hint 1: if Microsoft HyperV service is missing, execute the following command using a PowerShell as administrator:

```
dism.exe /Online /Enable-Feature:Microsoft-Hyper-V
```

A.2. Open a new PowerShell command line and search for the ubuntu images that are available using the following command:

```
docker search ubuntu
```

A.3. Download the ubuntu image with the following command:

```
docker run -it ubuntu
```

A.4. The container could be found, in your machine, with the following command:

```
docker container ls -a
```

A similar information will be presented to you:

```
8122c7f7cef9   ubuntu   "bash"    6 weeks ago         Exited (0)  6 weeks ago    flamboyant_wright
```

A.5. You can now start a container with the command:

```
docker container start -i 4ac7185afce3
```

where the container ID corresponds to the information identified in A.4.

A.6. Update your container with the commands:

```
apt update
apt-get install sudo
sudo apt-get install openjdk-8-jdk
apt-get install telnet
```

A.7. To finish the session, use the following command:

```
exit
```

Exercise 1: After starting a container, if you want to start a new session for that container, open a new PowerShell windows a use the following command:

```
docker exec -it 4ac7185afce3 bash
```

where the container ID corresponds to the information identified in A.4.

B. Install Kafka in localhost (for windows)

The goal of this section is to describe the required steps to install the Zookeeper and Kafka servers in your localhost. This section requires the previous execution of section A.

B.1. Download the Kafka binary version from https://kafka.apache.org/downloads. Recommended version is kafka_2.12-2.1.0.tgz (in December 2018).
B.2. Download the ZooKeeper binary version from http://mirror.cc.columbia.edu/pub/software/apache/zookeeper/stable/. Recommended version is zookeeper-3.4.12.tar.gz (in December 2018).
B.3. Copy the files B.1. and B.2. to your docker container with the following command:

```
docker cp .\zookeeper-3.4.12.tar.gz 29f9117a8312:/root
docker cp .\kafka_2.12-2.1.0.tgz 29f9117a8312:/root
```

B.4. Access your docker container accordingly with step A.5.
B.5. To change for your home directory, type the command:

```
cd
```

B.6. Then, type the command:

```
ls -ltr
```

to check if you have both B.1. and B.2. files available.
B.7. To extract the zookeeper files, type the command:

```
tar -zxf zookeeper-3.4.12.tar.gz
```

B.8. Move the new zookeeper directory to system typing the command:

```
mv zookeeper-3.4.12 /usr/local/zookeeper
```

B.9. Create a new directory using the command:

```
mkdir -p /var/lib/zookeeper
```

B.10. Create the baseline zookeeper server configuration with the command:

```
cat > /usr/local/zookeeper/conf/zoo.cfg << EOF
```

and then write the following content to the file directly in the command line:

```
tickTime=2000
dataDir=/var/lib/zookeeper
clientPort=2181
EOF
```

B.11. Start ZooKeeper server with the command:

```
/usr/local/zookeeper/bin/zkServer.sh start
```

B.12. To extract the kafka server, type the command:

```
tar -zxf kafka_2.12-2.1.0.tgz
```

B.13. Move the new kafka directory to system typing the command:

```
mv kafka_2.12-2.1.0 /usr/local/kafka
```

B.14. Create a new directory using the command:

```
mkdir /tmp/kafka-logs
```

B.15. Start Kafka with the command:

```
/usr/local/kafka/bin/kafka-server-start.sh -daemon /usr/local/kafka/config/server.properties
```

Hint 1: you can always check if Zookeeper and Kafka servers are started correctly with the command:

```
ps -ef |grep java
```

C. Testing Kafka locally (for windows)

The goal of this section if to test if the previous installation in section B. was executed successfully. The production and consumption of Kafka messages are tested. It is considered that in this environment the servers and the clients are installed in the same machine.

C.1. Access your docker container accordingly with step A.5.

C.2. Test ZooKeeper installation typing the command:

```
telnet localhost 2181
```

then write the command:

```
srvr
```

and it is expected to receive the following output:

```
root@29f9117a8312:~# telnet localhost 2181
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
srvr
Zookeeper version: 3.4.12-e5259e437540f349646870ea94dc2658c4e44b3b, built on 03/27/2018 03:55 GMT
Latency min/avg/max: 0/1/50
Received: 217
Sent: 217
Connections: 2
Outstanding: 0
Zxid: 0x151
Mode: standalone
Node count: 138
Connection closed by foreign host.
```

C.3. Test Kafka installation, with the creation of a new topic named “test”, typing the command:

```
/usr/local/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions
1 --topic test
```

C.4. Type the command to check if the topic “test” is created successfully:

```
/usr/local/kafka/bin/kafka-topics.sh --zookeeper localhost:2181 --describe --topic test
```

C.5. Type the command to produce messages to the topic “test”:

```
/usr/local/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test
```

And write some text messages, one per line. Then, to finish press Ctrl+D.

C.6. Type the command to consume messages from the topic “test”:

```
/usr/local/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-
beginning
```

All the sent messages from topic “test” will be presented. Then, to finish press Ctrl+C.

Exercise 2: Try creating different docker container sessions for producer and consumer and check the run-time production and consumption of messages. Use the command described in Exercise 1.