KIP-459: Improve KafkaStreams#close

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: *KAFKA-7996*

Released: (Not decided yet)

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

Motivation

As of 2.2.0, `KafkaStream#close` works like the following:

1. Launch a daemon thread that closes `StreamThreads`, `GlobalStreamThread`, and `AdminClient`. `StreamThread` and `GlobalStreamThread` have their own `Producers` and `Consumers`. All close methods are called without any timeout.
2. Wait for the daemon thread to complete its job and change `KafkaStreams` state into `NOT_RUNNING`. (see `KafkaStreams#waitOnState`)

However, for `Producer` and `AdminClient`'s default close timeout is `Duration.ofMillis(Long.MAX_VALUE)`, Kafka streams application may take a long time or even hang up for closing internal `Producers` or `AdminClient`.

To resolve this problem, this KIP proposes to provide a close timeout parameter for closing internal clients.

Public Interfaces

There are 3 alternatives for providing close timeout parameter.

Type A. fix the close timeout as a constant

This approach makes any change on the public interface; it just uses a constant like `KafkaStreams#DEFAULT_CLOSE_TIMEOUT` as a client close timeout.

Pros

Easy to implement.

Cons

Users have any way to configure close timeout. Moreover, some users may want current close timeout, i.e., `Duration.ofMillis(Long.MAX_VALUE)`.

Type B. Provide a new configuration option

Adding a new configuration option, `close.wait.ms`, and allow users to configure the close timeout. (default: 500ms)

Pros

Easy to configure.

Cons

Adds a new configuration option, which is already so many.

Type C. Extend KafkaStreams constructor

By adding a new optional parameter (`closeTimeout`) to the constructor of `KafkaStreams`, allow users to configure the close timeout if required (default: 500ms):

```
public KafkaStreams(final Topology topology, final Properties props, final Duration closeWaitTime);
public KafkaStreams(final Topology topology, final Properties props, final KafkaClientSupplier clientSupplier,
final Duration closeWaitTime);
public KafkaStreams(final Topology topology, final Properties props, final Time time, final Duration
closeWaitTime);
public KafkaStreams(final Topology topology, final Properties props, final KafkaClientSupplier clientSupplier,
final Time time, final Duration closeWaitTime);
```

Pros

Allows the users to configure the close timeout, without adding new configuration option.

Cons

This approach adds 4 overload constructors; Since there are already 2 required parameters and 2 optional parameters, there are 4 constructors and now becomes 8 constructors - it is too many.

This problem can be mitigated by deprecating the overloaded constructors and providing `KafkaStreams.Builder` instead. However, this issue is above the scope of this KIP.

Proposed Changes

(under discussion)

Compatibility, Deprecation, and Migration Plan

None.

Test Plan

By passing the existing unit tests.

Rejected Alternatives

Change the default close timeout for [Producer, AdminClient]#close

This approach has an advantage in fixing the default close timeout inconsistency between `Consumer` and `Producer`, `AdminClient`. However, this inconsistency was intended; The current `KafkaConsumer#close`'s default timeout, 30 seconds, was introduced in KIP-102 (0.10.2.0). To summarize, there are two differences between `Consumer` and `Producer`;

1. `Consumers` don't have large requests.
2. `Consumer#close` is affected by consumer coordinator, whose close operation is affected by `request.timeout.ms`.

By the above reasons, Consumer's default timeout was set a little bit different. So, we don't need to align the parity.

Moreover, this approach does not allow the users to configure a close timeout, and breaks backward compatibility. So rejected.