KIP-687: Automatic Reloading of Security Store

Status

Current state: *Approved*

Discussion thread: https://lists.apache.org/thread.html/ra490809bd850159fe4f9c4668c2ecf84cf9f4a28d50a4a461d212e72%40%3Cdev.kafka.apache.org%3E

JIRA: https://issues.apache.org/jira/browse/KAFKA-10345

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

Motivation

In the admin client (Incremental)AlterConfig API, we implicitly support a feature to reload key/trust store by sending an (Incremental)AlterConfig request directly to the target broker with the **exact same store path**. This special logic will no longer work once all the AlterConfig requests get forwarded to the active controller, and the target broker will not do a security store reload since the config change ZK notification contains the same key/trust store path as its local copy. In addition, the key/trust store reloading is a completely separate feature from AlterConfigs. It does not change any persistent broker config values in either ZK or metadata quorum. Instead of using client RPCs to directly trigger updates, we propose to use a file watcher on the security store file instead, to listen to any file content change and reload the config as necessary in the post-KIP-500 world. To protect the worst case where file-watch does not trigger properly, a time-based reloading mechanism will also be added.

Public Interfaces

We would enforce a file-watch based reloading mechanism to the following configs:

- **ssl.keystore.location**
- **ssl.truststore.location**

A broker side metrics will be added to track times of security store reloads for success and failure as:

`MBean:kafka.server:type=BrokerConfigMetrics,name={Success|Failed}SecurityStoreReloadPerSec,store_type=([-.\w]+)`

where the current supported store types are: key_store|trust_store.

Additionally, two dynamic broker config called `ssl.keystore.location.refresh.interval.ms` and `ssl.truststore.location.refresh.interval.ms` will be added to control the time based guarantee for an automatic reloading in case of a missed file-watch. To disable periodical reloading, users could set those configs to max long.

SecurityConfig.java

```
public static final String SSL_KEYSTORE_LOCATION_REFRESH_INTERVAL_MS_CONFIG = "ssl.keystore.location.refresh.
interval.ms";
public static final String SSL_KEYSTORE_LOCATION_REFRESH_INTERVAL_MS_DOC = "The refresh interval for in-place
ssl keystore updates. In general, " +
   "the update should trigger immediately when user modifies the security file path through file watch service,
while " +
   "this configuration is defining a time based guarantee of store reloading in worst case";

public static final String SSL_TRUSTSTORE_LOCATION_REFRESH_INTERVAL_MS_CONFIG = "ssl.truststore.location.
refresh.interval.ms";
public static final String SSL_TRUSTSTORE_LOCATION_REFRESH_INTERVAL_MS_DOC = "The refresh interval for in-place
ssl truststore updates. In general, " +
   "the update should trigger immediately when user modifies the security file path through file watch service,
while " +
   "this configuration is defining a time based guarantee of store reloading in worst case";
```

The default values will be set to 5 minutes and could be changed through AlterConfig API.

Proposed Changes

Once the editing for the security store file was done, user should expect the above metric to reflect the reloading of security store instantly. We would also add INFO level message to indicate a reloading event has been triggered, with the reload results. If the result is a failure, we would increment the failed metrics and log the reason in ERROR. When the reloading fails, previous store should still be effective.

The broker will also periodically check whether security files have been modified since last checking time and decide to reload or not, so that when the file watch does not work for unknown reason, user could wait until the time based reloading mechanism kicks in. If the time based reloading is not working either, user could still try to change the store path in an explicit AlterConfig call in the worst case.

This mechanism will be applied only to the brokers. Client side security store reloading mechanism is not affected.

Compatibility, Deprecation, and Migration Plan

The feature support for same name store reloading will be working in the new brokers automatically. We would try to communicate the deprecation of the old AlterConfig based path in the broker WARN log to let user read the metric instead to evaluate whether the security store is reloaded successfully.

Right now we don't plan to support disabling the auto reloading feature, since it will become the only option to do in-place security store update in the future.

We are also changing the audit log/authorization model for dynamic in-place updates of SSL stores. At the moment, only a user with powerful Cluster:Alter permissions can dynamically
update SSL stores on brokers. The KIP removes this restriction and relies purely on file system permissions for file-based stores.

Rejected Alternatives

We have proposed to use a new RPC which gets sent to the target broker directly. The solution involves a couple of compatibility related concerns, such as the following matrix suggests:

Broker Version	Client Version	Admin API	Expected Behavior
old	new	alterConfig	work with a warning log
old	new	storeReload	work with the underlying request translating to AlterConfigRequest
new	new	alterConfig	Not work, must use storeReload API
new	new	storeReload	work
new	old	alterConfig	work, with active controller sending a storeReload request to the target broker
old	old	alterConfig	N/A

We have also attempted a hacky solution by augmenting the security store path from single slash to double slash, like "/foo/bar" to "//foo//bar", in the hope of triggering an equivalent path reloading. This approach does solve the existing problem, but it is error-prone and does not meet a good maintainability standard. For example, it does not take care of the back slash path in Windows platform, also could potentially lead to path explosion if not implemented correctly as "///////foo///////bar". We certainly don't want to maintain such a hacky feature or extra effort to make it understandable to developers.