KIP-909: DNS Resolution Failure Should Not Fail the Clients

Status
Motivation
Proposed Changes
    - Public Facing Changes
    - Internal Changes

New Configuration
New Error
Compatibility, Deprecation, and Migration Plan
    - Compatibility
    - Deprecation
    - Migration

Case Study
    - KafkaConsumer
    - KafkaProducer
    - AdminClient
    - What should users do after the timeout expires?

Test Plan
Rejected Alternatives

Status

Current state: Accepted

Discussion thread: *here* *(Not happening yet)*

JIRA: *here*

Motivation

Instantiating a new client may result in a fatal failure if the bootstrap server cannot be resolved due to misconfiguration or transient network issues such as slow DNS. This is suboptimal because of the fact that it might take a long time for the address to become available at the DNS server, and users will need to continue to retry. Also, the ConfigException exception type does not accurately reflect the root cause of the problem, which makes it hard to handle this failure case. We think it is reasonable to allow users to have a grace period to retry if the address cannot be resolved immediately. Also, poisoning the clients during the construction can be obstructive; I think it is better to fail the client on its first attempt to connect to the network.

Proposed Changes

This KIP proposes moving bootstrapping logic from the constructor to the NetworkClient poll for two purposes,

1. not failing the client upon instantiation. In many cases, this behavior also kills the app, which might not be desirable.

2. piggybacking onto the client poll is a more natural way to retry.

We propose to add a new configuration option for timing out the bootstrapping process, a new exception type for handling bootstrap-related issues, and additional logging to aid in diagnosing bootstrapping failures.

Public Facing Changes

- **Timeout Configuration:** bootstrap.resolve.timeout.ms
- **Exception**: BootstrapResolutionException *extends KafkaException*
- **Logging:** log.WARN("Unable to bootstrap after {} ms.", elapsedMs)

Internal Changes

- **Client Constructor**: The constructor will only parse the bootstrap configuration.
- **NetworkClient:**
    - Bootstrapping will now occur in the poll method before attempting to update the metadata. This includes resolving the addresses and bootstrapping the metadata.
    - An error message will be logged in the event of a failed bootstrap process.
    - If the timeout exceeds, a non-retriable *BootstrapResolutionException* will be thrown.
- Consumer, Producer, and Admin Clients: The bootstrap code will be changed.

New Configuration

bootstrap.resolve.timeout.ms

The proposed configuration specifies the maximum amount of time clients can spend trying to resolve for the bootstrap server address. If the resolution cannot be completed within this timeframe, a BootstrapResolutionException will be thrown.

Type:	long
Default:	120000 (i.e. 2 minutes)
Valid Values:	0 - LONG_MAX
Importance:	high

New Error

Name: *BootstrapResolutionException* extends KafkaException

Message: "Couldn't resolve server {} from {} as DNS resolution failed for {}"

Type: Non-retriable.

Compatibility, Deprecation, and Migration Plan

Compatibility

- Failed DNS resolution throws BootstraResolutionException: Users are expected to catch the error or the client will be poisioned
- Users who tried to catch ConfigException for DNS resolution error will no longer need this logic.

Deprecation

- There's no deprecation plan

Migration

- There's no migration plan

Case Study

In this section, I outlined how clients can react to bootstrap failures. In particular, I want to cover two common cases:

1. Misconfiguration or non-transient issues with the network
2. Transient network issues, e.g., slow DNS resolution.

KafkaConsumer

Case 1: Non-transient case

When the bootstrap timeout expires, the client will throw a BootstrapResolutionException.

Case 2: Transient Network Issue

consumer poll won't return any record until the client has been bootstrapped. If the issue cannot be resolved within the bootstrap timeout, a BootstrapConnectionException will be thrown.

KafkaProducer

Case 1: Non-transient case

The BootstrapResolutionException will be thrown in send() and partitionsFor() when the bootstrap timeout expires. If the max.block.ms elapsed before the timeout expires, a TimeoutException will be thrown instead.

Case 2: Transient Network Issue

The *send()* and *partitionsFor()* methods will be blocked on bootstrap until either the *max.block.ms* or the bootstrap timeout elapses.

AdminClient

Case 1: Non-transient case

The API call results will either timeout if the request times out first or be completed exceptionally with a BootstrapResolutionException.

Case 2: Transient Network Issue

The user won't be able to get the results back until the address is resolved. Meanwhile, the API calls can expire.

What should users do after the timeout expires?

The exception is meant to be fatal, so the user should check their network setup, configuration, or adjust the timeout.

The user can continue to retry, but this exception is meant to alert user to take action upon failing to bootstrap.

Test Plan

1. NetworkClient
    a. Test DNS resolution upon its initial poll
    b. Test if the right exception type is thrown
2. Existing clients (Consumer, Producer, AdminClient)
    a. Test successful bootstrapping upon retrying

Rejected Alternatives

We've discussed many alternatives. Eventually, we asked ourselves what's the goal of this KIP, i.e., giving people a chance to retry on DNS resolution without poisoning the client. Which came down to two resolutions: 1. giving people a configurable timeout, and 2. adding a fatal error to alert the user.

Here are the rejected alternatives:

1. Maintain the current code behavior and add a retry loop with a timeout.
    a. **Pros**: Same logic, less code change.
    b. **Cons**: Do users want to be blocked on instantiating the client? I don't like this idea.
2. Throw DNS resolution upon failing but no retry
    a. **Pros**: No additional config is needed
    b. **Cons**: This is a behavioral change, and the application owner might need to rewrite the exception handling, i.e. catching the DNS failure logic.
3. No retry. The network client will continue to retry until it is interrupted.
    a. **Pros**: No compatibility break. No additional exception handling logic, the network client will just log the error and continue to retry upon the next poll
    b. **Cons**: I think we should have some failure mechanism to notify users.
4. Making BootstrapResolutionException retriable
    a. **Pros**: For the transient case, we might not even need a timeout, people are expected to retry on catching this exception
    b. **Cons**: Then we reply on alerting mechanism to alert users the issue. If it is indeed a configuration issue, then it is harder to discover
5. Combine DNS resolution and connection into a single timeout
    a. **Pros**: Using a single timer to account for the connection time.
    b. **Cons**: Should we make connection retry fatal after the timeout? Maybe not.
6. 5min default timeout
    a. We've decided to reduce it to 2min to stay coherent to the delivery.timeout.ms
    b. 5min can be too long