Old KIP-179 - Change ReassignPartitionsCommand to use AdminClient

Note this was initially erroneously assigned as KIP-178, which was already taken, and has been reassigned KIP-179.

- Status
- Motivation
- Public Interfaces
- Proposed Changes
  - kafka-reassign-partitions.sh and ReassignPartitionsCommand
  - AdminClient: alterTopics()
  - AdminClient: replicaStatus()
  - Authorization
  - Network Protocol: AlterTopicsRequest and AlterTopicsResponse
  - Policy
  - Network Protocol: ReplicaStatusRequest and ReplicaStatusResponse
    - Implementation
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives

Status

Current state: Under Discussion *[One of "Under Discussion", "Accepted", "Rejected"]*

Discussion thread: *here (when initially misnumbered as KIP-178)* and here (when assigned KIP-179)

JIRA: *here* ⚠ *Unable to render Jira issues macro, execution error.*

⚠ *Unable to render Jira issues macro, execution error.*

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

Motivation

Describe the problems you are trying to solve.

Firstly, the `ReassignPartitionsCommand` (which is used by the `kafka-reassign-partitions.sh` tool) talks directly to ZooKeeper. This prevents the tool being used in deployments where only the brokers are exposed to clients (i.e. where the zookeeper servers are intentionally not exposed). In addition, there is a general push to refactor/rewrite/replace tools which need ZooKeeper access with equivalents which use the `AdminClient` API. Thus it is necessary to change the `ReassignPartitionsCommand` so that it no longer talks to ZooKeeper directly, but via an intermediating broker. Similar work is needed for the `kafka-topics.sh` tool (which can also change assignments and numbers of partitions and replicas), so common `AdminClient` and protocol APIs are desirable.

Secondly, `ReassignPartitionsCommand` currently has no proper facility to report progress of a reassignment; `--verify` can be used periodically to check whether the request assignments have been achieved. It would be useful if the tool could report progress better.

Public Interfaces

Briefly list any new interfaces that will be introduced as part of this proposal or any existing interfaces that will be removed or changed. The purpose of this section is to concisely call out the public contract that will come along with this feature.

A public interface is any change to the following:

- *Binary log format*
- *The network protocol and api behavior*
- Any class in the public packages under clientsConfiguration, especially client configuration
    - org/apache/kafka/common/serialization
    - org/apache/kafka/common
    - org/apache/kafka/common/errors
    - org/apache/kafka/clients/producer
    - org/apache/kafka/clients/consumer (eventually, once stable)
- *Monitoring*
- *Command line tools and arguments*
- *Anything else that will likely break existing users in some way when they upgrade*

Two new network protocol APIs will be added:

- `AlterTopicsRequest` and `AlterTopicsResponse`
- `ReplicaStatusRequest` and `ReplicaStatusResponse`

The `AdminClient` API will have two new methods added (plus overloads for options):

- `alterTopics(Collection<AlterTopics>)`
- `replicaStatus(Collection<Replica> replicas)`

The options accepted by `kafka-reassign-partitions.sh` command will change:

- `--zookeeper` will be deprecated, with a warning message
- a new `--bootstrap-server` option will be added
- a new `--progress` action option will be added

Proposed Changes

Describe the new thing you want to do in appropriate detail. This may be fairly extensive and have large subsections of its own. Or it may be a few sentences. Use judgement based on the scope of the change.

`kafka-reassign-partitions.sh` and `ReassignPartitionsCommand`

The `--zookeeper` option will be retained and will:

1. Cause a deprecation warning to be printed to standard error. The message will say that the `--zookeeper` option will be removed in a future version and that `--bootstrap-server` is the replacement option.
2. Perform the reassignment via ZooKeeper, as currently.

A new `--bootstrap-server` option will be added and will:

1. Perform the reassignment via the given intermediating broker.

Using both `--zookeeper` and `--bootstrap-server` in the same command will produce an error message and the tool will exit without doing the intended operation.

It is anticipated that a future version of Kafka would remove support for the `--zookeeper` option.

A new `--progress` action option will be added. This will only be supported when used with `--bootstrap-server`. If used with `--zookeeper` the command will produce an error message and the tool will exit without doing the intended operation. `--progress` will report on the synchronisation of each of the partitions and brokers in the reassignment given via the `--reassignment-json-file` option

For example:

```
# If the following command is used to start a reassignment
bin/kafka-reassign-partitions.sh --bootstrap-server localhost:9878 \
  --reassignment-json-file expand-cluster-reassignment.json \
  --execute

# then the following command will print the progress of
# that reassignment, then exit immediately
bin/kafka-reassign-partitions.sh --bootstrap-server localhost:9878 \
  --reassignment-json-file expand-cluster-reassignment.json \
  --progress
```

That might print something like the following:

```
Topic      Partition  Broker  Status
-----------------------------------
my_topic   0          0       In sync
my_topic   0          1       Behind: 10456 messages behind
asdf       0          1       Unknown topic
my_topic   42         1       Unknown partition
my_topic   0          42      Unknown broker
my_topic   1          0       Broker does not host this partition
```

Internally, the `ReassignPartitionsCommand` will be refactored to support the above changes to the options. An interface will abstract the commands currently issued directly to zookeeper.

There will be an implementation which makes the current calls to ZooKeeper, and another implementation which uses the `AdminClient` API described below.

In all other respects, the public API of `ReassignPartitionsCommand` will not be changed.

AdminClient: `alterTopics()`

The following methods will be added to `AdminClient` to support the ability to reassign partitions:

```
/**
 * Request alteration of the given topics. The request can change the number of
 * partitions, replication factor and/or the partition assignments.
 * This can be a long running operation as replicas are migrated between brokers,
 * therefore the returned result conveys whether the alteration has been
 * started, not that it is complete. Progress information
 * can be obtained by calling the lead broker's
 * {@link #replicaStatus(Collection)}.
 */
public AlterTopicsResult alterTopics(Collection<AlteredTopic> alteredTopics)
public AlterTopicsResult alterTopics(Collection<AlteredTopic> alteredTopics,  AlterTopicsOptions options)
```

Where:

```
public class AlteredTopic {
    public AlteredTopic(String name, int numPartitions, int replicationFactor, Map<Integer,List<Integer>>
replicasAssignment) {
        // ...
    }
    /** The name of the topic to alter. */
    public String name();
    /** The new number of partitions, or -1 if the number of partitions should not be changed. */
    public int numPartitions();
    /** The new replication factor, or -1 if the replication factor should not be changed. */
    public short replicationFactor();
    /**
     * The new assignments of partition to brokers, or the empty map
     * if the broker should assign replicas automatically.
     */
    Map<Integer,List<Integer>> replicasAssignment();
}

public class AlterTopicsOptions {
    public AlterTopicsOptions validateOnly(boolean validateOnly);
    public boolean validateOnly();
    public AlterTopicsOptions timeoutMs(long timeoutMs);
        public long timeoutMs();
 }

public class AlterTopicsResult {
    // package-access constructor
    /** A mapping of the name of a requested topic to the error for that topic. */
    Map<String, KafkaFuture<Void>> values();
    /** Return a future which succeeds if all the topic alterations were accepted. */
        KafkaFuture<Void> all();
}
```

AdminClient: `replicaStatus()`

The following methods will be added to `AdminClient` to support the progress reporting functionality:

```
/**
 * Query the replication status of the given partitions.
 */
public ReplicaStatusResult replicaStatus(Collection<TopicPartition> replicas)
public ReplicaStatusResult replicaStatus(Collection<TopicPartition> replicas,  ReplicaStatusOptions options)
```

Where:

```
public class ReplicaStatusOptions {

}

public class ReplicaStatusResult {
    public KafkaFuture<Map<TopicPartition, List<ReplicaStatus>>> all()
}

/**
 * Represents the replication status of a partition
 * on a particular broker.
 */
public class ReplicaStatus {
    /** The topic about which this is the status of */
    String topic()
    /** The partition about which this is the status of */
    int partition()
    /** The broker about which this is the status of */
    int broker()

    /**
     * The time (as milliseconds since the epoch) that
     * this status data was collected. In general this may
     * be some time before the replicaStatus() request time.
     */
    public long statusTime()

    /**
     * The number of messages that the replica on this broker is behind
     * the leader.
     */
    public long lag()

}
```

Authorization

With broker-mediated reassignment it becomes possible limit the authority to perform reassignment to something finer-grained than "anyone with access to zookeeper".

The reasons for reassignment are usually operational. For example, migrating partitions to new brokers when expanding the cluster, or attempting to find a more balanced assignment (according to some notion of balance). These are cluster-wide considerations and so authority should be for the reassign operation being performed on the cluster. Therefore `alterTopics()` will require `ClusterAction` on the `CLUSTER`.

`replicaStatus()` will require `Describe` on the `CLUSTER`.

Network Protocol: `AlterTopicsRequest` and `AlterTopicsResponse`

An `AlterTopicsRequest` will initiate the process of topic alteration/partition reassignment

```
AlterTopicsRequest => [alter_topic_requests] validate_only
  alter_topic_requests => topic num_partitions replication_factor [partition_assignment]
    topic => STRING
    num_partitions => INT32
    replication_factor => INT16
    partition_assignment => partition_id brokers
      partition_id => INT32
      brokers => [INT32]
  validate_only => BOOLEAN
  timeout => INT32
```

Where

FIELD	DESCRIPTION
`topic`	the topic name

num_partition	the number of partitions. A num_partitions of -1 that would mean "no change"
replication_factor	the replication factor. A replication_factor of -1 would mean "no change"
partition_id	the partition id
brokers	the ids of the assigned brokers for this partition
validate_only	true to just validate the request, but not actually alter the topics
timeout	the timeout, in ms, to wait for the topic to be altered.

An empty partition_assignment would mean that the broker should calculate a suitable assignment. Such broker calculated assignment is unlikely to be balanced.

It is not necessary to send an AlterTopicsRequest to the leader for a given partition. Any broker will do.

The AlterTopicsResponse enumerates those topics in the request, together with any error in initiating alteration:

```
AlterTopicsResponse => throttle_time_ms [topic_errors]
  throttle_time_ms => INT32
  topic_errors => topic error_code error_message
    topic => STRING
    error_code => INT16
    error_message => NULLABLE_STRING
```

Where

Field	Description
throttle_time_ms	duration in milliseconds for which the request was throttled
topic	the topic name
error_code	the error code for altering this topic
error_message	detailed error information

Possible values for error_code:

- CLUSTER_AUTHORIZATION_FAILED (31) Authorization failed
- INVALID_TOPIC_EXCEPTION (17) If the topic doesn't exist
- INVALID_PARTITIONS (37) If the num_partitions was invalid
- INVALID_REPLICATION_FACTOR (38) If the replication_factor was invalid
- UNKNOWN_MEMBER_ID (25) If any broker ids in the partition_assignment included an unknown broker id
- INVALID_REQUEST (42) If trying to modify the partition assignment and the number of partitions or the partition assignment and the replication factor in the same request. Or if duplicate topics appeared in the request.
- PARTITION_REASSIGNMENT_IN_PROGRESS (new)
- INVALID_REPLICA_ASSIGNMENT (39) If a partition, replica or broker id in the partition_assignment doesn't exist or is incompatible with the requested num_partitions and /or replication_factor. The error_message would contain further information.
- NONE (0) If the request was successful and the alteration/reassignment has been started.

As currently, it will not be possible to have multiple reassignments running concurrently, hence the addition of the PARTITION_REASSIGNMENT_IN_PROGRESS error code.

Policy

The existing CreateTopicPolicy can be used to apply a cluster-wide policy on topic configuration at the point of creation via the create.topic.policy.class.name config property. To avoid an obvious loophole, it is necessary to also be able to apply a policy to topic alteration. Maintaining two separate policies in sync is a burden both in terms of class implementation and configuring the policy. It seems unlikely that many use cases would require a different policy for alteration than creation. On the other hand, just applying the CreateTopicPolicy to alterations is undesirable because:

- Its name doesn't convey that it would be applied to alterations too
- Its API (specifically its RequestMetadata member class) includes topic configs (i.e. Map<String, String>) which is not part of the API for topic alteration even though it is part of the API for topic creation.
- It prevents any use cases which legitimately did need to apply a different policy for alteration than creation.

Finding a balance between compatibility with existing deployments, and not opening the loophole is difficult.

The existing `create.topic.policy.class.name` config would continue to work, and would continue to name an implementation of `CreateTopicPolicy`. That policy would be applied to alterations automatically. The topic's config would be presented to the `validate()` method (via the `RequestMetadata`) even though it's not actually part of the `AlterTopicsRequest`. The documentation for the interface and config property would be updated.

Network Protocol: `ReplicaStatusRequest` and `ReplicaStatusResponse`

A `ReplicaStatusRequest` requests information about the progress of a number of replicas.

```
ReplicaStatusRequest => [replica_status_requests]
  replica_status_requests => topic partition_id broker
    topic => STRING
    partition_id => INT32
    broker => INT32
```

Where

Field	Description
`topic`	a topic name
`partition_id`	a partition id of this topic
`broker`	a follower broker id for this partition

The response includes replication information for each of the replicas in the request:

```
ReplicaStatusResponse => [replica_status]
  replica_status => topic partition_id broker error_code status_time lag
    topic => STRING
    partition_id => INT32
    broker => INT32
    error_code => INT16
    status_time => INT64
    lag => INT64
```

Where

Field	Description
`topic`	the topic name
`partition_id`	the partition id of this topic
`broker`	the follower broker id
`error_code`	an error code
`status_time`	the time the status was current
`lag`	the lag (#messages) of this broker, for this partition

Anticipated errors are:

- `CLUSTER_AUTHORIZATION_FAILED` (31) Authorization failed. (**or the TOPIC?**)
- `INVALID_TOPIC_EXCEPTION` (17) The topic is not known
- `INVALID_PARTITIONS` (37) The `partion_id` of the given topic is not valid
- `UNKNOWN_MEMBER_ID` (25) The given `broker` id is not known.
- `UNKNOWN_TOPIC_OR_PARTITION` (3) The given `broker` is not a follower for the partition identified by `topic`, `partition`.
- `NONE` (0) if the status request completed normally,

Implementation

The `AdminClient.replicaStatus()` will make the underlying `ReplicaStatusRequest` to the *leader* for the given partition. This saves the need for every broker (because any broker could be the `--bootstrap-server`) to have knowledge of the replication status of every replica, which would be inefficient in network IO and/or memory use.

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*
- *If we are changing behavior how will we phase out the older behavior?*
- *If we need special migration tools, describe them here.*
- *When will we remove the existing behavior?*

Existing users of the `kafka-reassign-partitions.sh` will receive a deprecation warning when they use the `--zookeeper` option. The option will be removed in a future version of Kafka. If this KIP is introduced in version 1.0.0 the removal could happen in 2.0.0.

Rejected Alternatives

If there are alternative ways of accomplishing the same thing, what were they? The purpose of this section is to motivate why the design is the way it is and not some other way.

One alternative is to do nothing: Let the ReassignPartitionsCommand continue to communicate with ZooKeeper directly.

Another alternative is to do exactly this KIP, but without the deprecation of `--zookeeper`. That would have a higher long term maintenance burden, and would prevent any future plans to, for example, provide alternative cluster technologies than ZooKeeper.