

# Java-Success.com

Prepare to fast-track, choose & go places with 800+ Java & Big Data Q&As with lots of code & diagrams.

[Home](#) [Why? ▾](#) [300+ Java FAQs ▾](#) [300+ Big Data FAQs ▾](#) [Courses ▾](#)[👤 Membership ▾](#) [Your Career ▾](#)

[Home](#) › [bigdata-success.com](#) › [Tutorials - Big Data](#) › [TUT - Kafka](#) › 02: Apache Kafka  
multi-broker cluster tutorial

## 02: Apache Kafka multi-broker cluster tutorial

 Posted on [February 16, 2019](#)

This extends [Getting started with Apache Kafka on Mac tutorial](#). This assumes that the zookeeper & kafka servers are started as per the previous tutorial.

### List topics

Create a topic first:

```
1 /usr/local/kafka_2.11-2.1.0]$ ./bin/kafka-topics.sh
```

List topics:

### 300+ Java Interview FAQs

300+ Java FAQs



16+ Java Key  
Areas Q&As



150+ Java  
Architect FAQs



80+ Java Code  
Quality Q&As



150+ Java Coding  
Q&As



### 300+ Big Data Interview FAQs

300+ Big Data  
FAQs



Tutorials - Big  
Data



TUT -  Starting Big  
Data

TUT - Starting Spark &  
Scala

```
1 /usr/local/kafka_2.11-2.1.0]$ ./bin/kafka-topics.sh
2
3 test
```

## Delete a topic

```
1 /usr/local/kafka_2.11-2.1.0]$ ./bin/kafka-topics.sh
```

This delete command will have no effect if in the Kafka server.properties file, if **delete.topic.enable** is not set to be true.

## Get info about a topic

```
1 /usr/local/kafka_2.11-2.1.0]$ ./bin/kafka-topics.sh
2
3 Topic:test      PartitionCount:1      ReplicationFactor
4   Topic: test   Partition: 0      Leader: 0      Rep
5
```

## Creating topics with partitions & replications with multiple brokers

```
1 /usr/local/kafka_2.11-2.1.0]$ ./bin/kafka-topics.sh
2
3 Error while executing topic command : Replication
4 [2019-02-16 12:36:36,802] ERROR org.apache.kafka.co
5 (kafka.admin.TopicCommand$)
```

The above error is thrown because of a single broker. Kafka is a distributed system, hence you need to harness its strength by having more brokers across multiple nodes. Replication gives you fault tolerance.

Stop the Kafka server and make the following changes to have 3 brokers.

TUT - Starting with Python

TUT - Kafka

TUT - Pig

TUT - Apache Storm

TUT - Spark Scala on Zeppelin

TUT - Cloudera

TUT - Cloudera on Docker

TUT - File Formats

TUT - Spark on Docker

TUT - Flume

TUT - Hadoop (HDFS)

TUT - HBase (NoSQL)

TUT - Hive (SQL)

TUT - Hadoop & Spark

TUT - MapReduce

TUT - Spark and Scala

TUT - Spark & Java

TUT - PySpark on Databricks

TUT - Zookeeper

## 800+ Java Interview Q&As

300+ Core Java Q&As



300+ Enterprise Java Q&As



150+ Java Frameworks Q&As



120+ Companion Tech Q&As



Tutorials - Enterprise Java



**Step 1:** Copy the `./config/server.properties` file into 2 additional `.properties` files.

```
1 /usr/local/kafka_2.11-2.1.0]$ cp ./config/server.properties ./config/server1.properties
```

```
1 /usr/local/kafka_2.11-2.1.0]$ cp ./config/server.properties ./config/server2.properties
```

**Step 2:** Modify `./config/server1.properties` & `./config/server2.properties` by changing the 3 property values as shown below.

```
1 /usr/local/kafka_2.11-2.1.0]$ vi ./config/server1.properties
```

```
1 broker.id=1
2 .....
3 listeners=PLAINTEXT://:9093
4 .....
5 log.dirs=/tmp/kafka-logs-1
```

```
1 /usr/local/kafka_2.11-2.1.0]$ vi ./config/server2.properties
```

```
1 broker.id=2
2 .....
3 listeners=PLAINTEXT://:9094
4 .....
5 log.dirs=/tmp/kafka-logs-2
```

**Step 3:** Start all 3 brokers (i.e. brokers 0, 1 & 2) on 3 different terminal windows

```
1 /usr/local/kafka_2.11-2.1.0]$ ./bin/kafka-server-start.sh ./config/server0.properties
```

```
2
```

```
1 /usr/local/kafka_2.11-2.1.0]$ ./bin/kafka-server-start.sh ./config/server1.properties
```

```
2
```

```
1 /usr/local/kafka_2.11-2.1.0]$ ./bin/kafka-server-s
2
```

**Step 4:** Now create a topic with 5 partitions and a replication factor of 3.

```
1 /usr/local/kafka_2.11-2.1.0]$ ./bin/kafka-topics.sl
Created topic "my-demo-topic".
2
```

**Step 5:** In a cluster, how do you know which broker is doing what? This is where **-describe** comes in handy.

```
1 /usr/local/kafka_2.11-2.1.0]$ ./bin/kafka-topics.sl
```

It shows the leader & the follower (i.e. replica) broker numbers. We have created 5 partitions for parallelism. "ISR" means in sync replicas.

```
1
2 Topic:my-demo-topic PartitionCount:5 Repl
3 Topic: my-demo-topic Partition: 0 Leader
4 Topic: my-demo-topic Partition: 1 Leader
5 Topic: my-demo-topic Partition: 2 Leader
6 Topic: my-demo-topic Partition: 3 Leader
7 Topic: my-demo-topic Partition: 4 Leader
8
```

**Leader** is the node responsible for all reads and writes for a given partition. Each broker will be the leader for a randomly selected portion of the partitions. Every partition in a Kafka topic has a write-ahead log where the messages are stored and every message has a unique offset that identifies its position in the partition's log.

**Replicas** are the list of nodes that replicate the log for this partition regardless of whether they are the leader or even if they are currently alive. Every topic partition in Kafka is replicated  $n$  (e.g. 3) times, where  $n$  is the replication factor of the topic. This allows Kafka to automatically failover to these replicas when a server in the cluster fails so that messages remain available in the presence of failures.

Replication in Kafka happens at the partition granularity where the partition's write-ahead log is replicated in order to  $n$  servers.

**"ISR"** is the subset of the replicas list that is currently alive and caught-up to the leader. When a producer sends a message to the broker, it is written by the leader and replicated to all the partition's replicas. A message is committed only after it has been successfully copied to all the in-sync replicas. The leader for every partition tracks this in-sync replica (aka ISR) list by computing the lag of every replica from itself.

◀ 01: Getting started with Apache Kafka on Mac tutorial

03: Apache Kafka Connector Tutorial ▶

## Disclaimer

The contents in this Java-Success are copyrighted and from EmpoweringTech pty Ltd. The EmpoweringTech pty Ltd has the right to correct or enhance the current content without any prior notice. These are general advice only, and one needs to take his/her own circumstances into consideration. The EmpoweringTech pty Ltd will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. Links to external sites do not imply endorsement of the linked-to sites. [Privacy Policy](#).

