KIP-1015: Limit number of ssl connections in brokers

Status

Current state: *Under Discussion*

Discussion thread: https://lists.apache.org/thread/hox43cslkps2mqtqo1yy441g1yo4nrjd

JIRA: https://issues.apache.org/jira/projects/KAFKA/issues/KAFKA-16081

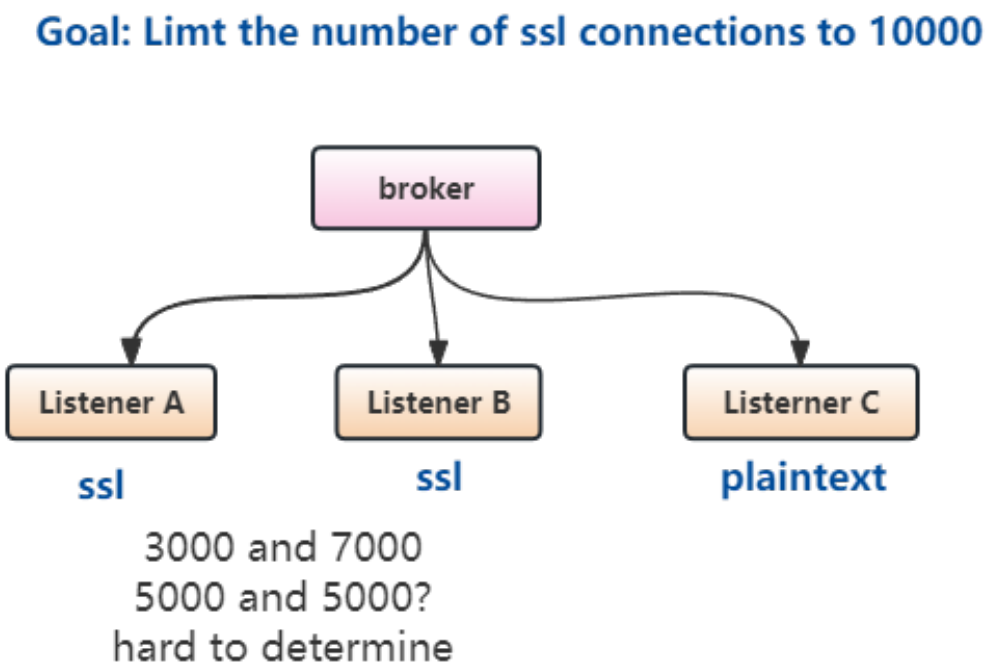
Motivation

For Kafka, an SSL connection occupies approximately 100KB of memory, while a plaintext connection occupies around 250 bytes, resulting in a memory footprint ratio of approximately 400:1. Therefore, there should be a limitation for SSL connections in broker wide to prevent potential OOM situations caused by an excessive number of SSL connections.

Currently, we have `max.connections` configuration at both broker and listener levels, allowing us to limit the maximum number of active connections on each listener and the overall broker.

However, the current implementation presents a challenge in how to **effectively controlling the number of SSL connections**. If we have a broker supporting 10,000 SSL connections with two SSL listeners, how should we determine the appropriate connection limit for each listener? Should we set the connection count to 5,000 for each listener, or allocate 3,000 connections for listener A and 7,000 connections for listener B?

In other words, we **cannot precisely control the total SSL connections count**, especially if one listener is heavily used while the other has fewer connections.



The new configuration could work together with max.connections to control ssl and non-ssl connections. For example, set max.connections to 10000, max.ssl.connections to 5000, which means non-ssl connections is litmit to 5000.

Public Interfaces

No new interfaces or will be added.

Two new metrics will be added to track the total numer of ssl connections and all connections , and it will be a yammer `Meter`.

- `kafka.network:type=ConnectionQuotas,name=SslConnectionsCount`
- `kafka.network:type=ConnectionQuotas,name=ConnectionsCount`

A new broker configuration `max.ssl.connections` option will be added to limit the total number of ssl connections.

This is in addition to the existing `max.connections.per.ip and max.connections` config that will continue to limit the total number of connections and from each host ip address. When this limitation is reached, new SSL connections will be restricted while non-SSL connections will not be affected until one or more SSL connections are disconnected. This will be a dynamic broker-wide config that can be updated without restarting the broker.

Config option: Name: `max.ssl.connections` Type: `Int` Default value: `Int.MaxValue`

Proposed Changes

> `Acceptor` will stop accepting new connections when the broker's `max.connections` limit is reached. New connections will be accepted as soon as a connection is closed by any of the `Processors`. `Acceptor` will also stop accepting new connections when its listener's `listener.name.{listener}.max.connections` limit is reached. New connections will be accepted as soon as a connection is closed by any of the `Processors` of that listener. Inter-broker connections will be protected in multi-listener brokers by closing client connections to accommodate inter-broker connections. Any time spent by `Acceptor` waiting for connections to close will also be included in the new `AcceptorBlockedPercent` metric. The existing `max.connections.per.ip` config will be applied without any changes. Connections dropped due to hitting the per-ip limit will not appear in the `AcceptorBlockedPercent` metric since these connections are accepted and then dropped.

The behavior when the SSL connection limit is reached will ***be similar to reaching the maximum connection limit*** *at the broker level. It is important to note that this configuration will not affect non-SSL listeners.*

Based on the current implementation, we will use the LRU algorithm to close connections for the listener in the following two scenarios:

1. Total number of connections exceeds the broker's connection limit.
2. The number of connections exceeds the listener's connection limit.

I propose adding an additional condition: if the number of SSL connections **exceeds the SSL connection limit** and the current listener is **using SSL proto col**, the connections may be closed.

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*

No externally visible interface changes are proposed in this KIP. During normal operations, this is unlikely to result in any impact. When a large number of ssl connections are made to the broker at the same time, connections may be established slower, and the least recently used connections may ***be disconnected****.*

Test Plan

Will be tested in **org.apache.kafka.network.ConnectionQuotasTest.scala** in the following cases:

- This configuration could be updated dynamically and correctly
- Hiting the limit of max.ssl.connections, new ssl connections will be blocked, while plaintext listener will not be affected.
- Removing one sasl connection should make the waiting connection to succeed
- sslConnectionCounts variables updated correctly

Rejected Alternatives

It might be argued that the existing listener-specific limit already provides satisfactory ability to limit SSL listener.

As mentioned above

1. Unable to limit ssl connections precisely.
2. The protocol of the listener may change dynamically, and the limit of the listener also needs to be modified.