KIP-727: Add --under-preferred-replica-partitions option to describe topics command

Status

Current state: *"Under Discussion"*

Discussion thread: https://lists.apache.org/thread.html/r27728a3b265d9ca26b2ef3672ad632e1a5b8da264ba1897cb01f0cca%40%3Cdev.kafka.apache.org%3E

JIRA: KAFKA-12556 - Getting issue details... STATUS

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Whether the preferred replica is the partition leader directly affects the external output traffic of the broker. When the preferred replica of all partitions becomes the leader, the external output traffic of the broker will be in a balanced state. When there are a large number of partition leaders that are not preferred replicas, it will be destroyed this state of balance.

Currently, the controller will periodically check the unbalanced ratio of the partition preferred replicas (if enabled) to trigger the preferred replica election, or manually trigger the election through the kafka-leader-election tool. However, if we want to know which partition leader is in the non-preferred replica, we need to look it up in the controller log or judge ourselves from the topic details list.

We hope to add an interface to TopicCommand to directly obtain the list of partitions whose leader is in a non-preferred replica.

One of the things a user might do after running this command would be to use `kafka-preferred-replica-election.sh` to try electing the preferred leader for (some of) those partitions.

kafka-preferred-replica-election.sh and kafka-leader-election.sh tools reads a JSON file containing the partitions which should have the preferred leader elected.

```
Therefore, we hope to be able to output json format.
```

Public Interfaces

kafka-topic.sh:

The topic command will accept the **--non-preferred-leader** or**--non-preferred-leader-json** options when using the **--bootstrap-server** option.

In order to reflect that the non-preferred replica was elected as the leader of the partition, I set **auto.leader.rebalance.enable=false**

1. --non-preferred-leader for the more human readable output, one example of use:

kafka-topics.sh --bootstrap-server localhost:9092 --describe **--non-preferred-leader**

Topic: test-2 Partition: 1 Leader: 1002 Replicas: 1001,1002 Isr: 1002,1001
Topic: test-2 Partition: 3 Leader: 1006 Replicas: 1005,1006 Isr: 1006,1005
Topic: test-2 Partition: 5 Leader: 1003 Replicas: 1004,1003 Isr: 1003,1004
Topic: test-2 Partition: 7 Leader: 1006 Replicas: 1001,1006 Isr: 1006,1001
Topic: test-2 Partition: 9 Leader: 1003 Replicas: 1005,1003 Isr: 1003,1005
Topic: test-2 Partition: 11 Leader: 1002 Replicas: 1004,1002 Isr: 1002,1004
Topic: test-1 Partition: 0 Leader: 1006 Replicas: 1001,1006 Isr: 1006,1001
Topic: test-1 Partition: 2 Leader: 1003 Replicas: 1005,1003 Isr: 1003,1005
Topic: test-1 Partition: 4 Leader: 1002 Replicas: 1004,1002 Isr: 1002,1004
Topic: test-1 Partition: 7 Leader: 1003 Replicas: 1006,1003 Isr: 1003,1006
Topic: test-1 Partition: 8 Leader: 1004 Replicas: 1005,1004 Isr: 1004,1005
Topic: test-3 Partition: 0 Leader: 1006 Replicas: 1001,1006 Isr: 1006,1001
Topic: test-3 Partition: 2 Leader: 1003 Replicas: 1005,1003 Isr: 1003,1005
Topic: test-3 Partition: 4 Leader: 1002 Replicas: 1004,1002 Isr: 1002,1004
Topic: test-3 Partition: 7 Leader: 1003 Replicas: 1006,1003 Isr: 1003,1006
Topic: test-3 Partition: 8 Leader: 1004 Replicas: 1005,1004 Isr: 1004,1005

...

2. --non-preferred-leader-json for the JSON output, one example of use:

kafka-topics.sh --bootstrap-server localhost:9092 --describe **--non-preferred-leader-json**

{"partitions":[{"topic":"test-3","partition":2},{"topic":"test-1","partition":7},{"topic":"test-3","partition":7},{"topic":"test-1","partition":8},{"topic":"test-2","partition":9},{"topic":"test-3","partition":4},{"topic":"test-2","partition":7},{"topic":"test-3","partition":0},{"topic":"test-1","partition":2},{"topic":"test-2","partition":3},{"topic":"test-2","partition":1},{"topic":"test-1","partition":0},{"topic":"test-3","partition":8},{"topic":"test-2","partition":11},{"topic":"test-1","partition":4},{"topic":"test-2","partition":5}]}

3. We can direct the output of example two to a json file named **preferred.json**, Then use the kafka-leader-election.sh tool to trigger the preferred replica election.

kafka-leader-election.sh --bootstrap-server localhost:9092 --election-type PREFERRED --path-to-json-file **preferred.json**

Successfully completed leader election (PREFERRED) for partitions test-3-2, test-2-1, test-1-0, test-1-2, test-3-4, test-2-3, test-2-5, test-1-4, test-3-7, test-3-8, test-2-7, test-1-7, test-3-0, test-2-9, test-1-8, test-2-11

Proposed Changes

Make some changes in TopicCommand.scala

TopicCommand.scala

```
    override def describeTopic(opts: TopicCommandOptions): Unit = {
      val topics = getTopics(opts.topic, opts.excludeInternalTopics)
      ensureTopicExists(topics, opts.topic, !opts.ifExists)

      if (topics.nonEmpty) {
        val allConfigs = adminClient.describeConfigs(topics.map(new ConfigResource(Type.TOPIC, _)).
asJavaCollection).values()
        val liveBrokers = adminClient.describeCluster().nodes().get().asScala.map(_.id())
        val topicDescriptions = adminClient.describeTopics(topics.asJavaCollection).all().get().values().asScala
        val describeOptions = new DescribeOptions(opts, liveBrokers.toSet)
        val topicPartitions = topicDescriptions
          .flatMap(td => td.partitions.iterator().asScala.map(p => new TopicPartition(td.name(), p.
partition())))
          .toSet.asJava
        val reassignments = listAllReassignments(topicPartitions)

        if (opts.reportNonPreferredLeaderJson) {
          var nonPreferredLeader = Set.empty[TopicPartition]
          for (td <- topicDescriptions) {
            val topicName = td.name
            val config = allConfigs.get(new ConfigResource(Type.TOPIC, topicName)).get()
            val sortedPartitions = td.partitions.asScala.sortBy(_.partition)

            nonPreferredLeader = nonPreferredLeader.++(sortedPartitions.toSet.filter { tpInfo =>
              val reassignment = reassignments.get(new TopicPartition(topicName, tpInfo.partition))
              val partitionDesc = PartitionDescription(topicName, tpInfo, Some(config), markedForDeletion =
false, reassignment)
              describeOptions.shouldPrintTopicPartition(partitionDesc)
            }.map(tp => new TopicPartition(topicName, tp.partition())))
          }
          println(formatAsJson(nonPreferredLeader))
        } else {
          for (td <- topicDescriptions) {
            val topicName = td.name
            val topicId = td.topicId()
            val config = allConfigs.get(new ConfigResource(Type.TOPIC, topicName)).get()
            val sortedPartitions = td.partitions.asScala.sortBy(_.partition)

            if (describeOptions.describeConfigs) {
              val hasNonDefault = config.entries().asScala.exists(!_.isDefault)
              if (!opts.reportOverriddenConfigs || hasNonDefault) {
                val numPartitions = td.partitions().size
                val firstPartition = td.partitions.iterator.next()
```

```
                val reassignment = reassignments.get(new TopicPartition(td.name, firstPartition.partition))
                val topicDesc = TopicDescription(topicName, topicId, numPartitions, getReplicationFactor
(firstPartition, reassignment), config, markedForDeletion = false)
                topicDesc.printDescription()
              }
            }

            if (describeOptions.describePartitions) {
              for (partition <- sortedPartitions) {
                val reassignment = reassignments.get(new TopicPartition(td.name, partition.partition))
                val partitionDesc = PartitionDescription(topicName, partition, Some(config), markedForDeletion
= false, reassignment)
                describeOptions.maybePrintPartitionDescription(partitionDesc)
              }
            }
          }
        }
      }
    }

    def shouldPrintTopicPartition(partitionDesc: PartitionDescription): Boolean = {
      describeConfigs ||
        shouldPrintUnderReplicatedPartitions(partitionDesc) ||
        shouldPrintUnavailablePartitions(partitionDesc) ||
        shouldPrintUnderMinIsrPartitions(partitionDesc) ||
        shouldPrintAtMinIsrPartitions(partitionDesc) ||
        shouldPrintNonPreferredLeader(partitionDesc)
    }

    def isNonPreferredLeader: Boolean = {
      hasLeader && !info.leader.equals(info.replicas.asScala.head)
    }

    private def shouldPrintNonPreferredLeader(partitionDescription: PartitionDescription): Boolean = {
      (opts.reportNonPreferredLeader || opts.reportNonPreferredLeaderJson) && partitionDescription.
isNonPreferredLeader
    }

    private val reportNonPreferredLeaderOpt = parser.accepts("non-preferred-leader",
      "if set when describing topics, only show partitions whose leader is not equal to the first replica in
the replica list. Not supported with the --zookeeper option.")
    private val reportNonPreferredLeaderJsonOpt = parser.accepts("non-preferred-leader-json",
      "if set when describing topics, only show partitions whose leader is not equal to the first replica in
the replica list and output as json format. Not supported with the --zookeeper option.")

    private val allReplicationReportOpts = Set(reportUnderReplicatedPartitionsOpt,
reportUnderMinIsrPartitionsOpt, reportAtMinIsrPartitionsOpt, reportUnavailablePartitionsOpt,
reportNonPreferredLeaderOpt, reportNonPreferredLeaderJsonOpt)

    def reportNonPreferredLeader: Boolean = has(reportNonPreferredLeaderOpt)
    def reportNonPreferredLeaderJson: Boolean = has(reportNonPreferredLeaderJsonOpt)

    private def formatAsJson(nonPreferredLeader: Set[TopicPartition]): String = {
      Json.encodeAsString(Map(
        "partitions" -> nonPreferredLeader.map { tp =>
          Map(
            "topic" -> tp.topic(),
            "partition" -> tp.partition()
          ).asJava
        }.asJava
      ).asJava)
    }

        # Some other minor changes omitted
```

Compatibility, Deprecation, and Migration Plan

The new option has no effect on existing usage.

Rejected Alternatives

1. A better name for the option would be --non-preferred-leader rather than --under-preferred-replica-partitions.

2.Because the other outputs from kafka-topics.sh don't naturally get used as inputs to the other tools, so such as `--output=json` top level option would not be a good fit.