KAFKA BIGDATA

How to easily install kafka without zookeeper

05 Jun 2021 · 9 mins read

Photo by AbsolutVision On Unsplash

You can also check out my Youtube Channel to get access to various tutorials created by me.

In this article we will see how kafka can be setup without using zookeeper. We will setup a 3 node kafka cluster and create a test topic. We will use a kafka producer to produce data into the test topic and also use a kafka consumer to consume data from the kafka topic.

Why is zookeeper used in kafka

Zookeeper is used to store kafka cluster metadata information. Zookeeper stores information like topic configuration, topic partition locations and so on.

How will kafka work without zookeeper

In order to run kafka without zookeeper, it can be run using **Kafka Raft metadata mode (KRaft)**. In **KRaft** the kafka metadata information will be stored as a partition within kafka itself. There will be a KRaft Quorum of controller nodes which will be used to store the metadata. The metadata will be stored in an internal kafka topic **@metadata** .

This is available in an experimental mode in kafka 2.8.0

Right now using KRaft is experimental and should not be used in production

Setting up a kafka cluster without zookeeper

Download kafka

Download kafka 2.8.0 from https://kafka.apache.org/downloads

```
wget https://apachemirror.wuchna.com/kafka/2.8.0/kafka_2.12-2.8.0.tgz
```

Extract kafka

```
tar xzf  kafka_2.12-2.8.0.tgz
```

Open the kafka folder. **All kafka commands should be run in the kafka folder**

```
cd kafka_2.12-2.8.0
```

Kafka cluster configuration

If you go to `config/kraft` folder inside the kafka home directory, you will see a file called `server.properties` . This is a sample file which is provided by kafka, to show how kafka can be started without zookeeper

```
cd config/kraft
cp server.properties server1.properties
cp server.properties server2.properties
cp server.properties server3.properties
```

In server1.properties, modify the following properties. Please keep the other properties as is.

```
node.id=1

process.roles=broker,controller

inter.broker.listener.name=PLAINTEXT

controller.listener.names=CONTROLLER

listeners=PLAINTEXT://:9092,CONTROLLER://:19092

log.dirs=/tmp/server1/kraft-combined-logs

listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAI

controller.quorum.voters=1@localhost:19092,2@localhost:19093,3@localhost:19094
```

Let me explain what these properties do:

node.id: This will act as the node Id in the cluster. This will help us identify which broker this is. It will also help us identify which kraft controller node this is.

process.roles: A node can act as a **broker** or **controller** or **both**. Here we are indicating that this node can be both a kafka broker and a kraft controller node.

inter.broker.listener.name: Here the broker listener name is set to **PLAINTEXT**

listeners: Here we indicate that the broker will use port 9092 and the kraft controller will use port 19092

log.dirs: This is the log directory where kafka will store the data

listener.security.protocol.map: Here the connection security details are added

controller.quorum.voters: This is used to indicate all the kraft controllers which are available. Here we are indicating that we will have 3 kraft controller nodes running on ports 19092, 19093 and 19094

For **server2.properties** modify the following properties. Please keep the other properties as is.

```
node.id=2

process.roles=broker,controller

controller.quorum.voters=1@localhost:19092,2@localhost:19093,3@localhost:19094

listeners=PLAINTEXT://:9093,CONTROLLER://:19093

inter.broker.listener.name=PLAINTEXT

controller.listener.names=CONTROLLER

log.dirs=/tmp/server2/kraft-combined-logs

listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAI
```

In Server 2, the broker port is 9093 and controller port is 19093. Also the log.dirs is different

For **server3.properties** modify the following properties. Please keep the other properties as is.

```
node.id=3
```



```
controller.quorum.voters=1@localhost:19092,2@localhost:19093,3@localhost:19094
```

```
listeners=PLAINTEXT://:9094,CONTROLLER://:19094

inter.broker.listener.name=PLAINTEXT

controller.listener.names=CONTROLLER

log.dirs=/tmp/server3/kraft-combined-logs

listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAI
```

In Server 3 the broker port is 9094 and the controller port is 19094. Also the log.dirs is different

The complete server property files are available in this Git repo

Kafka cluster id creation and log directory setup

First we need to create kafka cluster id before starting the servers. This can be done using the following command:

```
./bin/kafka-storage.sh random-uuid
```

Note down the uuid that comes after running the above statement. In my case I got the following uuid

```
./bin/kafka-storage.sh random-uuid
9dJzdGvfTPaCY4e8klXaDQ
```

Next we need to format all the storage directories. This is basically the directory that we put in **log.dirs** property.

This can be done with the following command

```
./bin/kafka-storage.sh format -t <uuid> -c <server_config_location>
```

Replace `<uuid>` with the uuid that you got in the previous step. Replace `<server_config_location>` with the server property files

In my case, I will be running the following commands

For Server 1:

```
./bin/kafka-storage.sh format -t 9dJzdGvfTPaCY4e8klXaDQ -c ./config/kraft/server1.properties
```

For Server 2:

```
./bin/kafka-storage.sh format -t 9dJzdGvfTPaCY4e8klXaDQ -c ./config/kraft/server2.properties
```

For Server 3:

```
./bin/kafka-storage.sh format -t 9dJzdGvfTPaCY4e8klXaDQ -c ./config/kraft/server3.properties
```

This is the result which you get when you run the 3 commands

```
$ ./bin/kafka-storage.sh format -t 9dJzdGvfTPaCY4e8klXaDQ -c ./config/kraft/server1.properties
Formatting /tmp/server1/kraft-combined-logs
$ ./bin/kafka-storage.sh format -t 9dJzdGvfTPaCY4e8klXaDQ -c ./config/kraft/server2.properties
Formatting /tmp/server2/kraft-combined-logs
$ ./bin/kafka-storage.sh format -t 9dJzdGvfTPaCY4e8klXaDQ -c ./config/kraft/server3.properties
Formatting /tmp/server3/kraft-combined-logs
```

Starting the kafka servers

The kafka servers can be started in daemon mode using the following commands:

```
export KAFKA_HEAP_OPTS="-Xmx200M -Xms100M"
```

Here we are giving a very small max heap of 200M since we are running all the servers in a single local machine. If you have bigger servers you can give heap size of 1GB or Above.

Start Server 1:

```
./bin/kafka-server-start.sh -daemon ./config/kraft/server1.properties
```

Start Server 2:

```
./bin/kafka-server-start.sh -daemon ./config/kraft/server2.properties
```

Start Server 3:

```
./bin/kafka-server-start.sh -daemon ./config/kraft/server3.properties
```

Create a kafka topic

Let us create a topic **kraft-test** in this cluster

The topic can be created using the following command:

```
./bin/kafka-topics.sh --create --topic kraft-test --partitions 3 --replication-factor 3 --bootstrap-serv
```

We are creating a topic with 3 partitions and 3 replicas since we have 3 nodes.

Running the above command gives the below result

```
$ bin/kafka-topics.sh --bootstrap-server localhost:9093 --list
kraft-test
```

We can describe the topics present in the cluster using the following command:

```
bin/kafka-topics.sh --bootstrap-server localhost:9093 --describe --topic kraft-test
```

Running the above command gives the below result

```
$ bin/kafka-topics.sh --bootstrap-server localhost:9093 --describe --topic kraft-test
Topic: kraft-test       TopicId: vZKswHHlQk2mEOw0yzAGAA PartitionCount: 3       ReplicationFactor: 3    C
        Topic: kraft-test       Partition: 0    Leader: 3       Replicas: 3,2,1 Isr: 3,2,1
        Topic: kraft-test       Partition: 1    Leader: 2       Replicas: 2,3,1 Isr: 2,3,1
        Topic: kraft-test       Partition: 2    Leader: 2       Replicas: 2,3,1 Isr: 2,3,1
```

Exploring the kafka metadata using metadata shell

Similar to zookeeper cli, there is a **metadata shell** provided by kafka so that we can read the data in the **@metadata** internal topic.

Open the shell using the following command

```
./bin/kafka-metadata-shell.sh  --snapshot /tmp/server1/kraft-combined-logs/\@metadata-0/0000000000000000
```

The structure here is very similar to what we see in zookeeper

```
>> ls brokers/
1  2  3
```

In order to list the topics you can type `ls topics/`

```
>> ls topics/
kraft-test
```

In order to see the topic metadata you can type `cat topics/kraft-test/0/data`

```
>> cat topics/kraft-test/0/data
{
  "partitionId" : 0,
  "topicId" : "vZKswHHlQk2mEOw0yzAGAA",
  "replicas" : [ 3, 2, 1 ],
  "isr" : [ 3, 2, 1 ],
  "removingReplicas" : null,
  "addingReplicas" : null,
  "leader" : 3,
  "leaderEpoch" : 0,
  "partitionEpoch" : 0
}
```

In order to find the controller leader node you can type `cat metadataQuorum/leader`

```
>> cat metadataQuorum/leader
MetaLogLeader(nodeId=2, epoch=4)
```

Type `exit` to get out of the metadata shell

Producing and consuming data from kafka

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

In a different terminal, use the following command to start a kafka consumer

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic kraft-test
```

In the producer terminal send a bunch of messages as shown below

```
$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic kraft-test
>message 1
>message 2
>message 3
>hello
>bye
```

In the consumer terminal you would see the messages coming as shown below

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic kraft-test
message 1
message 2
message 3
hello
bye
```

Congrats 😊

You have now learnt how to setup kafka without zookeeper. **This is still an experimental feature and should not be used in production.**

References

https://kafka.apache.org/

https://cwiki.apache.org/confluence/display/KAFKA/KIP-500

Feel free to connect with me in LinkedIn or follow me in Twitter

You can also check out my Youtube Channel to get access to various tutorials created by me.

‹ What is the use of Synchron...
How to get started with kaf... ›