# KAFKA DATA REBALANCING -- PARTITION RE-ASSIGNMENT

## Kafka-reassign-partitions

This tool provides substantial control over partitions in a Kafka cluster. It is mainly used to balance storage loads across brokers through the following reassignment actions.

- Change the ordering of the partition assignment list. Used to control leader imbalances between brokers.
- Reassign partitions from one broker to another. Used to expand existing clusters.
- Reassign partitions between log directories on the same broker. Used to resolve storage load imbalance among available disks in the broker.
- Reassign partitions between log directories across multiple brokers. Used to resolve storage load imbalance across multiple brokers.
- Here we are going to see Partition re-assignment in detail as removing/broker is time consuming so we can't able to demonstrate the first technique in short period of time.

## Cluster Info:

We set up a 4 node clusters using AWS instance. We have following 3 brokers and one ZooNavigator and the details are below. Mentioned Ips are Public which will change every time we stop/start the instances.

- Kafka Broker1(kafka1, zookeeper1) - 172.31.17.2
- Kafka Broker2(kafka2, zookeeper2) - 172.31.1.2
- Kafka Broker2(kafka3, zookeeper3) - 172.31.18.2
- ZooNavigator - 172.31.12.118

Kafka uses ZooKeeper to store persistent cluster metadata and is a critical component of the Confluent Platform deployment. For example, if you lost the Kafka data in ZooKeeper, the mapping of replicas to Brokers and topic configurations would be lost as well, making your Kafka cluster no longer functional and potentially resulting in total data loss.

In a production environment, the Zookeeper servers will be deployed on multiple nodes. This is called an ensemble. One will be the leader and others will be followers. An ensemble is a set of **2n + 1** ZooKeeper servers where n is any number greater than 0. The odd number of servers allows ZooKeeper to perform majority elections for leadership. **At any given time, there can be up to n failed servers in an ensemble and the ZooKeeper cluster will keep quorum. If at any time, quorum is lost, the ZooKeeper cluster will go down.**

## Partition Re-assignment:

1. **Before Partition re-assignment:**

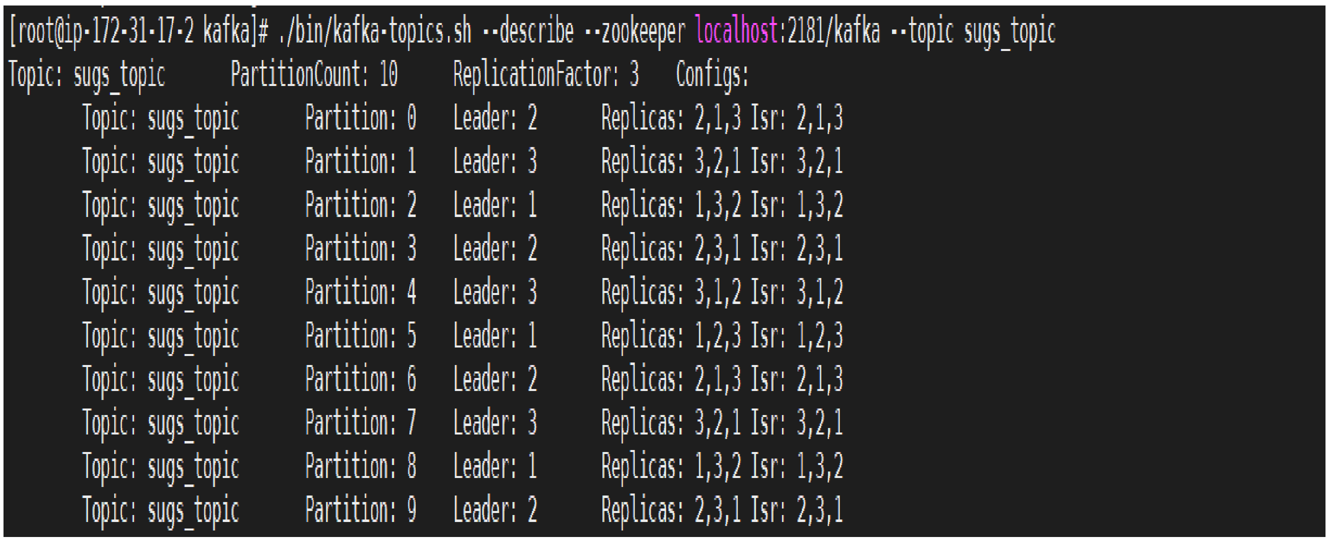
We have created a Topic named 'sugs_topic' with replication factor=3 and Partition=10. Below is the description and distribution of partitions and their ISR (in sync replicas) on the brokers. 1,2 and 3 are the unique broker Ids. Port 2181 is used by zookeeper clients to connect to the zookeeper servers here.

**Creating Topic:**

```
$bin/kafka-topics.sh --zookeeper
zookeeper1:2181,zookeeper2:2181,zookeeper3:2181/kafka --create --
topic sugs_topic --replication-factor 3 --partitions 10
```

**Describing Topic:**

```
$bin/kafka-topics.sh --describe --zookeeper localhost:2181/kafka --
topic sugs_topic
```



```
[root@ip-172-31-17-2 kafka]# ./bin/kafka-topics.sh --describe --zookeeper localhost:2181/kafka --topic sugs_topic
Topic: sugs_topic       PartitionCount: 10      ReplicationFactor: 3    Configs:
        Topic: sugs_topic       Partition: 0    Leader: 2       Replicas: 2,1,3 Isr: 2,1,3
        Topic: sugs_topic       Partition: 1    Leader: 3       Replicas: 3,2,1 Isr: 3,2,1
        Topic: sugs_topic       Partition: 2    Leader: 1       Replicas: 1,3,2 Isr: 1,3,2
        Topic: sugs_topic       Partition: 3    Leader: 2       Replicas: 2,3,1 Isr: 2,3,1
        Topic: sugs_topic       Partition: 4    Leader: 3       Replicas: 3,1,2 Isr: 3,1,2
        Topic: sugs_topic       Partition: 5    Leader: 1       Replicas: 1,2,3 Isr: 1,2,3
        Topic: sugs_topic       Partition: 6    Leader: 2       Replicas: 2,1,3 Isr: 2,1,3
        Topic: sugs_topic       Partition: 7    Leader: 3       Replicas: 3,2,1 Isr: 3,2,1
        Topic: sugs_topic       Partition: 8    Leader: 1       Replicas: 1,3,2 Isr: 1,3,2
        Topic: sugs_topic       Partition: 9    Leader: 2       Replicas: 2,3,1 Isr: 2,3,1
```

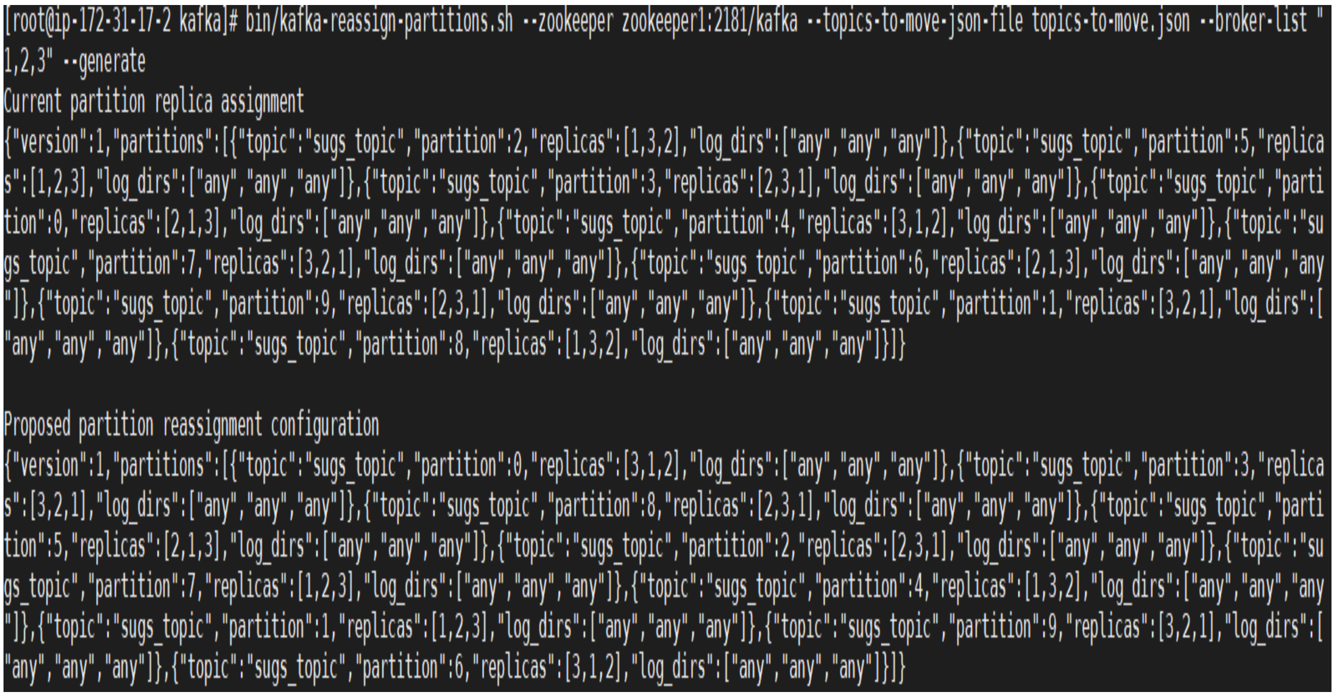
2. **Kafka-Reassign-Partition:**

This tool uses two JSON files for input. One is created by the user and another one will be created by tool as a proposed plan.We need to create one JSON file named ‘Topics-to-move’(could be any name) to get the current relpica assignment and proposed partition replica re-assignment in order to rebalance the partition across the cluster. In the file we will mention the topic that needs to be rebalanced.

```
$cat > topics-to-move.json
{
"topics": [{"topic":"sugs_topic"}],
"version":1
}
```

3. Next step would be generating a partition re-assignment configuration with below command so that we know the better partition distribution which can be used to rebalance.

```
$bin/kafka-reassign-partitions.sh --zookeeper zookeeper1:2181/kafka
--topics-to-move-json-file topics-to-move.json --broker-list "1,2,3"
--generate
```

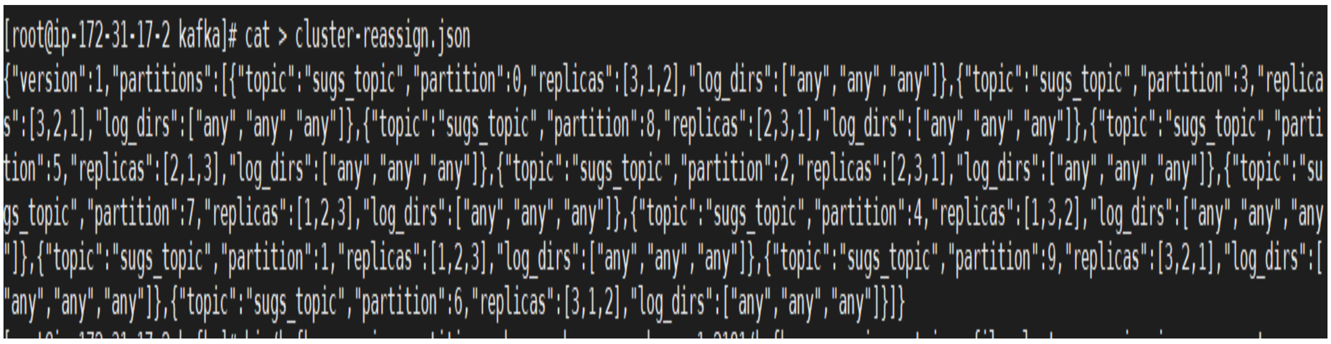


```
[root@ip-172-31-17-2 kafka]# bin/kafka-reassign-partitions.sh --zookeeper zookeeper1:2181/kafka --topics-to-move-json-file topics-to-move.json --broker-list "
1,2,3" --generate
Current partition replica assignment
{"version":1,"partitions":[{"topic":"sugs_topic","partition":2,"replicas":[1,3,2],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":5,"replica
s":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":3,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","parti
tion":0,"replicas":[2,1,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":4,"replicas":[3,1,2],"log_dirs":["any","any","any"]},{"topic":"su
gs_topic","partition":7,"replicas":[3,2,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":6,"replicas":[2,1,3],"log_dirs":["any","any","any
"]},{"topic":"sugs_topic","partition":9,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":1,"replicas":[3,2,1],"log_dirs":[
"any","any","any"]},{"topic":"sugs_topic","partition":8,"replicas":[1,3,2],"log_dirs":["any","any","any"]}]}

Proposed partition reassignment configuration
{"version":1,"partitions":[{"topic":"sugs_topic","partition":0,"replicas":[3,1,2],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":3,"replica
s":[3,2,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":8,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","parti
tion":5,"replicas":[2,1,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":2,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"su
gs_topic","partition":7,"replicas":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":4,"replicas":[1,3,2],"log_dirs":["any","any","any
"]},{"topic":"sugs_topic","partition":1,"replicas":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":9,"replicas":[3,2,1],"log_dirs":[
"any","any","any"]},{"topic":"sugs_topic","partition":6,"replicas":[3,1,2],"log_dirs":["any","any","any"]}]}
```

4. We created one more Json file  named ‘Cluster-reassign’ and we need to copy the proposed reassignment configuration to that file and execute the partition re-assignment tool. This updates the metadata and then starts to move data around to balance the load.

```
      $cat > cluster-reassign.json
```

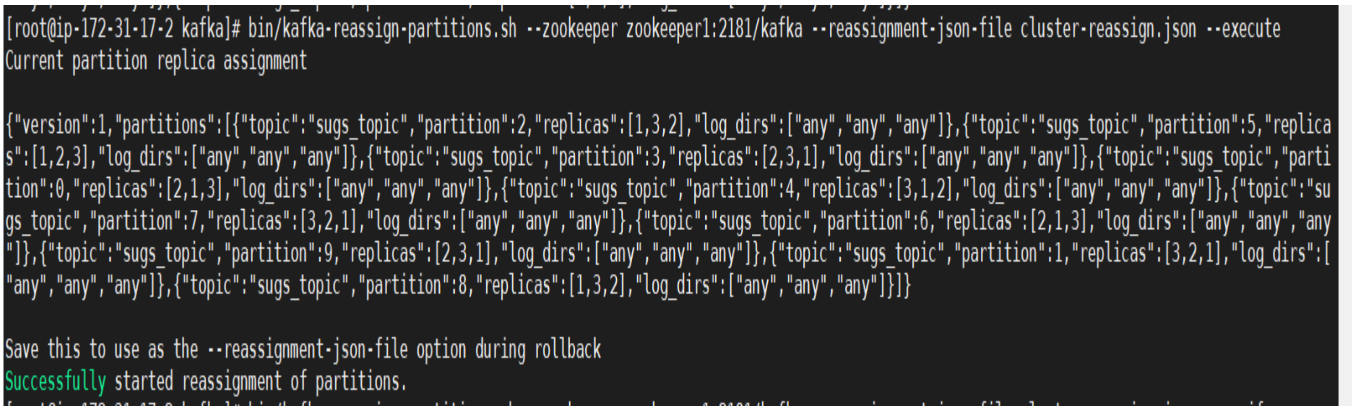


```
[root@ip-172-31-17-2 kafka]# cat > cluster-reassign.json
{"version":1,"partitions":[{"topic":"sugs_topic","partition":0,"replicas":[3,1,2],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":3,"replica
s":[3,2,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":8,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","parti
tion":5,"replicas":[2,1,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":2,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"su
gs_topic","partition":7,"replicas":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":4,"replicas":[1,3,2],"log_dirs":["any","any","any
"]},{"topic":"sugs_topic","partition":1,"replicas":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":9,"replicas":[3,2,1],"log_dirs":[
"any","any","any"]},{"topic":"sugs_topic","partition":6,"replicas":[3,1,2],"log_dirs":["any","any","any"]}]}
```

**Execution:**

```
$bin/kafka-reassign-partitions.sh --zookeeper zookeeper1:2181/kafka
--reassignment-json-file cluster-reassign.json --execute
```

Now the re-assignment of partitions has been completed successfully.



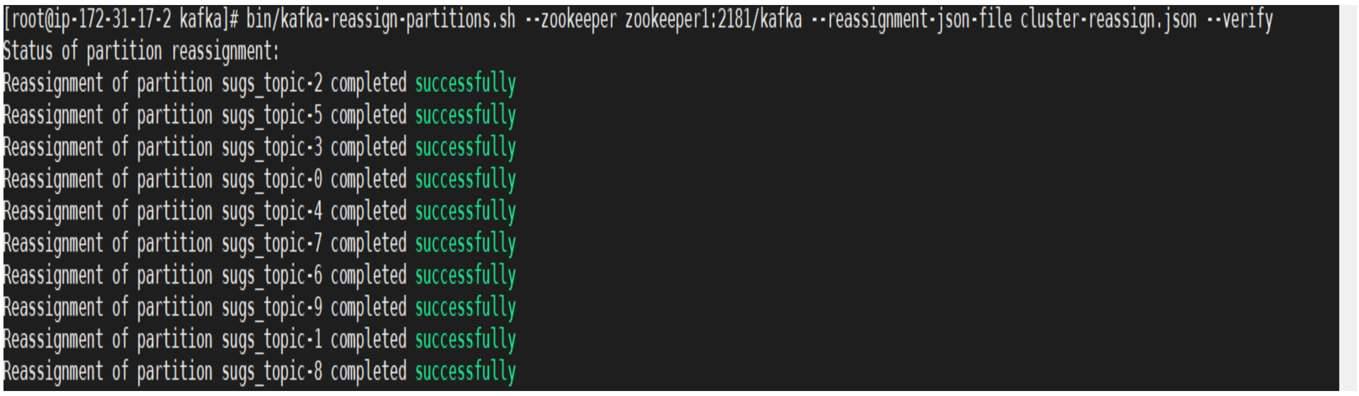
```
[root@ip-172-31-17-2 kafka]# bin/kafka-reassign-partitions.sh --zookeeper zookeeper1:2181/kafka --reassignment-json-file cluster-reassign.json --execute
Current partition replica assignment

{"version":1,"partitions":[{"topic":"sugs_topic","partition":2,"replicas":[1,3,2],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":5,"replica
s":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":3,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","parti
tion":0,"replicas":[2,1,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":4,"replicas":[3,1,2],"log_dirs":["any","any","any"]},{"topic":"su
gs_topic","partition":7,"replicas":[3,2,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":6,"replicas":[2,1,3],"log_dirs":["any","any","any
"]},{"topic":"sugs_topic","partition":9,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":1,"replicas":[3,2,1],"log_dirs":[
"any","any","any"]},{"topic":"sugs_topic","partition":8,"replicas":[1,3,2],"log_dirs":["any","any","any"]}]}

Save this to use as the --reassignment-json-file option during rollback
Successfully started reassignment of partitions.
```

## Verification:

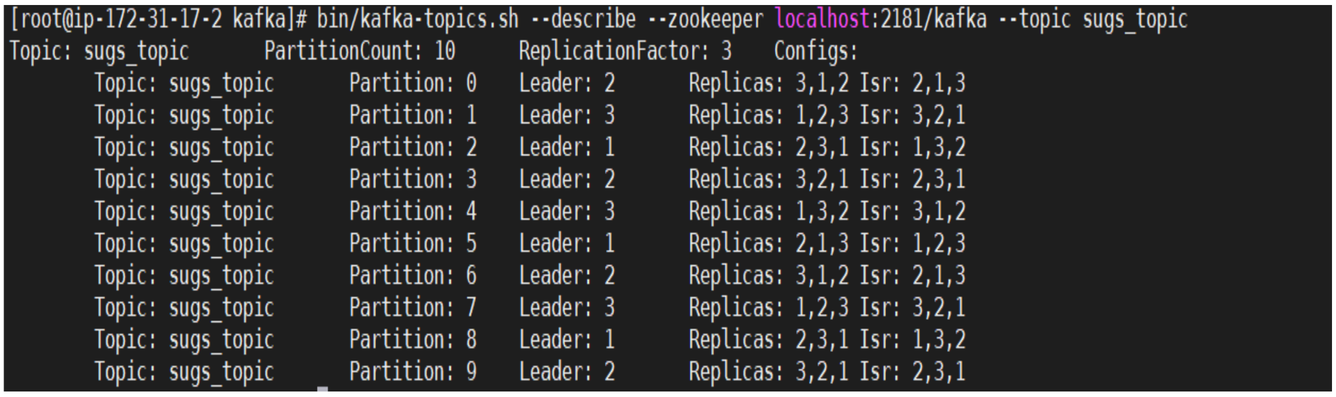
```
$bin/kafka-reassign-partitions.sh --zookeeper zookeeper1:2181/kafka
--reassignment-json-file cluster-reassign.json --verify
```



```
[root@ip-172-31-17-2 kafka]# bin/kafka-reassign-partitions.sh --zookeeper zookeeper1:2181/kafka --reassignment-json-file cluster-reassign.json --verify
Status of partition reassignment:
Reassignment of partition sugs_topic-2 completed successfully
Reassignment of partition sugs_topic-5 completed successfully
Reassignment of partition sugs_topic-3 completed successfully
Reassignment of partition sugs_topic-0 completed successfully
Reassignment of partition sugs_topic-4 completed successfully
Reassignment of partition sugs_topic-7 completed successfully
Reassignment of partition sugs_topic-6 completed successfully
Reassignment of partition sugs_topic-9 completed successfully
Reassignment of partition sugs_topic-1 completed successfully
Reassignment of partition sugs_topic-8 completed successfully
```

## 5. After Partition Re-assignment:

Now if we describe the same topic, we could see the partitions are re-assigned as proposed and looks rebalanced.

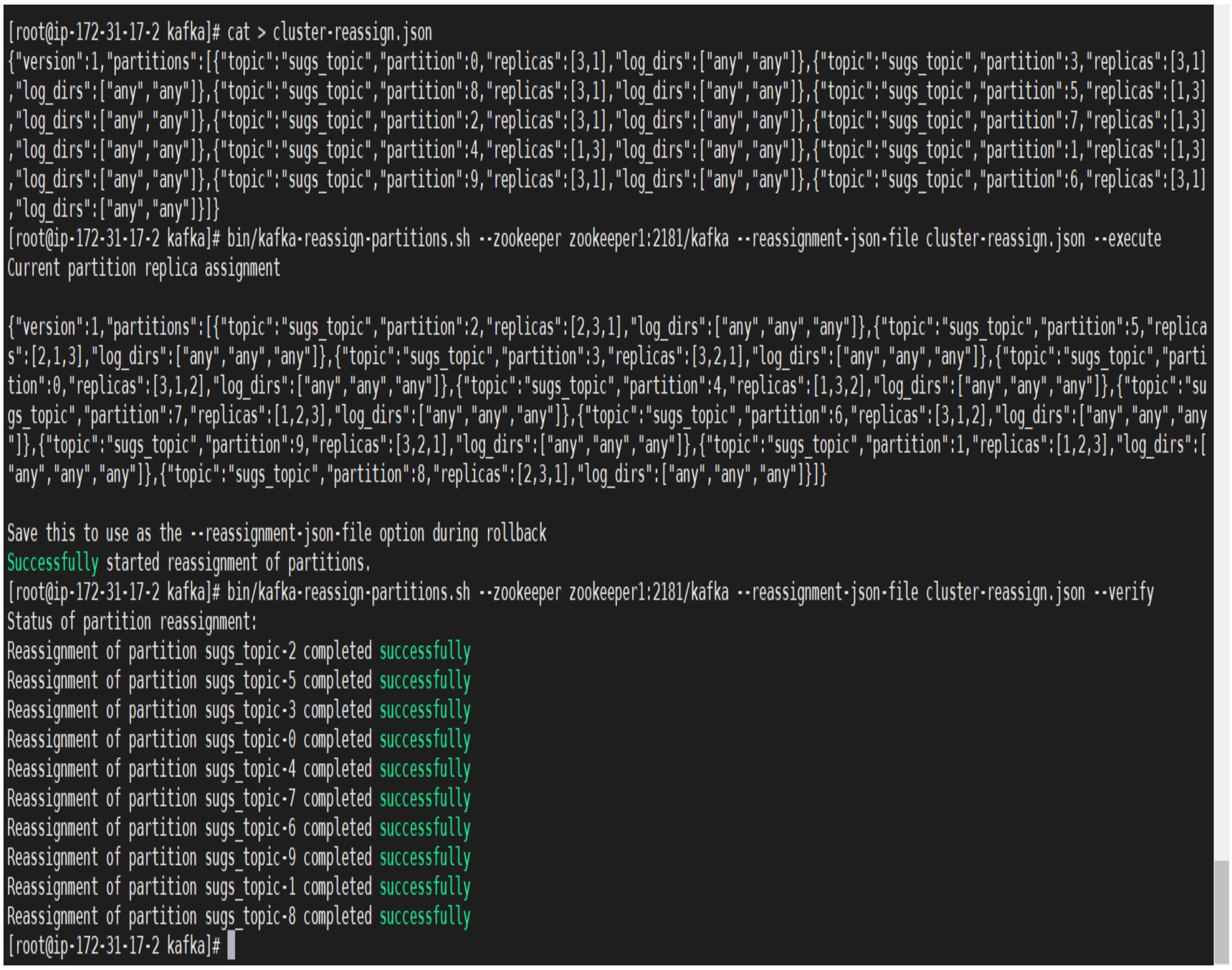


```
[root@ip-172-31-17-2 kafka]# bin/kafka-topics.sh --describe --zookeeper localhost:2181/kafka --topic sugs_topic
Topic: sugs_topic       PartitionCount: 10      ReplicationFactor: 3    Configs:
        Topic: sugs_topic       Partition: 0    Leader: 2       Replicas: 3,1,2 Isr: 2,1,3
        Topic: sugs_topic       Partition: 1    Leader: 3       Replicas: 1,2,3 Isr: 3,2,1
        Topic: sugs_topic       Partition: 2    Leader: 1       Replicas: 2,3,1 Isr: 1,3,2
        Topic: sugs_topic       Partition: 3    Leader: 2       Replicas: 3,2,1 Isr: 2,3,1
        Topic: sugs_topic       Partition: 4    Leader: 3       Replicas: 1,3,2 Isr: 3,1,2
        Topic: sugs_topic       Partition: 5    Leader: 1       Replicas: 2,1,3 Isr: 1,2,3
        Topic: sugs_topic       Partition: 6    Leader: 2       Replicas: 3,1,2 Isr: 2,1,3
        Topic: sugs_topic       Partition: 7    Leader: 3       Replicas: 1,2,3 Isr: 3,2,1
        Topic: sugs_topic       Partition: 8    Leader: 1       Replicas: 2,3,1 Isr: 1,3,2
        Topic: sugs_topic       Partition: 9    Leader: 2       Replicas: 3,2,1 Isr: 2,3,1
```

Note: Since we don’t have the large running cluster with data, we can’t able to notice any major difference in the partition re-assignment. This is for demonstrating purpose with small setup.

## 7.Removing one broker:

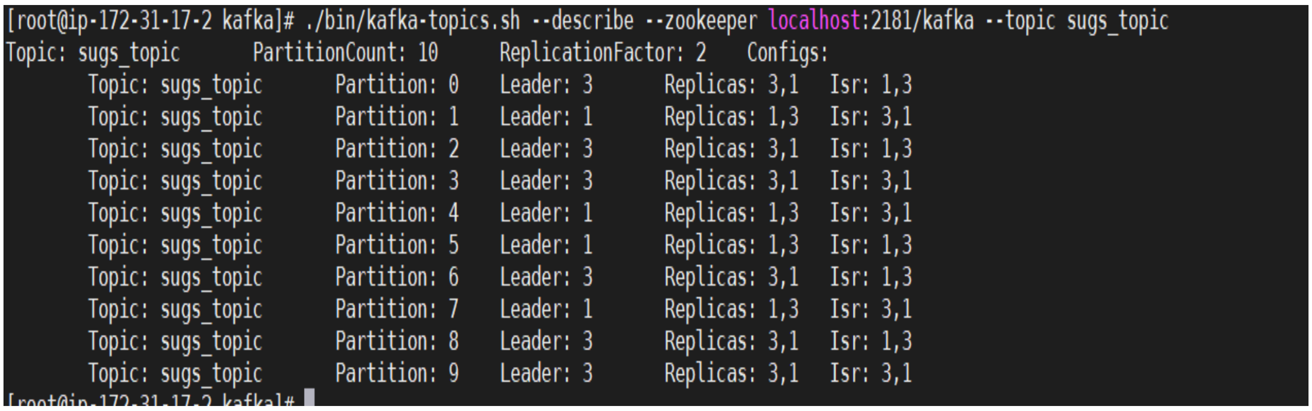
Now we tried to remove one of the brokers (Id 2) from the proposed partition re-assigned plan and proceed the same process again to check if the Leader partition will be distributed between two brokers (1 and 3) not in broker 2 and also the replication factor is also 2.



```
[root@ip-172-31-17-2 kafka]# cat > cluster-reassign.json
{"version":1,"partitions":[{"topic":"sugs_topic","partition":0,"replicas":[3,1],"log_dirs":["any","any"]},{"topic":"sugs_topic","partition":3,"replicas":[3,1]
,"log_dirs":["any","any"]},{"topic":"sugs_topic","partition":8,"replicas":[3,1],"log_dirs":["any","any"]},{"topic":"sugs_topic","partition":5,"replicas":[1,3]
,"log_dirs":["any","any"]},{"topic":"sugs_topic","partition":2,"replicas":[3,1],"log_dirs":["any","any"]},{"topic":"sugs_topic","partition":7,"replicas":[1,3]
,"log_dirs":["any","any"]},{"topic":"sugs_topic","partition":4,"replicas":[1,3],"log_dirs":["any","any"]},{"topic":"sugs_topic","partition":1,"replicas":[1,3]
,"log_dirs":["any","any"]},{"topic":"sugs_topic","partition":9,"replicas":[3,1],"log_dirs":["any","any"]},{"topic":"sugs_topic","partition":6,"replicas":[3,1]
,"log_dirs":["any","any"]}]}
[root@ip-172-31-17-2 kafka]# bin/kafka-reassign-partitions.sh --zookeeper zookeeper1:2181/kafka --reassignment-json-file cluster-reassign.json --execute
Current partition replica assignment

{"version":1,"partitions":[{"topic":"sugs_topic","partition":2,"replicas":[2,3,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":5,"replica
s":[2,1,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":3,"replicas":[3,2,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","parti
tion":0,"replicas":[3,1,2],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":4,"replicas":[1,3,2],"log_dirs":["any","any","any"]},{"topic":"su
gs_topic","partition":7,"replicas":[1,2,3],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":6,"replicas":[3,1,2],"log_dirs":["any","any","any
"]},{"topic":"sugs_topic","partition":9,"replicas":[3,2,1],"log_dirs":["any","any","any"]},{"topic":"sugs_topic","partition":1,"replicas":[1,2,3],"log_dirs":[
"any","any","any"]},{"topic":"sugs_topic","partition":8,"replicas":[2,3,1],"log_dirs":["any","any","any"]}]}

Save this to use as the --reassignment-json-file option during rollback
Successfully started reassignment of partitions.
[root@ip-172-31-17-2 kafka]# bin/kafka-reassign-partitions.sh --zookeeper zookeeper1:2181/kafka --reassignment-json-file cluster-reassign.json --verify
Status of partition reassignment:
Reassignment of partition sugs_topic-2 completed successfully
Reassignment of partition sugs_topic-5 completed successfully
Reassignment of partition sugs_topic-3 completed successfully
Reassignment of partition sugs_topic-0 completed successfully
Reassignment of partition sugs_topic-4 completed successfully
Reassignment of partition sugs_topic-7 completed successfully
Reassignment of partition sugs_topic-6 completed successfully
Reassignment of partition sugs_topic-9 completed successfully
Reassignment of partition sugs_topic-1 completed successfully
Reassignment of partition sugs_topic-8 completed successfully
[root@ip-172-31-17-2 kafka]# 
```


```
[root@ip-172-31-17-2 kafka]# ./bin/kafka-topics.sh --describe --zookeeper localhost:2181/kafka --topic sugs_topic
Topic: sugs_topic       PartitionCount: 10      ReplicationFactor: 2    Configs:
        Topic: sugs_topic       Partition: 0    Leader: 3       Replicas: 3,1   Isr: 1,3
        Topic: sugs_topic       Partition: 1    Leader: 1       Replicas: 1,3   Isr: 3,1
        Topic: sugs_topic       Partition: 2    Leader: 3       Replicas: 3,1   Isr: 1,3
        Topic: sugs_topic       Partition: 3    Leader: 3       Replicas: 3,1   Isr: 3,1
        Topic: sugs_topic       Partition: 4    Leader: 1       Replicas: 1,3   Isr: 3,1
        Topic: sugs_topic       Partition: 5    Leader: 1       Replicas: 1,3   Isr: 1,3
        Topic: sugs_topic       Partition: 6    Leader: 3       Replicas: 3,1   Isr: 1,3
        Topic: sugs_topic       Partition: 7    Leader: 1       Replicas: 1,3   Isr: 3,1
        Topic: sugs_topic       Partition: 8    Leader: 3       Replicas: 3,1   Isr: 1,3
        Topic: sugs_topic       Partition: 9    Leader: 3       Replicas: 3,1   Isr: 3,1
[root@ip-172-31-17-2 kafka]#
```

**Tip:**

**How to determine the number of partitions each of your Kafka topics requires?**

Kafka can be tuned to handle large messages. This can be done by configuring broker and consumer properties relating to maximum message and file size. You should adjust the exact number of partitions to number of consumers or producers, so that each consumer and producer achieve their target throughput. This calculation gives you a rough indication of the number of partitions. It's a good place to start. So a simple formula could be,

**Partitions = max(NP, NC)**

where:

NP is the number of required producers determined by calculating: TT/TP

NC is the number of required consumers determined by calculating: TT/TC

TT is the total expected throughput for our system

TP is the max throughput of a single producer to a single partition

TC is the max throughput of a single consumer from a single partition

**Drawback :**

Kafka-reassign-partitions has 2 flaws:

- It is not aware of partitions size
- Does not limit partition migration between brokers

**Future:**

“Topicmappr” by DATADOG is a drop-in replacement for the kafka kafka-reassign-partitions.sh script --generate function with a few additional features:

- Deterministic Output.
- Minimal movement broker replacements
- Configurable, rack-aware partition placement
- Replication factor updates
- Clear action summaries.