Create initial data file

On a node where you can do some work with relatively large files, log on as the kafka users and copy the kawkfa files to a working directory.

Use the dump-partitions.sh command to create a data file of all existing topics and partitions.

sudo su - kafka

mkdir -p work/data

cd work

<git command>

./dump-partitions.sh

The dump-partitions.sh script will provide its current status of walking the entire Kafka topic list and then display information regarding the data file name and attributes.

The file that is created by this script has the naming convention of:

AllTopicsDetails.YYYYMMDD-HHMMSS

Once this file is created, make a note of its name, because we will be using this data file throughout the procedure below.

Preparations before bringing a broker node back into the cluster:

Log on to the node that is being returned to the cluster and become the kafka user and remove existing Kafka partition directories on the node being returned to the cluster:

ssh <hostname-of-host-to-return-to-cluster>

sudo su - kafka

cd /disk/kafka-logs

remove all partition directories and the contents

Make a note of the broker.id identifier for the node by running the command below and making a note of the broker.id entry. In the examples used throughout this procedure, we will use broker.id of 1234:

cat /disk/kafka-logs/meta.properties

As user kafka, create the json file that will remove the broker.id from all partitions. In our example for this procedure, we are going to say that the broker.id of the node returning to the cluster is 1234 and that the data file name is AllTopicsDetails.20170512-123456 :

./kawkfa-pre-shutdown.sh AllTopicsDetails.20170512-123456 1234 > 1234-pre-shutdown.json

Due to an existing bug in the kawkfa-pre-shutdown.sh script, we have to edit the 1234-pre-shutdown.json file with vi to remove the comma at the end of the last partition entry.  For example, we would change:

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 39, "replicas": [1132,1118]},

  ],

  "version":1

}

to

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 39, "replicas": [1132,1118]}

  ],

  "version":1

}

Now run the doReassign.sh to schedule the partition reassignment with the Controller node:

./doReassign.sh 1234-pre-shutdown.json

This will submit the reassignment job.  It will take a few minutes to complete.  You can watch for the completion of the reassignment job by running the Zookeeper zkCli.sh script and watching the /admin section:

sudo su - zookeeper

/usr/hdp/current/zookeeper-client/bin/zkCli.sh -server mykafkanode:2181

Within the zkCli.sh utility, run the following command:

ls /admin

if the output includes an entry named "reassign\_partitions", then the reassign partitions request above is still in progress.

[zk: mykafkanode:2181(CONNECTED) 0] ls /admin

[reassign\_partitions, delete\_topics]

Only after the reassign\_partitions entry is no longer displayed in the output is the job that was kicked-off by the above ” ./doReassign.sh 1234-pre-shutdown.json" command actually completed.

Once this completion is indicated by the removal of the "reassign\_partitions" entry, run a spot-check of Kafka topics to confirm that the broker.id  to be returned (in this case 1234) has been removed from all partitions.

You can do this with the following command:

./checkTopic.sh <topic-name> | grep <broker-id>

in our example, we could use:

./checkTopic  MY.BIG.KAFKA.TOPIC  | grep 1234

As long as there are no entries found where 1234 is a replicas entry, then we are assured that the 1234 broker has been removed from the topic.  Spot check a number of topics if you like to confirm.

Start up the Kafka Broker on the node to be returned to the cluster:

Now that we have removed the stale partition directories from /disk/kafka-logs and removed the broker.id (1234) from all partitions, we can now start the Kafka Broker in question without worrying that the node will cause a disruption on the cluster.

From the Ambari webGUI, find the node with the stopped Kafka broker and click the "Stopped" button and select "Start".

Watch for the Kafka broker to start up successfully, and then proceed to the portion of the procedure where we repopulate the topic partitions in a controlled manner.

Repopulate the Kafka broker with partition Replicas in a controlled manner:

From the kafka account again, run the following command to create the json file that adds the broker.id back to the replicas list, but does so in such a way as to not make the broker the preferred replica.

./kawkfa-post-startup.sh  AllTopicsDetails.20170512-123456  1234 > 1234-post-startup.json

This creates a complete repopulation json file.  If your Kafka cluster is currently over-subscribed, you may want to break this json file up in files of 50 operations each.

To do this, use the following split command to create as many files as needed in order to have all of them with 50 or less entries:

split  -l  50  1234-post-startup.json  1234-post-startup-50-

This will create a number of files that look something like this:

-rw-r--r--  1 kafka hadoop     4176 May 15 11:01 1234-post-startup-50-aa

-rw-r--r--  1 kafka hadoop     4232 May 15 11:01 1234-post-startup-50-ab

-rw-r--r--  1 kafka hadoop     4742 May 15 11:01 1234-post-startup-50-ac

-rw-r--r--  1 kafka hadoop     4543 May 15 11:01 1234-post-startup-50-ad

-rw-r--r--  1 kafka hadoop     4554 May 15 11:01 1234-post-startup-50-ae

Now edit each of these files to remove the lines that have:

{"partitions": [

and

  ],

  "version":1

}

additionally, remove the very last comma at the end of each file:

change:

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 53, "replicas": [1128,1118,1234]},

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 82, "replicas": [1107,1108,1234]},

to

{"topic": "MY.BIG.KAFKA.TOPIC”, "partition": 53, "replicas": [1128,1118,1234]},

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 82, "replicas": [1107,1108,1234]}

at the bottom of each file.

Now, we need to make the json files complete by adding additional information to the top and the bottom of the file with the catit.sh command:

./catit.sh 1234-post-startup-50-

This will create json files similar to the below:

-rw-r--r--  1 kafka hadoop     4176 May 15 11:01 1234-post-startup-50-aa.json

-rw-r--r--  1 kafka hadoop     4232 May 15 11:01 1234-post-startup-50-ab.json

-rw-r--r--  1 kafka hadoop     4742 May 15 11:01 1234-post-startup-50-ac.json

-rw-r--r--  1 kafka hadoop     4543 May 15 11:01 1234-post-startup-50-ad.json

-rw-r--r--  1 kafka hadoop     4554 May 15 11:01 1234-post-startup-50-ae.json

Now we can begin to run one-by-one these json file in order to repopulate the partition replicas on the node being returned to the cluster.  We do this by running each of the files one by one and letting each reassignment request complete after each command:

./doReassign.sh  1117-post-startup-50-aa.json

If the reassignment request is submitted successfully, the json file will be moved to the "~kafka/done" directory and the rollback information will be placed in a file with .rollback appended to the name of the file.

In our example, if the reassignment request was accepted, a file named 1234-post-startup-50-aa.json.rollback would be created and the 1234-post-startup-50-aa.json file will have been moved to the done directory.

You can watch for the completion of the reassignment request by running the zkCli.sh script and checking the /admin section.  If an entry for "reassign\_partitions" is returned, then the last repopulation request has not yet completed:

As the zookeeper user on one of your zookeeper nodes:

/usr/hdp/current/zookeeper-client/bin/zkCli.sh -server  myzookeepernode:2181

[zk: myzookeepernode:2181(CONNECTED) 0] ls /admin

[reassign\_partitions, delete\_topics]

In the above example the reassign\_partitions exists, so the work for that repopulation has not completed yet.  Wait for that to disappear from the listing before running the next ./doReassign.sh command on the next json file.

The amount of time that the reassign\_partitions operation takes to complete will vary substantially depending on the topic involved.  Some may take 30 minutes while other may take a matter of seconds to complete.

Normalize the cluster:

Once all partitions have been repopulated on the returning Kafka broker, run the dump-partitions.sh command again to create a new data file and confirm that all partitions have three replicas and all three replicas are in Isr.

If there are any entries remaining that do not pass the above test, create manual json file to repopulate them as needed.

Using the original AllTopicsDetails.YYYYMMDD-HHMMSS file that use used with the previous kawkfa-pre-shutdown.sh  and kawkfa-post-startup.sh scripts above, run the following command to create a json file to normalize the replicas list for the partitions that were repopulated:

./kawkfa-normalize.sh  AllTopicsDetails.20170508-123456  1234 > 1234-normalize.json

Again, edit the output file with vi, and remove the last comma from the last partition entry:

change:

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 23, "replicas": [1234,1101,1102]},

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 24, "replicas": [1117,1103,1234]},

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 49, "replicas": [1129,1234,1128]}**,**

  ],

  "version":1

}

to:

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 23, "replicas": [1234,1101,1102]},

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 24, "replicas": [1117,1103,1234]},

{"topic": "MY.BIG.KAFKA.TOPIC", "partition": 49, "replicas": [1129,1234,1128]}

  ],

  "version":1

}

save the json file and then as the kafka user submit the reassignment job with:

./doReassign.sh  1234-normalize.json

This request will take a substantial amount of time to queue to the system and to execute.  Be patient as it normalizes the broker.ids to allow the 1234 broker to become the Leader of the partition in a controlled manner.