sahilsk / kafka-rebalancing.md

Last active 10 days ago • Report abuse

Kafka-rebalancing

kafka-rebalancing.md

Lets say i want to rebalance `my-sample-topic` topic on my kafka cluster

Create *topcis.json*

```
{
  "version": 1,
  "topics": [
    { "topic": "my-sample-topic"
  ]
}
```

where,

- topics : is an array and you can put multiple topic names there. But in production it's good to go one by one so that whole cluster is not impacted because of the failure of one topic rebalancing.

Generate Proposed partition reassignment configuration

```
kafka-reassign-partitions.sh --zookeeper $ZK_HOSTS  --topics-to-move-json-file topics.json --broker-list
"1,2,3,4,5" --generate
```

Copy the partition reassignment configuration from the ouptut in a file : *reassignment.json*

Finally, run reassignment

```
kafka-reassign-partitions.sh --zookeeper $ZK_HOSTS --reassignment-json-file reassignment.json --execute
```

Verify the status

```
kafka-reassign-partitions.sh --zookeeper $ZK_HOSTS --reassignment-json-file reassignment.json --verify
```

Once the reassignment is successful for all partitions, we run the preferred replica election tool to balance the topics and then run 'describe topic' to check balancing of topics.

```
kafka-preferred-replica-election.sh --zookeeper $ZK_HOSTS

kafka-topics.sh --zookeeper $ZK_HOSTS --describe | less
```

Some considerations:

If topic have too much data then reassingment will take lots of time. So, do this activity at least trffic time.

If data is too much, you can always reduce it by reducing the `retention.ms` i.e SLA

Reducing topic retention to 5 minutes

```
kafka-topics.sh --zookeeper  $ZK_HOSTS --alter --topic my-sample-topic --config retention.ms=300000

-- Do the reassignment ---
```

Now revert the retention

(Here am reverting to 8hours SLA)

```
kafka-topics.sh --zookeeper  $ZK_HOSTS --alter --topic my-sample-topic --config retention.ms=28800000
```

If you want to take one node out or down for maintenance or anything, you want to move data away from it.

Source

sleepcoding - Forcing kafka partition leaders

We now have controlled.shutdown.enable with default value true, which means broker will leave partition leadership before shutting down. But if you're upgrading from an earlier version, the broker will have some leaders and since it won't move them automatically, you might think there should be a manual way to force move them. There is, but it's some effort. But if your producers don't work with request.required.acks -1, **you will lose data** so you'd probably want to take that effort.

There is a tool called Reassign Partitions and another called Preferred Replica Leader Election in kafka replication tools. In summary, we tell kafka that the partition replicas have been changed to exclude the broker we want to shutdown. If you do not have enough spare brokers, you'll end up including that broker in the replicas anyway, but only in the last position. Second step will be signalling kafka to set preferred leaders. An example will clear things up.

We (and some users we chat around) have 3 kafka brokers, and topics work with replicationfactor 3. This way we think we have a cheap setup with good level of fault tolerance. All brokers have all the data and leadership and the load is distributed amongst them, if one of them goes down (unexpectedly or more importantly for some upgrade) the other two keep the data pipeline online. Normal state with brokerids 1,2,3 is like:

```
$ ./bin/kafka-topics.sh --describe --topic stream-log
Topic:stream-log    PartitionCount:9    ReplicationFactor:3 Configs:
    Topic: stream-log   Partition: 0    Leader: 3   Replicas: 3,2,1 Isr: 1,3,2
    Topic: stream-log   Partition: 1    Leader: 1   Replicas: 1,3,2 Isr: 1,3,2
    Topic: stream-log   Partition: 2    Leader: 2   Replicas: 2,1,3 Isr: 3,1,2
    Topic: stream-log   Partition: 3    Leader: 3   Replicas: 3,1,2 Isr: 3,1,2
    Topic: stream-log   Partition: 4    Leader: 1   Replicas: 1,2,3 Isr: 1,3,2
    Topic: stream-log   Partition: 5    Leader: 2   Replicas: 2,3,1 Isr: 1,3,2
    Topic: stream-log   Partition: 6    Leader: 3   Replicas: 3,2,1 Isr: 3,1,2
    Topic: stream-log   Partition: 7    Leader: 1   Replicas: 1,3,2 Isr: 1,3,2
    Topic: stream-log   Partition: 8    Leader: 2   Replicas: 2,1,3 Isr: 3,1,2
```

This example topic has 9 partitions and they are evenly distributed/replicated around 3 brokers. Say that we want to take broker 3 offline for some reason. What we want is for partitions 0, 3 and 6 (currently with leaders at broker 3) to set leaders to brokers either 1 or 2. But AFAIK there is no direct way to do that. Instead we prepare replica configs with broker 3 set as last with a json file like:

```
{"partitions": [
    {"topic": "stream-log", "partition": 0, "replicas": [2,1,3]},
    {"topic": "stream-log", "partition": 1, "replicas": [1,2,3]},
    {"topic": "stream-log", "partition": 2, "replicas": [2,1,3]},
    {"topic": "stream-log", "partition": 3, "replicas": [1,2,3]},
    {"topic": "stream-log", "partition": 4, "replicas": [1,2,3]},
    {"topic": "stream-log", "partition": 5, "replicas": [2,1,3]},
    {"topic": "stream-log", "partition": 6, "replicas": [2,1,3]},
    {"topic": "stream-log", "partition": 7, "replicas": [1,2,3]},
    {"topic": "stream-log", "partition": 8, "replicas": [2,1,3]}
  ],
  "version":1
}
```

We use this file to tell kafka to change replica configs using the first tool:

```
$ ./bin/kafka-reassign-partitions.sh --reassignment-json-file manual_assignment.json --execute
```

Now after a few seconds state will be:

```
$ ./bin/kafka-topics.sh --describe --topic stream-log
Topic:stream-log    PartitionCount:9    ReplicationFactor:3 Configs:
    Topic: stream-log   Partition: 0    Leader: 3   Replicas: 2,1,3 Isr: 3,1,2
    Topic: stream-log   Partition: 1    Leader: 1   Replicas: 1,2,3 Isr: 3,1,2
    Topic: stream-log   Partition: 2    Leader: 2   Replicas: 2,1,3 Isr: 3,1,2
    Topic: stream-log   Partition: 3    Leader: 3   Replicas: 1,2,3 Isr: 3,1,2
    Topic: stream-log   Partition: 4    Leader: 1   Replicas: 1,2,3 Isr: 3,1,2
    Topic: stream-log   Partition: 5    Leader: 2   Replicas: 2,1,3 Isr: 3,1,2
    Topic: stream-log   Partition: 6    Leader: 3   Replicas: 2,1,3 Isr: 3,1,2
    Topic: stream-log   Partition: 7    Leader: 1   Replicas: 1,2,3 Isr: 3,1,2
    Topic: stream-log   Partition: 8    Leader: 2   Replicas: 2,1,3 Isr: 3,1,2
```

We have changed the config but leaders are not changed. To force it:

```
$ ./bin/kafka-preferred-replica-election.sh
```

Kafka will prefer the first broker in each replica config to be the leader, so the final state will be:

```
$ ./bin/kafka-topics.sh --describe --topic stream-log
Topic:stream-log    PartitionCount:9    ReplicationFactor:3 Configs:
    Topic: stream-log   Partition: 0    Leader: 2   Replicas: 2,1,3 Isr: 3,1,2
    Topic: stream-log   Partition: 1    Leader: 1   Replicas: 1,2,3 Isr: 3,1,2
    Topic: stream-log   Partition: 2    Leader: 2   Replicas: 2,1,3 Isr: 3,1,2
    Topic: stream-log   Partition: 3    Leader: 1   Replicas: 1,2,3 Isr: 3,1,2
    Topic: stream-log   Partition: 4    Leader: 1   Replicas: 1,2,3 Isr: 3,1,2
    Topic: stream-log   Partition: 5    Leader: 2   Replicas: 2,1,3 Isr: 3,1,2
    Topic: stream-log   Partition: 6    Leader: 2   Replicas: 2,1,3 Isr: 3,1,2
    Topic: stream-log   Partition: 7    Leader: 1   Replicas: 1,2,3 Isr: 3,1,2
    Topic: stream-log   Partition: 8    Leader: 2   Replicas: 2,1,3 Isr: 3,1,2
```

Now you can shutdown broker 3 to upgrade or do whatever you want to do with it. ISRs will shrink for that time, but after the broker starts again it'll catch up.

As a last note, you'll probably want to go back to the original state after the operation, what you need to do is to reverse the operation: Reassign-partitions with the original config and start prefered-replica-election again. The first reassign-partitions will output the original state for the reference, so you can save and use that as the parameter for the second run.

Reference

- https://blog.imaginea.com/how-to-rebalance-topics-in-kafka-cluster/