KIP-433: Block old clients on brokers

Status

Current state: *Under Discussion*

Discussion thread: discussion thread

JIRA: KAFKA-7975

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

Motivation

We still have some users using very old Kafka client libraries, such as Kafka 8 and Kafka 9. We need a mechanism to force users to upgrade their client libraries, and forbid outdated client libraries.

The old libraries are still supported by the latest brokers, but can cause problems when we starting using some new features. Some examples are:

1. Kafka 8 / 9 / 10 consumer hangs when the message contains message header (⚠ Unable to render Jira issues macro, execution error.)
2. LZ4 is not correctly handled in Kafka 8 and Kafka 9 (⚠ Unable to render Jira issues macro, execution error.)
3. Performance penalty of converting message format from V3 to V1 or V2 for the old consumers (KIP-31 - Move to relative offsets in compressed message sets)

In general, we think it's reasonable to give cluster administrator the capability to forbid old client library versions.

In this KIP, we propose to add a new broker configuration to specify the minimum client version.

Public Interfaces

A public interface is any change to the following:

- *Configuration*

  Add a new broker configuration:

Name	Description	Type	Default	Importance	Dynamic Update Mode
min.api.version	Minimum API versions specified in a list of coma-separated <api-name>:<version> pairs. Client requests older than the specified versions will be rejected by the broker. For the APIs that are not specified in this list, the min version is 0. For example, "Fetch: 5, Produce: 3" 1) rejects fetch requests from Kafka 0.10 or earlier clients, and produce requests from Kafka 0.11 or earlier clients, but allow any client version for the other APIs. Valid API names and versions can be found in Kafka protocol guide	string	""	low	cluster-wide

- *Binary log format*

  No
- *The network protocol and api behavior*

  Client requests older than the specified versions will be rejected by the broker.
- Any class in the public packages under clientsConfiguration, especially client configuration

  No
- *Monitoring*

  No

- *Command line tools and arguments*

  No

- *Anything else that will likely break existing users in some way when they upgrade*
  No

Proposed Changes

All the changes of this KIP are in Kafka broker:

1. Add min.api.version configuration in Kafka broker
2. Allow dynamic update of min.api.version configuration
3. Reject the API requests of older versions. The broker will return an UNSUPPORTED_VERSION error to the client, and log an INFO level log message.
4. In ApiVersionResponse, the MinVersion of each API is set to the current setting of min.api.version.

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*
  By default, all the existing client libraries are allowed. If the administrator set

Rejected Alternatives

Alternative 1 - Provide api versions to authorizer

Provide client API version to authorizer. So that, the user-defined authorizer can block clients based on api versions. This is more flexible. But there are 2 concerns: 1) Blocking old clients is not a security issue. It doesn't make much sense to add this feature to authorizer. 2) For most users, it's easier to set a broker configuration than write a user-defined authorizer.

Alternative 2 - Use Kafka release version rather than API version

Users are more familiar with Kafka release versions. So far, the API version is mostly a Kafka internal thing. It's more convenient for the users to use Kafka release versions than API versions.

However, we can't use Kafka release versions here, because Kafka clients do not report the release versions to brokers. Brokers can only see the clients' API versions. An API version doesn't map a single release version. For example, a user sets the min Kafka version of produce request to Kafka 1.1. She would expect the broker will reject Kafka 1.0 producers. However, both Kafka 1.1 and Kafka 1.0 are using api version 5. The broker can't distinguish the 2 versions. So, Kafka 1.0 producers are still allowed. This can be version confusing to the users.

Moreover, there are 3rd party implementations of Kafka clients (e.g. go client, c++ client). Those client libraries have their own release versions, and are not related with any specific Java / Scala client version.