Replication tools

1. Preferred Replica Leader Election Tool

What does the tool do?

With replication, each partition can have multiple replicas. The list of replicas for a partition is called the "assigned replicas". The first replica in this list is the "preferred replica". When topic/partitions are created, Kafka ensures that the "preferred replica" for the partitions across topics are equally distributed amongst the brokers in a cluster. In an ideal scenario, the leader for a given partition should be the "preferred replica". This guarantees that the leadership load across the brokers in a cluster are evenly balanced. However, over time the leadership load could get imbalanced due to broker shutdowns (caused by controlled shutdown, crashes, machine failures etc). This tool helps to restore the leadership balance between the brokers in the cluster. A summary of the steps that the tool does is shown below -

1. The tool updates the zookeeper path "/admin/preferred_replica_election" with the list of topic partitions whose leader needs to be moved to the preferred replica.
2. The controller listens to the path above. When a data change update is triggered, the controller reads the list of topic partitions from zookeeper.
3. For each topic partition, the controller gets the preferred replica (the first replica in the assigned replicas list). If the preferred replica is not already the leader and it is present in the isr, the controller issues a request to the broker that owns the preferred replica to become the leader for the partition.

Note that the tool only updates the zookeeper path and exits. The controller moves the leader for a partition to the preferred replica asynchronously.

How to use the tool?

```
bin/kafka-preferred-replica-election.sh --zookeeper localhost:12913/kafka --path-to-json-file
topicPartitionList.json
```

The tool takes a mandatory list of zookeeper hosts and an optional list of topic partitions provided as a json file. If the list is not provided, the tool queries zookeeper and gets all the topic partitions for the cluster. The tool exits after updating the zookeeper path "/admin/preferred_replica_election" with the topic partition list.

Example json file (This is optional. This can be specified to move the leader to the preferred replica for specific topic partitions)

```
{
 "partitions":
  [
    {"topic": "topic1", "partition": 0},
    {"topic": "topic1", "partition": 1},
    {"topic": "topic1", "partition": 2},
    {"topic": "topic2", "partition": 0},
    {"topic": "topic2", "partition": 1}
  ]
}
```

FAQ

What happens if the preferred replica is not in the ISR?

The controller will fail to move the leadership to the preferred replica if it is not in the ISR. This is to ensure that there is no dataloss. When the replica becomes "in-sync" with the leader, the tool can be run again to move the leader.

How to find if all the partitions have been moved to the "preferred replica" after running the tool?

ListTopicCommand is an excellent tool that provides an overview of all the topic partitions in the cluster. For each topic partition, it displays the leader, assigned replicas and current "in-sync" replica set. If the leader and the first replica in the assigned replica set are the same then the Preferred replica leader election" tool succeeded. If not, the tool failed and may have to be run again.

2. Topics tool

Kafka topics tool is handling all management operations related to topics:

- List and describe topics
- Create topics
- Change topics
- Delete topics

2.1 List and describe Topics

What does the tool do?

This tool lists the information for a given list of topics. If no topics are provided in the command line, the tool queries zookeeper to get all the topics and lists the information for them. The fields that the tool displays are - topic name, partition, leader, replicas, isr. Two optional arguments can be provided to the tool. If "under-replicated-partitions" is specified, the tool only provides information for those topic / partitions which have replicas that are under replicated. If "unavailable-partitions" is specified, the tool only provides information for those topic/partitions whose leader is not available.

How to use the tool?

```
# List only single topic named "topic1" (prints only topic name)
bin/kafka-topics.sh --list --zookeeper localhost:2121 --topic topic1

# List all topics (prints only topic names)
bin/kafka-topics.sh --list --zookeeper localhost:2121

# Describe only single topic named "topic1" (prints details about the topic)
bin/kafka-topics.sh --describe --zookeeper localhost:2121 --topic topic1

# Describe all topics (prints details about the topics)
bin/kafka-topics.sh --describe --zookeeper localhost:2121

# List info for topics which have under replicated count
bin/kafka-topics.sh --describe --zookeeper localhost:2121 --under-replicated-partitions

# List info for topics whose leader for a partition is not available
bin/kafka-topics.sh --describe --zookeeper localhost:2121 --unavailable-partitions
```

2.2 Create Topics

What does the tool do?

By default, Kafka auto creates topic if "auto.create.topics.enable" is set to true on the server. This creates a topic with a default number of partitions, replication factor and uses Kafka's default scheme to do replica assignment. Sometimes, it may be required that we would like to customize a topic while creating it. This tool helps to create a topic and also specify the number of partitions, replication factor and replica assignment list for the topic.

How to use the tool?

```
# Create topic with default settings
bin/kafka-topics.sh --create --zookeeper localhost:2181 --topic topic1

# Create topic with specific number of partitions and/or replicas
bin/kafka-topics.sh --create --zookeeper localhost:2181 --topic topic1 --replication-factor 3 --partitions 3


# Create topic with manual replica assignment
bin/kafka-topics.sh --create --zookeeper localhost:2181 --topic topic1 --replica-assignment 0:1:2,0:1:2,0:1:2

# Create topic with configuration override
bin/kafka-topics.sh --create --zookeeper localhost:2181 --topic topic1 --config min.insync.replicas=1
```

2.3 Add Partition to Topic

What does the tool do?

In Kafka partitions act as the unit of parallelism: messages of a single topic are distributed to multiple partitions that can be stored and served on different servers. Upon creation of a topic, the number of partitions for this topic has to be specified. Later on more partitions may be needed for this topic when the volume of this topic increases. This tool helps to add more partitions for a specific topic and also allow manual replica assignment of the added partitions.

How to use the tool?

```
# Increase number of partitions for topic
bin/kafka-topics.sh --alter --zookeeper localhost:2181 --topic topic1 --partitions 4

# Increase number of partitions with specific replica assignment
bin/kafka-topics.sh --alter --zookeeper localhost:2181 --topic topic1 --replica-assignment 0:1:2,0:1:2,0:1:2,2:
1:0 --partitions 4
```

2.4 Delete Topic

What does the tool do?

When topic deletion is enabled in the broker (`delete.topic.enable`), topics can be deleted using the Kafka Topics tool.

How to use the tool?

```
# Delete topic named topic1
bin/kafka-topics.sh --delete --zookeeper localhost:2181 --topic topic1
```

3. Change topic configuration

What does the tool do?

Kafka Confings tool can be used to modify topic configuration:

- Add new config options
- Change existing config options
- Remove config options

How to use the tool?

```
# Add new option or change exsiting option
bin/kafka-configs.sh --alter --zookeeper localhost:2181 --entity-name topic1 --entity-type topics --add-config
cleanup.policy=compact


# Remove exsiting option
bin/kafka-configs.sh --alter --zookeeper localhost:2181 --entity-name topic1 --entity-type topics --delete-
config cleanup.policy
```

4. Reassign Partitions Tool

What does the tool do?

The goal of this tool is similar to the Preferred Replica Leader Election Tool as to achieve load balance across brokers. But instead of only electing a new leader from the assigned replicas of a partition, this tool allows to change the assigned replicas of partitions – remember that followers also need to fetch from leaders in order to keep in sync, hence sometime only balance the leadership load is not enough.

A summary of the steps that the tool does is shown below -

1. The tool updates the zookeeper path "/admin/reassign_partitions" with the list of topic partitions and (if specified in the Json file) the list of their new assigned replicas.
2. The controller listens to the path above. When a data change update is triggered, the controller reads the list of topic partitions and their assigned replicas from zookeeper.
3. For each topic partition, the controller does the following:
3.1. Start new replicas in RAR - AR (RAR = Reassigned Replicas, AR = original list of Assigned Replicas)
3.2. Wait until new replicas are in sync with the leader
3.3. If the leader is not in RAR, elect a new leader from RAR
3.4 4. Stop old replicas AR - RAR
3.5. Write new AR
3.6. Remove partition from the /admin/reassign_partitions path

Note that the tool only updates the zookeeper path and exits. The controller reassign the replicas for the partitions asynchronously.

How to use the tool?

```
bin/kafka-reassign-partitions.sh

Option                                 Description
------                                 -----------
--bootstrap-server <String: Server(s)  the server(s) to use for
  to use for bootstrapping>              bootstrapping. REQUIRED if an
                                         absolution path of the log directory
                                         is specified for any replica in the
                                         reassignment json file
--broker-list <String: brokerlist>     The list of brokers to which the
                                         partitions need to be reassigned in
                                         the form "0,1,2". This is required
                                         if --topics-to-move-json-file is
                                         used to generate reassignment
                                         configuration
--disable-rack-aware                   Disable rack aware replica assignment
--execute                              Kick off the reassignment as specified
                                         by the --reassignment-json-file
                                         option.
--generate                             Generate a candidate partition
                                         reassignment configuration. Note
                                         that this only generates a candidate
                                         assignment, it does not execute it.
--reassignment-json-file <String:      The JSON file with the partition
  manual assignment json file path>      reassignment configurationThe format
                                         to use is -
                                       {"partitions":
                                               [{"topic": "foo",
                                                 "partition": 1,
                                                 "replicas": [1,2,3],
                                                 "log_dirs": ["dir1","dir2","dir3"]
                                         }],
                                       "version":1
                                       }
                                       Note that "log_dirs" is optional. When
                                         it is specified, its length must
                                         equal the length of the replicas
                                         list. The value in this list can be
                                         either "any" or the absolution path
                                         of the log directory on the broker.
                                         If absolute log directory path is
                                         specified, it is currently required
                                         that the replica has not already
                                         been created on that broker. The
                                         replica will then be created in the
                                         specified log directory on the
                                         broker later.
--throttle <Long: throttle>            The movement of partitions will be
                                         throttled to this value (bytes/sec).
                                         Rerunning with this option, whilst a
                                         rebalance is in progress, will alter
                                         the throttle value. The throttle
                                         rate should be at least 1 KB/s.
                                         (default: -1)
--timeout <Long: timeout>              The maximum time in ms allowed to wait
                                         for partition reassignment execution
                                         to be successfully initiated
                                         (default: 10000)
--topics-to-move-json-file <String:    Generate a reassignment configuration
  topics to reassign json file path>     to move the partitions of the
                                         specified topics to the list of
                                         brokers specified by the --broker-
                                         list option. The format to use is -
                                       {"topics":
                                               [{"topic": "foo"},{"topic": "foo1"}],
                                       "version":1
                                       }
--verify                               Verify if the reassignment completed
                                         as specified by the --reassignment-
                                         json-file option. If there is a
                                         throttle engaged for the replicas
                                         specified, and the rebalance has
                                         completed, the throttle will be
                                         removed
--zookeeper <String: urls>             REQUIRED: The connection string for
                                         the zookeeper connection in the form
                                         host:port. Multiple URLS can be
                                         given to allow fail-over.
```

Please note that by default the script runs in a dry-run mode and does not initiate the partition movement. Only when the --execute option is specified, the tool proceeds to start the partition movement

Cluster Expansion

The partition reassignment tool can be used to expand an existing Kafka cluster. Cluster expansion involves including brokers with new broker ids in a Kafka cluster. Typically, when you add new brokers to a cluster, they will not receive any data from existing topics until this tool is run to assign existing topics/partitions to the new brokers. The tool allows 2 options to make it easier to move some topics in bulk to the new brokers. These 2 options are a) topics to move b) list of newly added brokers. Using these 2 options, the tool automatically figures out the placements of partitions for the topics on the new brokers and generates new JSON data which can be used in the next step (with the `--reassignment-json-file` option) to execute the move.

The following example moves 2 topics (foo1, foo2) to newly added brokers in a cluster (5,6,7)

```
$ ./bin/kafka-reassign-partitions.sh --topics-to-move-json-file topics-to-move.json --broker-list "5,6,7" --
generate --zookeeper localhost:2181

$ cat topics-to-move.json
{"topics":
     [{"topic": "foo1"},{"topic": "foo2"}],
     "version":1
}
```

Selectively moving some partitions to a broker

The partition movement tool can also be moved to selectively move some replicas for certain partitions over to a particular broker. Typically, if you end up with an unbalanced cluster, you can use the tool in this mode to selectively move partitions around. In this mode, the tool takes a single file which has a list of partitions to move and the replicas that each of those partitions should be assigned to.

The following example moves 1 partition (foo-1) from replicas 1,2,3 to 1,2,4

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

$ cat partitions-to-move.json
{"partitions":
             [{"topic": "foo",
               "partition": 1,
               "replicas": [1,2,4] }],
  "version":1
}
```

5. StateChangeLogMerger Tool

What does the tool do ?

The goal of this tool is to collect data from the brokers in a cluster and format it in a central log to help troubleshoot issues with state changes. Every broker in a Kafka cluster emits a state-change.log that logs the lifecycle of every state change received by the broker. Often times, there is some problem with leader election for a subset of topics/partitions and the question is what caused the problem. In order to answer this question, we need a global view of state changes in the kafka cluster, possibly filtered on a time range and/or specific topics/partitions. This is exactly what the StateChangeLogMerger tool does. It takes in a list of state-change.log files, merges them in time order, filters on a certain time range if specified by the user, filters on topics/partitions if specified by the user, and outputs a merged and formatted state-change.log that is easy to query and understand the root cause.

How to use the tool ?

```
nnarkhed-mn:kafka-git-idea nnarkhed$ ./bin/kafka-run-class.sh kafka.tools.StateChangeLogMerger
Provide arguments to exactly one of the two options "[logs]" or "[logs-regex]"
Option                                  Description
------                                  -----------
--end-time <end timestamp in the        The latest timestamp of state change
  format java.text.                       log entries to be merged (default:
  SimpleDateFormat@f17a63e7>              9999-12-31 23:59:59,999)
--logs <file1,file2,...>                Comma separated list of state change
                                          logs or a regex for the log file
                                          names
--logs-regex <for example: /tmp/state-  Regex to match the state change log
  change.log*>                            files to be merged
--partitions <0,1,2,...>                Comma separated list of partition ids
                                          whose state change logs should be
                                          merged
--start-time <start timestamp in the    The earliest timestamp of state change
  format java.text.                       log entries to be merged (default:
  SimpleDateFormat@f17a63e7>              0000-00-00 00:00:00,000)
--topic <topic>                         The topic whose state change logs
                                          should be merged
```