KIP-21 - Dynamic Configuration

Status

Current state: *Accepted*

Discussion thread: here

JIRA: KAFKA-2204

Motivation

In Kafka, there is no general mechanism to change entity configuration without doing a rolling restart of the entire cluster. Currently, only topic configs can be changed dynamically. This proposal attempts to build a unified mechanism for modeling configuration across various entities within Kafka i.e. topics, clients etc.

Public Interfaces

- We will add a new tool called ConfigChangeCommand that can manage all config changes in ZK. New methods will be added in AdminUtils to change configuration. This will be similar to the TopicCommand tool already present.
- AlterConfig and DescribeConfig APIs will be added (after KIP-4 is complete) to alter and view configs
- There will be new zookeeper paths under "config" but they are not considered to be public interfaces.

Proposed Changes

This proposal will remove the TopicConfigManager and introduce a "DynamicConfigManager" to handle config changes for all entities via Zookeeper. Initially, we will have only 2 entities Topic and Clients.

ZNode Structure

```
There will be 3 paths within config
/config/clients/<client_id>
/config/topics/<topic_name>
/config/changes/

Internally, the znodes are comma-separated key-value pairs where key represents the configuration property to change.
{"version": x, "config" : {X1 : Y1, X2 : Y2..}}
```

Upon startup, all brokers will load all the configs from zookeeper. In order to receive notifications, it's undesirable to watch every path under the root config directory. We can model change notifications the same way as they are currently handled for topic based configs. Here is the workflow for changing or adding a config of any type:

- Create a znode (or modify existing) under the required path with the configs that you want to update. Example, if you want to change quotas for producer ***"Adi"***, add a path under ***/config/clients/Adi*** as shown below.
- Create a sequential znode under "config/changes/config_change_XX". This will send a notification to all the watchers. The data within the change node should indicate what has changed i.e. topic config + topic name, client config + clientId.
- The brokers process all the configs for the entity that has changed.

Config Precedence

Configs in Zookeeper take precedence over any configuration read from the properties file. There is currently no plan to move away from file based configuration for service configs.

Applying Client Config Changes

The first usecase for client configs is to process quota changes per-client. This class will process all notifications and change clientId quotas accordingly. Some changes to the metrics package are also required to allow modification of quotas. Producer and consumer quotas are distinguished by having different properties since it's possible for a producer and consumer to have the same client id.

The client znode will look like this:

```
{"version" : 1, "config" : {"producer_byte_rate" : 1000, "consumer_byte_rate" : 2000}}
```

Config Change Notification

Currently, the change notification znode only contains the topic name. We need to add more information to distinguish whether this config change is for a topic, client or other config. Version numbering should also be added.

```
The notification data can be:
{"version" : 1, "entity_type":"topic/client", "entity_name" : "topic_name/client_id"}
```

Config API

This proposal also adds broker APIs to alter and view configs for any entity. These requests can be sent to any broker within the cluster. See KIP-4 for more details on the implementation. Once work for KIP-4 completes, we can add these API's as described below.

AlterConfig

- AddedConfigEntry is an array of configs that will be added for that entity
- DeletedConfig is an array of configs that will be deleted for that entity. For deletion on the config property name needs to be specified and not the value. Also, when a config property is deleted, that entity will reapply the default value for that property. For example: if you delete the quota for a clientId, the quota for that client will be set to the default quota.

```
// EntityType can be either topic or client. AddedConfigEntry and DeletedConfig will be an array.
AlterConfigRequest => [EntityType EntityName [AddedConfigEntry] [DeletedConfig]]
    EntityType => string
    EntityName => string
    AddedConfigEntry => [ConfigKey ConfigValue]
        ConfigKey => string
        ConfigValue => string
    DeletedConfig => string

AlterConfigResponse => [EntityType EntityName ErrorCode]
        EntityType => string
          EntityName => string
        ErrorCode => int16
```

DescribeConfig

The DescribeConfig request is used to query configs for entities.

```
DescribeConfigRequest => [EntityType EntityName]
    EntityType => string
        EntityName => string

// ConfigEntry is an array. It will be empty if there is an error. ErrorCode will be non-zero in case of error
DescribeConfigResponse => [EntityType EntityName ConfigEntry]
    EntityType => string
    EntityName => string
        ErrorCode => int16
    ConfigEntry => [ConfigKey ConfigValue]
        ConfigKey => string
        ConfigValue => string
```

Error Codes

We will add these new error codes to the protocol.

Error	Description	Requests
`InvalidEntityConfig`	If the config key or value used in --alter-config and --delete-config is incorrect, InvalidConfig is returned	`Alter`
`InvalidEntity`	Either the entityType or entityName is incorrect. Entity Type must be (topics/clients)	`Alter, Describe`

CLI and Interactive Shell

As described in KIP-4, these commands will have scripts and an interactive shell.

```
# Topic Commands - options are ported from TopicCommand.scala
bin/kafka.sh --alter-config --entity-type [topic/client] --entity-name name --added-config key=value,key=value
--deleted-config key,key,key --broker-list <host : port>
bin/kafka.sh --describe-config --entity-type [topic/client] --entity-name name --broker-list <host : port>
```

Compatibility, Deprecation, and Migration Plan

- *TopicConfigManager has a config_change_XX sequential znode under configs/. The format of data within the config_change node is going to change hence it is important to not make any config changes using TopicCommand until the cluster is fully upgraded.*
- *We will eventually deprecate the tooling that changes entity configs by modifying znodes directly. All requests should be sent to one of the brokers after KIP-4 is complete.*

Migration plan for notifications

Since the format of notifications is going to change, the new code will no longer be able to read topic change notifications written in the older format. The purge interval for notifications is 15 minutes, so any notification older than 15 minutes should be ignored regardless of the content in the znode. In order to not lose any notifications, cluster administrators should not allow any config changes for at least 15 minutes prior to upgrading the cluster. Upon startup, the brokers will parse all notifications and purge the old ones.
After the upgrade is complete, config changes cannot be done using tooling from older releases of Kafka.

If a rollback must be done after config changes have been made using the new format, the config changes must be purged from zookeeper prior to the rollback (since the old code will throw an exception if it reads a notification in the new format).

Rejected Alternatives

Dynamic Service Configs: After plenty of discussion, we've decided to not allow broker(service) configuration be dynamic. There were a couple of approaches:

- Service Config in ZK
- Reload config file via SIGHUP

See the attached discussion thread for more details on this decision.