KIP-1011: Use incrementalAlterConfigs when updating broker configs by kafka-configs.sh

Status

Current state: *Accepted*

Discussion thread: https://lists.apache.org/thread/xd28mgqy75stgsvp6qybzpljzflkqcsy

JIRA: https://issues.apache.org/jira/browse/KAFKA-16181

Released:

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

Motivation

We have 2 methods in `AdminClient` for updating config, `alterConfigs` and `incrementalAlterConfigs`, the former has many restrictions and has been deprecated from 2.3.0. However, It's still used in `ConfigCommand` to update broker config and is resulting in a bug

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

, so I'm inclined to move it to `incrementalAlterConfigs` and fallback to use alterConfigs if the server is under 2.3.0. There are some other reasons for this change:

1. `alterConfigs` has been deprecated and will be removed in a future release;
2. we are using `incrementalAlterConfigs` to change user/topic/client-metrics configs, it would be benefit to unify broker configs;
3. `incrementalAlterConfigs` is more convenient especially for updating configs of list data type, such as "leader.replication.throttled.replicas", though we can't subtract or append configs using kafka-configs.sh, we can make way for the future for appending/subtracting list properties by use `incrementalAlterConfigs.`
4. We are forced to pass all sensitive configs to update broker configs when using `alterConfigs` with the current cli tool because sensitive config values are never returned to the client, this result in KAFKA-13788, it must be resolved.

Note that I'm only changing the way we updating broker configs, user/topic/client-metrics configs are already being updated using `incrementalAlterConfigs`

Public Interfaces

kafka-configs.sh

we are changing the semantics of kafka-configs.sh without changing any command arguments.

1. Existing sensitive broker properties no longer have to be explicitly specified on the command line if they're not being changed
2. A small race condition is fixed where the broker config is updated by a separate operation in between when the CLI reads the existing broker config and writes the new broker config
3. Usage of a new broker API that has been supported since version 2.3.0,

Proposed Changes

When updating broker config, instead of using Admin.alterConfigs, we will use Admin.incrementalAlterConfigs and fallback to use `alterConfigs` automatically if `incrementalAlterConfigs` is not supported, we are doing this heuristically instead of manually.

1. We only do this when (--alter) and (--broker <broker-id>)/(--entity-types brokers) are specified, or it will be ignored, for example when updating topic configs.
2. We only do this when (--bootstrap-server)/(bootstrap-controller) is specified, we are leaving zookeeper case unchanged.
3. use Admin.`incrementalAlterConfigs` firstly, which will fail if the broker is before 2.3.0, and we will retry with Admin.alterConfigs()
4. AdminClient.alterConfigs is deprecated and will be remove In the future together, and AdminClient.`incrementalAlterConfigs` will be the only choice then.

here is an example of changing the broker configs twice:

1. set log.cleaner.threads=2
2. set background.threads=1

in case 1 with old version client, here are what happened:

1. Use kafka-configs.sh to set log.cleaner.threads=2, the client will fetch all broker configs and get a empty properties {}, merge it with the delta and get {log.cleaner.threads=2}, send it to server using AdminClient.alterConfigs(), the server will persist it to metadata storage.
2. Use kafka-configs.sh to set background.threads=1, the client will fetch all broker configs and get {log.cleaner.threads=2}, merge it with the delta and get {log.cleaner.threads=2, background.threads=1 }, send it to server using AdminClient.alterConfigs(), the server will persist it to metadata storage.

in case 2 with new version client, or when we removed the deprecated AdminClient.alterConfigs, here are what happened:

1. Use kafka-configs.sh to set log.cleaner.threads=2, the client will send it to server using AdminClient.incrementalAlterConfigs(), the broker will merge the old snapshot({}) with delta and save the new snapshot({log.cleaner.threads=2}) to metadata storage.
2. Use kafka-configs.sh to set background.threads=1, the client will send it to server using AdminClient.incrementalAlterConfigs(), the broker will merge the old snapshot({log.cleaner.threads=2}) with delta and got the new snapshot ({log.cleaner.threads=2, background.threads=1 }) to metadata storage.

in case 3 with new version client and old version server, we will try firstly as case 2, and fail with UnsupportedVersionException, and retry as case 3.

The result is expected to be the same for both way, except that we can avoid some issues for AdminClient.alterConfigs().

Compatibility, Deprecation, and Migration Plan

There is no backward compatibility problems brought in because we are changing it in a compatible way.

We should pay attention to KAFKA-10140 in which we have a problem updating jmx configs using `incrementalAlterConfigs` but we can still update jmx configs using `alterConfigs`, however, the problem only appear when append/subtract list type config which is not supported by kafka-configs.sh, so it will not influence this KIP, and we leave it in another one when we try to support append/subtract in kafka-configs.sh.

Test Plan

new client tool with older server can be tested locally and using test cases.

Documentation Plan

There are some api changes and we should document about kafka-configs.sh change list.

Rejected Alternatives

1. Adding new option --enable-incremental to give this privilege to users, since we can do it heuristically without and side effect, and avoid adding a deprecated argument which should be removed in the future.
2. we make incrementalAlterConfigs the default way and add a "--disable-incremental" flag for old servers before Kafka 3.X, this is the same with solution 1.
3. We just forward all invocations of alterConfigs to incrementalAlterConfigs. Similar to first one, we need to unify the semantics, it's not recommend since alterConfigs is deprecated and we just leave it as it was.