KIP-919: Allow AdminClient to Talk Directly with the KRaft Controller Quorum and add Controller Registration

Status

Current state: Accepted

Discussion thread:

JIRA: ⚠ 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

Background

The KRaft controller was designed to be isolated from Kafka clients. This isolation helps prevent misbehaving clients from compromising the performance of the system. It also clarifies node roles: brokers are responsible for client traffic. However, there are certain edge cases where it is reasonable for clients to communicate with KRaft controllers.

Sometimes, we would like to target controllers directly. Typically this is so that we can perform an administrative operation without involving the brokers. DESCRIBE_QUORUM is a great example. This operation has nothing to do with the brokers, and may indeed be useful for debugging when other parts of the system are down. Another good example is using INCREMENTAL_ALTER_CONFIGS to make log4j level changes on a KRaft controller.

Overview

AdminClient

New Kafka admin clients which support KIP-919 will be able to target KRaft controllers directly. Because we don't support producing or consuming on the controllers, this only applies to admin clients, and not to producers or consumers.

The choice to communicate directly with the kcontroller quorum should not be taken lightly. In some cases, this bypasses safety checks built into the system. For example, when using an `incrementalAlterConfigs` operation to alter a broker configuration, normally the configuration change will be validated on the broker side first. However, when we send the RPC to the controller quorum directly, this validation step does not occur, since the broker is not involved.

In order to make it clear what is going on, communicating directly with the controller quorum will require special configurations and command-line flags. If these are not passed, the communication will not be allowed. Similarly, we will not be able to talk to brokers when using these configurations and flags.

Controller Registration

Once after starting up, Kafka controllers will register themselves with the active controller. This registration will include information about the endpoints which they possess, as well as information about whether they are ready to perform zk migration, and a randomly generated UUID uniquely identifying the specific incarnation of the controller. The registration will be persisted in the metadata log as hard state.

Each controller will check that the controller registration for its ID is as expected. If it is not, it will re-register. It examines the registration information found in the metadata log. This check is necessary to fix cases where a delayed message from an older incarnation of controller N somehow arrives later than a more recent registration for controller N. Although we expect this to be quite rare, it is possible.

Security

From the perspective of security, nothing is changing. It is already possible for custom clients to send operations directly to the controller. The only new ability we are adding is the ability to get back metadata responses. And the information they contain is already present in the MetadataResponses returned from the brokers.

We will continue to enforce access controls on all operations performed by the controller, by means of the Authorizer system. We also continue to recommend that users not be given direct network access to kcontrollers – for example, by putting them on a separate network, or by setting up a firewall.

Public Interfaces

bootstrap.controllers configuration

There will be a new `AdminClient` configuration, `bootstrap.controllers` . This configuration contains a comma-separated series of hostname:port entries. When this configuration is specified, the AdminClient will talk directly to the controller quorum and the brokers will not be involved.

`KafkaProducer` and `KafkaConsumer` will not support `bootstrap.controllers.` Only `AdminClient` will support it.

It is an error to set both `bootstrap.controllers` and `bootstrap.servers` . Only one can be set at a time. It is also an error to include broker endpoints in `--bootstrap-controller` . If we contact a broker via this mechanism, the command will fail.

Just as with `bootstrap.servers`, the supplied server list doesn't need to be exhaustive. As long as we can contact one of the provided controllers the RPC can proceed.

Command-line Changes

New Arguments

The following command-line tools will get a new `--bootstrap-controller` argument:

- kafka-acls.sh
- kafka-cluster.sh
- kafka-configs.sh
- kafka-delegation-tokens.sh
- kafka-features.sh
- kafka-metadata-quorum.sh
- kafka-metadata-shell.sh
- kafka-reassign-partitions.sh

When the `--bootstrap-controller` argument is used `--bootstrap-servers` must **not** be specified.

The `--bootstrap-controller` flag will set the `bootstrap.controllers` configuration described above. It will also clear the `bootstrap.servers` configuration if that has been set in some other way (for example, via a configuration file provided to the command-line tool).

The `--bootstrap-controller` flag will be documented as follows:

```
   --bootstrap-controller CONTROLLERS
                         A comma-separted list of bootstrap.controllers that can be supplied instead of
bootstrap-servers.
                         This is useful for administrators who wish to bypass the brokers.
```

Note that it is not necessary to specify the controller IDs when using `--bootstrap-controller` .

Here is an example `--bootstrap-controller` usage:

```
./bin/kafka-cluster.sh cluster-id --bootstrap-controller example.com:9090,example2.com:9090,example3.com:9090
```

Changes to kafka-metadata-shell.sh

The metadata shell will now have these arguments:

```
The Apache Kafka metadata tool

positional arguments:
  command               The command to run.

optional arguments:
  -h, --help            show this help message and exit
  --directory DIRECTORY, -d DIRECTORY
                        The __cluster_metadata-0 directory to read.
  --bootstrap-controller CONTROLLERS, -q CONTROLLERS
                        The bootstrap.controllers, used to communicate directly with the metadata quorum.
  --config CONFIG       Path to a property file containing a Kafka configuration
```

Note that:

- The `--snapshot` argument has been replaced by a `--directory` argument that reads the whole directory, not just a snapshot file
- There is no need for a `--cluster-id` flag, since we will query the controller for its cluster ID prior to creating the Raft client.
- There is now a `--config` argument which can be used to pass a configuration file.

Since `kafka-metadata-shell.sh` is at an "evolving" level of interface stability, these changes should be OK to make without a deprecation period.

New Errors

There will be a new `MISMATCHED_ENDPOINT_TYPE` error.

```
MISMIATCHED_ENDPOINT_TYPE([next], "The request was sent to an endpoint of the wrong type.",
MismatchedEndpointTypeException::new),
```

There will also be a new `UNSUPPORTED_ENDPOINT_TYPE` error.

```
UNSUPPORTED_ENDPOINT_TYPE([next], "This endpoint type is not supported yet.", UnsupportedEndpointTypeException::
new),
```

There will also be a new `UNKNOWN_CONTROLLER_ID` error.

```
UNKNOWN_CONTROLLER_ID([next], "This controller ID is not known.", UnknownControllerIdException::new),
```

DescribeCluster Changes

When `bootstrap.controller` is set, the AdminClient will use `DescribeClusterRequest` rather than `MetadataRequest` to obtain the cluster toplolgy.

We will add a new `EndpointType` field to `DescribeClusterRequest` . It will be set to 2 (controllers) when `bootstrap.controller` is in use. Otherwise, it will be set to 1 (brokers).

If the provided endpoint type does not match the actual endpoint type, we will return the `MISMATCHED_ENDPOINT_TYPE` error. So, for example, sending a request to a broker with an `EndpointType` of 2 (controller) will result in this error.

```
diff --git a/clients/src/main/resources/common/message/DescribeClusterRequest.json b/clients/src/main/resources
/common/message/DescribeClusterRequest.json
index 192e4d87d44..d1a4f432533 100644
--- a/clients/src/main/resources/common/message/DescribeClusterRequest.json
+++ b/clients/src/main/resources/common/message/DescribeClusterRequest.json
@@ -18,10 +18,12 @@
   "type": "request",
   "listeners": ["zkBroker", "broker"],
   "name": "DescribeClusterRequest",
-  "validVersions": "0",
+  "validVersions": "0-1",
   "flexibleVersions": "0+",
   "fields": [
     { "name": "IncludeClusterAuthorizedOperations", "type": "bool", "versions": "0+",
-      "about": "Whether to include cluster authorized operations." }
+      "about": "Whether to include cluster authorized operations." },
+    { "name": "EndpointType", "type": "int8", "versions": "1+", "default": "1",
+      "about": "The endpoint type to describe. 1=brokers, 2=controllers." }
   ]
 }
```

There will be a corresponding new version of `DescribeClusterResponse` .

```
diff --git a/clients/src/main/resources/common/message/DescribeClusterResponse.json b/clients/src/main/resources
/common/message/DescribeClusterResponse.json
index 1cd26c3d3c1..d774137d080 100644
--- a/clients/src/main/resources/common/message/DescribeClusterResponse.json
+++ b/clients/src/main/resources/common/message/DescribeClusterResponse.json
@@ -17,7 +17,7 @@
   "apiKey": 60,
   "type": "response",
   "name": "DescribeClusterResponse",
-  "validVersions": "0",
+  "validVersions": "0-1",
   "flexibleVersions": "0+",
   "fields": [
     { "name": "ThrottleTimeMs", "type": "int32", "versions": "0+",
@@ -26,20 +26,22 @@
       "about": "The top-level error code, or 0 if there was no error" },
     { "name": "ErrorMessage", "type": "string", "versions": "0+", "nullableVersions": "0+", "default": "null",
       "about": "The top-level error message, or null if there was no error." },
+    { "name": "EndpointType", "type": "int8", "versions": "1+", "default": "1",
+      "about": "The endpoint type. 1=brokers, 2=controllers." },
```

The EndpointType field will still be populated when there is an error, such as a `MISMATCHED_ENDPOINT_TYPE` error.

When `EndpointType` is "controllers", the ControllerId field will be set to the ID of the current active kcontroller, or -1 if there is no current active kcontroller.

If the MetadataVersion is too old to support controller registrations, and EndpointType was passed as "controllers," the controller will return `UNSUPPORTED_ENDPOINT_TYPE` . This reflects the fact that it doesn't have metadata about the controller endpoints in these older MetadataVersions.

ControllerRegistrationRequest / Response

There will be a new ControllerRegistrationRequest. All controllers will send this to the active controller.

```
{
  "apiKey": ...,
  "type": "request",
  "name": "ControllerRegistrationRequest",
  "validVersions": "0",
  "flexibleVersions": "0+",
  "fields": [
    { "name": "ControllerId", "type": "int32", "versions": "0+",
      "about": "The ID of the controller to register." },
    { "name": "ActiveControllerEpoch", "type": "int32", "versions": "0+",
      "about": "The epoch of the current active controller." },
    { "name": "IncarnationId", "type": "uuid", "versions": "0+",
      "about": "The controller incarnation ID, which is unique to each process run." },
    { "name": "ZkMigrationReady", "type": "bool", "versions": "0+",
      "about": "Set if the required configurations for ZK migration are present." },
    { "name": "Listeners", "type": "[]Listener",
      "about": "The listeners of this controller", "versions": "0+", "fields": [
      { "name": "Name", "type": "string", "versions": "0+", "mapKey": true,
        "about": "The name of the endpoint." },
      { "name": "Host", "type": "string", "versions": "0+",
        "about": "The hostname." },
      { "name": "Port", "type": "uint16", "versions": "0+",
        "about": "The port." },
      { "name": "SecurityProtocol", "type": "int16", "versions": "0+",
        "about": "The security protocol." }
    ]
    },
    { "name": "Features", "type": "[]Feature",
      "about": "The features on this controller", "versions": "0+", "fields": [
      { "name": "Name", "type": "string", "versions": "0+", "mapKey": true,
        "about": "The feature name." },
      { "name": "MinSupportedVersion", "type": "int16", "versions": "0+",
        "about": "The minimum supported feature level." },
      { "name": "MaxSupportedVersion", "type": "int16", "versions": "0+",
        "about": "The maximum supported feature level." }
    ]
    }
  ]
}
```

and a corresponding ControllerRegistrationResponse:

```
{
  "apiKey": ...,
  "type": "response",
  "name": "ControllerRegistrationResponse",
  "validVersions": "0",
  "flexibleVersions": "0+",
  "fields": [
    { "name": "ErrorCode", "type": "int16", "versions": "0+",
      "about": "The response error code." },
    { "name": "ErrorMessage", "type": "string", "nullableVersions": "0+", "versions": "0+",
      "about": "The response error message, or null if there was no error." }
  ]
}
```

The active controller will persist all registrations that are sent in with the correct permissions (CLUSTERACTION on CLUSTER). If the controller is not active, we'll send back a NOT_CONTROLLER error.

The active controller may also return STALE_CONTROLLER_EPOCH if the wrong epoch was passed.

ApiVersionsResponse

The ZkMigrationReady field in ApiVersionsResponse is now deprecated, and won't be filled out.

```
diff --git a/clients/src/main/resources/common/message/ApiVersionsResponse.json b/clients/src/main/resources
/common/message/ApiVersionsResponse.json
index 9fda953e10e..eb449f07c54 100644
--- a/clients/src/main/resources/common/message/ApiVersionsResponse.json
+++ b/clients/src/main/resources/common/message/ApiVersionsResponse.json
@@ -70,8 +70,6 @@
           "about": "The cluster-wide finalized min version level for the feature."}
       ]
     },
-    { "name":  "ZkMigrationReady", "type": "bool", "versions": "3+", "taggedVersions": "3+",
-      "tag": 3, "ignorable": true, "default": "false",
-      "about": "Set by a KRaft controller if the required configurations for ZK migration are present" }
+    { "name":  "ZkMigrationReady", "type": "deprecated", "versions": "3+", "taggedVersions": "3+", "tag": 3 }
   ]
 }
```

RegisterControllerRecord

The data from the registration request will be written to a new `RegisterControllerRecord`.

```
{
  "apiKey": ...,
  "type": "metadata",
  "name": "RegisterControllerRecord",
  "validVersions": "0+",
  "flexibleVersions": "0+",
  "fields": [
    { "name": "ControllerId", "type": "int32", "versions": "0+",
      "about": "The controller id." },
    { "name": "IncarnationId", "type": "uuid", "versions": "0+",
      "about": "The incarnation ID of the controller process" },
    { "name": "ZkMigrationReady", "type": "boolean", "versions": "0+",
      "about": "Set if the required configurations for ZK migration are present." },
    { "name": "EndPoints", "type": "[]ControllerEndpoint", "versions": "0+",
      "about": "The endpoints that can be used to communicate with this controller.", "fields": [
        { "name": "Name", "type": "string", "versions": "0+", "mapKey": true,
          "about": "The name of the endpoint." },
        { "name": "Host", "type": "string", "versions": "0+",
          "about": "The hostname." },
        { "name": "Port", "type": "uint16", "versions": "0+",
          "about": "The port." },
        { "name": "SecurityProtocol", "type": "int16", "versions": "0+",
          "about": "The security protocol." }
    ]},
    { "name": "Features", "type": "[]ControllerFeature",
      "about": "The features on this controller", "versions": "0+", "fields": [
      { "name": "Name", "type": "string", "versions": "0+", "mapKey": true,
        "about": "The feature name." },
      { "name": "MinSupportedVersion", "type": "int16", "versions": "0+",
        "about": "The minimum supported feature level." },
      { "name": "MaxSupportedVersion", "type": "int16", "versions": "0+",
        "about": "The maximum supported feature level." }
    ]}
  ]
}
```

At the moment, there will be no incremental version of this record, since there is no equivalent to broker fencing and unfencing on the controller. There is also no epoch – the incarnation ID, chosen by the controller itself, is what we use.

AdminClient Changes

As described above, AdminClient will now support `bootstrap.controller` in addition to `bootstrap.server` . The following APIs will be supported:

API	Notes

alterConfigs	
createAcls	
deleteAcls	
describeCluster	
describeAcls	
describeConfigs	
describeClientQuotas	
alterClientQuotas	
incrementalAlterConfigs	This can be used to alter log4j settings on inactive controllers. That is the one case where we don't send the RPC to the active controller.
describeDelegationToken	
electLeaders	
alterPartitionReassignments	
listPartitionReassignments	
describeClientQuotas	
describeUserScramCredentials	
describeFeatures	
updateFeatures	
describeMetadataQuorum	
unregisterBroker	this affects brokers, not controllers

When an unsupported API is used with `bootstrap.controller` , `UnsupportedEndpointType` is returned as an error.

Controller Changes

Fix compatibility gates

Previously, the controller used ApiVersionsResponse messages obtained from the other controllers to determine:

- whether all controllers in the cluster supported a proposed change to the MetadataVersion
- During a ZK migration, whether all controllers in the cluster were ready to exit premigration

This method was flawed, because it relied on the idea that controllers were always connected to each other. In reality, after the Raft election concludes, the only communication that takes place is inactive controllers connecting to the active controller. The other connections are gradually closed, and we lose information about those other nodes.

With the introduction of controller registrations, the active controller will simply use this information instead. This will be more accurate than the old system.

New APIs

The following RPCs will now be supported on the controller:

API	Notes
DESCRIBE_CONFIGS	Even on inactive controllers, this can be used to alter the log4j settings dynamically.
DESCRIBE_CLUSTER	As described above, DESCRIBE_CLUSTER will be used by AdminClient for bootstrapping.
REGISTER_CONTROLLER	

Compatibility, Deprecation, and Migration Plan

Client Compatibility Matrix

	pre-KIP-919 broker	post-KIP-919 broker	pre-KIP-919 controller	post KIP-919 controller
bootstrap.server	success	success	Fails because METADATA request is not supported.	Fails because METADATA request is not supported.
bootstrap.controller	Fails because DESCRIBE_CLUSTER returns MISMATCHED_ENDPOINT	Fails because DESCRIBE_CLUSTER v1 is not supported.	Fails because DESCRIBE_CLUSTER v1 is not supported.	If the metadata version is too old to support controller registrations, UNSUPPORTED_ENDPOINT_TYPE. Otherwise, success.

Controller Registration MetadataVersion Gate

Controller registrations will be gated behind a new metadata version. The controllers will not attempt to register themselves if the current metadata version is too old.

Rejected Alternatives

bootstrap.controllers versus direct.to.controller configuration

Rather than having a `bootstrap.controllers` configuration, we could have a separate configuration like `direct.to.controller` and put the controller servers into `bootstrap.servers` . Similarly, we could reuse `--bootstrap-server` erather than adding `--bootstrap-controller`.

We decided to go with the scheme proposed above to make it clearer when a tool was going directly to the controller. This also makes it clearer which command-line tools have this capability and which do not.

For example, `kafka-console-consumer.sh` does not have the capability to go direct to the controller, since the controller does not handle produces. Therefore, it's intuitive that `kafka-console-consumer.sh` lacks the `--bootstrap-controller` flag.

Another issue is that in the future, we may want to support using the controllers as bootstrap servers for the brokers. The scheme above leaves the door open for this, whereas a scheme that reused existing configurations would not.

Extending MetadataRequest instead of using DescribeClusterRequest

Instead of extending DescribeClusterRequest, we could have extended MetadataRequest so that the AdminClient could send that to the controllers. However, MetadataRequest doesn't return a top-level error code. So we cannot cleanly handle the many compatibility scenarios described above. We would have no choice but to simply terminate the TCP connection, which would leave the user guessing what the problem was.

Integration with Raft

When we implement dynamic KRaft quorum reconfiguration, we will want to store information about Raft voters in the Raft log itself. This will allow the quorum to change and grow beyond the initial static configuration. So it's reasonable to ask if there is some overlap between the reconfiguration project and this one.

However, I think these two KIPs should be separate. The registration information QuorumController cares about is different than what Raft will care about. For example, Raft will not care about the currently active features, or whether ZK migration is ready.

Future Work

Bootstrap-to-broker

In the future, we might want to allow the controllers to be used as bootstrap servers for the brokers. This would be helpful, for example, in cases where a plugin running on the controller itself wanted to create a consumer or producer, without hard-coding broker addresses in the configuration.

This is a separate use-case from the direct-to-controller one, so probably needs different configuration.

One major problem is what broker endpoints to return. Perhaps we could always return the inter-broker endpoints in the response. However, it's unclear how the client should proceed when the controller has different security settings than the selected broker endpoints. This might require more complex client configuration.

Better API for returning topic data

MetadataRequest has many limitations.

- It unconditionally returns broker information, even when we don't care about that (such as in AdminClient)
- Its response does not incorporate a top-level error code, so there is no reasonable way to indicate failure
- All requested topics are returned in one big lump, which makes the garbage collector choke in larger clusters

In the future, we should have a better API for listing topic data, that does not have these problems. This would also give us a clean way to implement `Admin#listTopics` when using `bootstrap.controller` .