# KIP-899: Allow producer and consumer clients to rebootstrap



## Status

**Current state**: *Under Discussion*

**Discussion thread**: *here*

**JIRA**: KAFKA-8206

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

## Motivation

A Kafka client performs bootstrapping when it's initialized, i.e. it connects to a server from `bootstrap.servers` and fetches the cluster metadata, including the addresses of online brokers. This list of brokers from the fetched metadata is used for the real work. The client periodically updates the metadata during the network client's polls so even if the set of brokers change over time, this generally works well. However, brokers already known to the client are used for fetching the subsequent metadata updated instead of the bootstrap servers.

The problem happens when two conditions are met:

1. The client's polls are infrequent.
2. The current set of online brokers after some update is completely non-overlapping with the previous set of online brokers.

In this case after the cluster update the client will not be able to connect to any broker known to it before the update and will fail.

Both conditions are not impossible and can be found in real life setups. Clients may stay idle for a long time if they serve some very specific purpose. An example of such a client is `restoreConsumer` in Kafka Streams which is idle for most of the time.

The set of online brokers could realistically drift from the set known to a long-idle client. For example, a cluster can have an initial set of brokers `broker1.kafka.company.net,broker2.kafka.company.net,broker3.kafka.company.net`. They are grouped together under `bootstrap.kafka.company.net` for bootstrapping. A client bootstraps and remembers broker1-broker3. However, after a cluster update, the old machines are shut down and a new set of brokers is started: `broker4.kafka.company.net,broker5.kafka.company.net,broker6.kafka.company.net` (`bootstrap.kafka.company.net` now resolves to these machines).

There is a number of tickets related to the issue in the Kafka JIRA, which indicates the issue does happen to Kafka users:

- Unable to render Jira issues macro, execution error.
- Unable to render Jira issues macro, execution error.
- Unable to render Jira issues macro, execution error.

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

# Proposed Changes

This KIP proposes to allow Kafka producer and consumer clients to repeat the bootstrap process when updating metadata if none of the known brokers are available. A broker is unavailable when the client doesn't have an established connection with it and cannot establish a connection (e.g. due to the reconnect backoff).

During the rebootstrap process, the client forgets the brokers it knows about and falls back on the bootstrap brokers (i.e. provided by `bootstrap.servers` provided by the client configuration) as if it had just been initialized.

The client will check the cluster ID returned by the broker during the rebootstrap process. If no cluster ID was known to the client (i.e. it was originally bootstrapped with an old broker version that doesn't support cluster IDs), any returned value will be considered valid. Otherwise, the client will fail if the returned cluster ID doesn't match the previously known one.

The admin client behaves differently and doesn't update metadata. Because of this, it is excluded from the scope of this KIP.

`reconnect.backoff.max.ms` can be configured so low that brokers that are truly unavailable will never be considered as such, i.e. always will be eligible for reconnect. This is a known limitation. Unfortunately, it's hard to find a good criteria when to ignore this and trigger rebootstrapping nevertheless. It was decided to keep this out of the scope of this KIP.

Since this changes the user-facing behavior, it's proposed to make this configurable (see *Public Interfaces*), defaulting to the current behavior.

# Public Interfaces

## Configuration Keys

| Key Name | Description | Valid Values | Default Value |
|---|---|---|---|
| metadata.recovery.strategy | Controls how the consumer or producer client recovers when none of the brokers known to it is available. If set to `none`, the client fails. If set to `rebootstrap`, the client repeats the bootstrap process using `bootstrap.servers`. `reconnect.backoff.max.ms` may be so low that it prevents identifying brokers as unavailable, in this case rebootstrapping won't happen. | `none`, `rebootstrap` | `none` |

# Compatibility, Deprecation, and Migration Plan

Migrating to the new version will have no impact on clients as the default configuration value keeps the old behavior.

The behavior will remain configurable for the foreseeable future.

No special migration process or tool is needed to migrate to the new version.

# Test Plan

The proposed change could be tested on the integration level. The KIP proposed two test cases, one for the producer and one for the consumer. In the tests, clients will bootstrap using `bootstrap.servers=broker1,broker2`, where only `broker1` is in the cluster. After that, `broker1` will be shut down and `broker2` will be brought up and the client will be made to communicate with the cluster. As `broker1`, previously known to it, is unavailable, it'll be forced to rebootstrap and connect to `broker2`.

Also, the cluster ID checking logic will be tested, preferably on the unit level.

# Rejected Alternatives

One alternative is to introduce a thread that periodically refreshes metadata in the background, independently of the network client explicit polls. This was considered more complex, introducing new failure modes, while bringing little more value compared to the proposed approach.