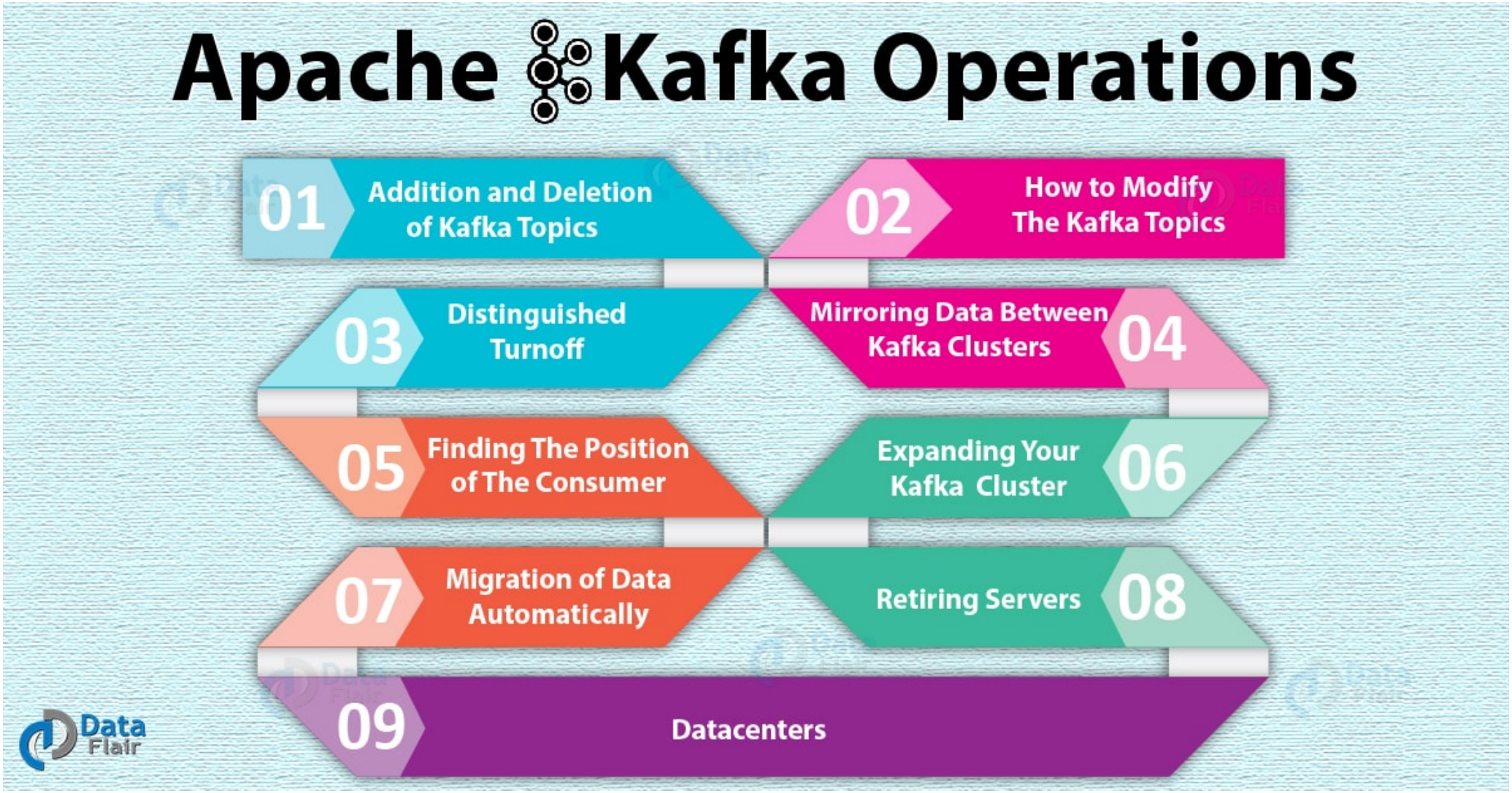
Apache Kafka Operations With Commands

BY DATAFLAIR TEAM · UPDATED · FEBRUARY 5, 2020

Keeping you updated with latest technology trends, Join DataFlair on Telegram

1. Objective

There are several **Apache Kafka** Operations we can perform on our **Kafka cluster**. So, in this article, we will discuss all Apache Kafka Operations in detail. It also includes such commands that will help to implement these Kafka Operations.
So, let's discuss all the Kafka Operations in detail.



Apache Kafka Operations With Commands

2. What are Apache Kafka Operations?

Below, we are discussing operations in Kafka, let's see them one by one:

a. Addition and Deletion of Kafka Topics

Both automatically and manually we can add and delete Kafka Topics. The **command** for the addition will be:

```
> bin/Kafka-Topics.sh –zookeeper zk_host:port/chroot –create –Topic my_Topic_name
–partitions 20 –replication-factor 3 –config x=y
```

In addition, **Kafka Brokers** are the messages that are written and replicated by the servers. However, by the replication factors, the whole process of replication is done. Although, make sure that the standard replication factor should be either 2 or 3 for bouncing without any obstacles.
The Topic is split into a number of logs and the partition count controls it. However, partition count consists of following features, like they should fit singly on one server. Hence, in this way, a total of 50 partitions will result in 50 servers. Also, partitions obtain maximum equalizations among the consumers. Basically, the addition of Topics overrides the default settings.

b. How to modify the Kafka Topics

We can best modify the Topics by adding as well as deleting contexts. To add partitions the **command** is-

```
> bin/Kafka-Topics.sh –zookeeper zk_host:port/chroot –alter –Topic my_Topic_name
–partitions 40
```

To divide the data into smaller sections, we use Partitions. We should follow Adding configs as:
Read Apache Kafka Workflow | Kafka Pub-Sub Messaging

```
> bin/Kafka-Topics.sh –zookeeper zk_host:port/chroot –alter –Topic my_Topic_name –config x=y
```

In order to delete configs:

```
> bin/Kafka-Topics.sh -zookeeper zk_host:port/chroot -alter -Topic my_Topic_name -deleteConfig x
```

Also, to delete off the entire Topic,

```
> bin/Kafka-Topics.sh -zookeeper zk_host:port/chroot -delete -Topic my_Topic_name
```

c. Distinguished Turnoff

If any Kafka Broker shutdown or fail and elect new partition leaders, the **Kafka cluster** will automatically detect. However, if a server fails or it is brought down intentionally for maintenance, this will occur. Moreover, Kafka supports a more graceful mechanism for stopping a server than just destroying it for configuration changes. Also, it has two optimizations, when we cut down a server in an elegant way:

1. The cluster will automatically detect it, irresponsible of, whether the Kafka broker is failed or broken, if any of the servers is failed, also it can beautifully shut down the servers.
2. If done, then the need of log recovery is removed and so the process becomes faster. Also, it will send all the partitions of that server to its replicas before shutting down.

d. Mirroring Data between Kafka Clusters

The mirroring process reads the materials from the sources and writes to the destination. Basically, this mirror replicates the sources. Also, then writes it down on the destination.

Learn Kafka Performance Tuning – Ways for Kafka Optimization

In order, to keep the process stay robust, Mirroring increases the throughput. Also, we can use a lot of it. It transfers data from clusters to clusters. However, make sure the source cluster and destination clusters are almost different from each other. Basically, the mirror maker is the same for both. Because destination cluster data after mirroring is same as the data the source

cluster contains before mirroring. Although, with different addresses, both names will be the same in both the clusters due to its data replication and different offsets and partitions.

Hence, it provides less fault-tolerance because mirror cluster will copy the data in a consumer with a different address. However, we should use normal in-cluster replication in order to obtain more fault-tolerance.

For mirroring a single Topic named as your-Topic from two inputs, the **command** is:

```
> bin/Kafka-run-class.sh Kafka.tools.MirrorMaker
-consumer.config consumer-1.properties -consumer.config consumer-
2.properties
-producer.config producer.properties -whitelist your-Topic
```

e. Finding the position of the Consumer

It is very important to find out the positions of the consumers. Hence, for finding out the consumer's location, the **command** is:

```
> bin/Kafka-run-class.sh Kafka.tools.ConsumerOffsetChecker -zkconnect localhost:2181 -group test
```

f. Expanding Your Kafka Cluster

By adding servers up with a new id we can anytime add new servers to the clusters. Further, we start them up with new servers. But they will not work at all and sit ideally until and unless partitions are assigned to these new servers. Hence, the existing ones need to change their positions a little bit, to add new ones.

Explore Apache Kafka Producer For Beginners

Therefore, automatically the data moves behind the covers. Now, we have to add the new server. But its position must be behind the partition that seeking for migration. Hence, the newly introduced server walks behind the migrating partition. Also, all the data from the migrating partition will be copied to the server. Ultimately, the partition is kicked out.

Basically, in 3 ways we run this reassigning of partitions:

–generate:

It is important to assign all the Topics to the new servers. Hence, to move all the partitions to the new servers prior to migration this tool creates the reassignment tool.

–execute:

For execution, the tool kicks off the reassignment of the Topics as planned by the user, in the execution plan.

–verify:

As soon as reassignment completes, this tool verifies. This tool also tells about the status. It can be done, ongoing, finished or even if it is failed.

g. Migration of Data Automatically

Basically, the reassignment tools assign the Topics from the present servers to the new servers. However, in order to move the entire Topic rather than moving a part of it, it expands the partitions. Hence, a list of the servers to whom the Topics are to be moved is included in a single list and another list of servers from whom the Topics are to be transferred.

Further, the Kafka Topics from the old to the new servers are transferred by the reassignment tool transfers and all the replicas are also transferred to new servers keeping the mirroring in even mood.

Let’s discuss Apache Kafka Streams | Stream Processing Topology

At first, to save the Topics, create the JSON file. Creation of this file is:

```
> cat Topics-to-move.json
{“Topics”: [{“Topic”: “top1”},
{“Topic”: “top2”}],
“version”:1
}
```

Where top1 and top2 are Topics.

The assignment is introduced, after generating the JSON file:

```
> bin/Kafka-reassign-partitions.sh –zookeeper localhost:2181 –Topics-to-move-json-file Topics-to-move.json –broker-list "5,6" –generate
```

For assignment:

```
{"version":1,
"partitions":[{"Topic":"top1","partition":2,"replicas":[1,2]},
{"Topic":"top1","partition":0,"replicas":[3,4]},
{"Topic":"top2","partition":2,"replicas":[1,2]},
{"Topic":"top2","partition":0,"replicas":[3,4]},
{"Topic":"top1","partition":1,"replicas":[2,3]},
{"Topic":"top2","partition":1,"replicas":[2,3]}]
}
```

Further, for reassignment :

```
{"Version":1,
"Partitions":[{"Topic":"top1","partition":2,"replicas":  [1,2] } ,
{"Topic":"top1","partition":0,"replicas":[1,2]},
{"Topic":"top2","partition":2,"replicas":[1,2]},
{"Topic":"top2","partition":0,"replicas":[1,2]},
{"Topic":"top1","partition":1,"replicas":[1,2]},
{"Topic":"top2","partition":1,"replicas":[1,2]}]
}
```

Therefore, we will move all partitions from Topics top1, top2 to the servers 1 and 2. Although, make sure to save the new assignment. Then it will be easy to move all Topics top1top2 to the new servers 5and 6. In this way, we know what actually we need to move from where to which server.

So, in the JSON file, the new positions of the Topics are saved. Also, the present positions of the Topics are saved for keeping a backup, in case, we really want to bring the Topics back to their old server.

The commands are:

```
> bin/Kafka-reassign-partitions.sh –zookeeper localhost:2181 –reassignment-json-file expand-cluster-reassignment.json –execute
```

Now, replica assigning for the present one:

```
{"Version":1,
"Partitions": [{"Topic":"top1","partition":2,"replicas":[1,2]},
{"Topic":"top1","partition":0,"replicas":[3,4]},
{"Topic":"top2","partition":2,"replicas":[1,2]},
{"Topic":"top2","partition":0,"replicas":[3,4]},
{"Topic":"top1","partition":1,"replicas":[2,3]},
{"Topic":"top2","partition":1,"replicas":[2,3]}]
}
```

For the purpose of reassigning partitions:

```
{"Version":1,
"Partitions":[{"Topic":"top1","partition":2,"replicas":[1,2]},
{"Topic":"top1","partition":0,"replicas":[1,2]},
{"Topic":"top2","partition":2,"replicas":[1,2]},
{"Topic":"top2","partition":0,"replicas":[1,2]},
{"Topic":"top1","partition":1,"replicas":[1,2]},
{"Topic":"top2","partition":1,"replicas":[1,2]}]
}
```

Further, to verify that the process of reassigning is perfect or not, use this **command**:

```
> bin/Kafka-reassign-partitions.sh –zookeeper localhost:2181 –reassignment-json-file expand-cluster-reassignment.json –verify
```

Here is the **Output** of Verification:

```
Reassignment of partition [top1,0] completed successfully
```

Reassignment of partition [top1,1] completed successfully Reassignment of partition [top1,2] completed successfully Reassignment of partition [top2,0] is in progress Reassignment of partition [top2,1] completed successfully Reassignment of partition [top2,2] is in progress.
In order to assign, we can select a certain number of partitions to new servers rather than moving the entire set of partitions. Also, we have to directly go to the execute step.
Let's learn Apache Kafka + Spark Streaming Integration

h. Retiring Servers

There are times when few servers become de-active. Then, we need to remove the Topics from the retiring server to the new servers. Although to help this process, the generation of new tools is still in process.

i. Datacenters

For the purpose of maintenance of operations, we use Data Centers. Basically, they manage the data pipelines. We are towards operating with some local clusters hence, the data centers communicate directly to the nearest cluster, in Kafka.
Basically, the data centers work independently. Hence, we can say those data centers can work independently are best in operation, even if in-between links are broken. Also, keep in mind that the failure of the inner links leads to the failure of the mirroring process. Moreover, only when the links are redeveloped again it comes back to its actual position.

Moreover, in the Topics, some data centers ask for viewing the complete set of data. In that case, start mirroring the clusters of data to show to the new data centers. Afterwards, these mirrored data are accumulated to data clusters, that we can read when we need the full reading of the complete data by the applications.

Read Apache Kafka Security | Need and Components of Kafka

To obtain the clusters, this deployment pattern is very proper and can warranty latency. The Kafka tools offer better throughput than other simpler messaging tools, even with high latency. It is because Kafka batches the data in both the sides i.e. in the sources as well as in the destinations. Here, source refers the producer whereas the destination refers the consumer.

Also, make sure prevent using high latency links, that will introduce high latency for the writing acts of the Kafka technology. Because of that if there is the problem in the network, Kafka will not be available in all the locations.

So, this was all about Kafka Operations. Hope you like our explanation.

3. Conclusion: Kafka Operations

Hence, we have seen all the Apache Kafka Operations in detail. We discussed different Kafka Operations, such as addition and deletion of Kafka Topics, Modifying the Kafka Topics, Distinguishing the Turnoff and many more. Still, if you feel any query regarding Kafka Operations, feel free to ask in the comment tab.

See also –

Apache Kafka Monitoring – Methods & Tools

For reference