KIP-133: Describe and Alter Configs Admin APIs

Note that this KIP proposal is derived from a proposal by Grant Henke that was part of KIP-4 - Command line and centralized administrative operations.

Status

Current state: *Adopted*

Discussion thread: *here*

JIRA: KAFKA-3267 - Getting issue details... STATUS

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

Motivation

KIP-4 - Command line and centralized administrative operations outlines the motivation for exposing admin operations via the protocol:

- *Allows clients in any language to administrate Kafka*
    - *Wire protocol is supported by any language*
- *Provides public client for performing admin operations*
    - *Ensures integration test code in other projects and clients maintains compatibility*
    - *Prevents users from needing to use the Command classes and work around standard output and system exits*
- *Removing the need for admin scripts (kafka-configs.sh, kafka-topics.sh, etc) to talk directly to Zookeeper.*
    - *Allows ZNodes to be completely locked down via ACLs*
    - *Further hides the Zookeeper details of Kafka*

A few specific use cases worth pointing out:

1. The Metadata request exposes topic metadata, but it does not expose topic configs. DescribeConfigs will make that information available to any client of the Kafka protocol and the AdminClient will expose it to normal users.
2. AlterConfigs would make it possible to update topic configs.
3. One should be able to tell how a topic or broker is configured including defaults and overrides.

Public Interfaces

ACLs

We will introduce 2 new ACL operations: DescribeConfigs and AlterConfigs. These operations apply to resources that have configs (i.e. Broker and Topic). In addition, DescribeConfigs/AlterConfigs on the Cluster resource allows one to read and write configs on any resource with configs. Finally, the DescribeConfigs and AlterConfigs operations are included in the `All` operation.

Protocol APIs

Wire Format types

KIP-140: Add administrative RPCs for adding, deleting, and listing ACLs introduced a wire format representation for ResourceType and AclOperation. We weill add new values to both types as follows:

`AclOperation`

- `0: Unknown`
- `1: Any`

- 2: All
- 3: Read
- 4: Write
- 5: Create
- 6: Delete
- 7: Alter
- 8: Describe
- 9: ClusterAction
- 10: DescribeConfigs (new)
- 11: AlterConfigs (new)

ResourceType

- 0: Unknown
- 1: Any
- 2: Topic
- 3: Group
- 4: Cluster
- 5: Broker (new)

Describe Configs

DescribeConfigs Request

```
DescribeConfigs Request (Version: 0) => [resource [config_name]]
  resource => resource_type resource_name
    resource_type => INT8
    resource_name => STRING
  config_name => STRING
```

Request semantics:

1. Can be sent to any broker
2. If there are multiple instructions for the same resource in one request the extra request will be ignored
    - This is because the list of resources is modeled server side as a set
    - Multiple resources results in the same end goal, so handling this error for the user should be okay
    - This is similar to how delete topics handles requests
3. If the config_name array is null, all configs are returned. Otherwise, configs with the provided names are returned.
4. Valid resource types are "Topic" and "Broker".
5. If resource_type is "Broker", the resource_name must be the id of the broker processing the request as the brokers don't have any dynamic configs currently (apart from replication quotas configs, which are excluded). All broker configs are read-only. Conversely, there are no read-only topic configs at the moment.
6. Replication, User and Client Quota configs are not supported. See "Future work" for more details.
7. Authorization is as follows:
    a. Topic configs: "DescribeConfigs" on the "Topic" or "Cluster" resource ("DescribeConfigs" is also included in the "All" operation). Unauthorized requests will receive an appropriate AuthorizationFailed error code.
    b. Broker configs: "DescribeConfigs" on the "Broker" or "Cluster" resource ("DescribeConfigs" is also included in the "All" operation). Unauthorized requests will receive an appropriate AuthorizationFailed error code.
8. Errors are reported independently per resource.

DescribeConfigs Response

```
DescribeConfigs Response (Version: 0) => error_code [entities]
  entities => error_code resource_type resource_name [configs]
        error_code => INT16
        resource_type => INT8
    resource_name => STRING
    configs =>
      config_name => STRING
      config_value => STRING
      read_only => BOOLEAN
      is_default => BOOLEAN
      is_sensitive => BOOLEAN
```

Alter Configs

AlterConfigs Request

```
AlterConfigs Request (Version: 0) => [resources] validate_only
  validate_only => BOOLEAN
  resources => resource_type resource_name [configs]
    resource_type => INT8
    resource_name => STRING
    configs =>
      config_name => STRING
      config_value => STRING
```

Request Semantics

1. Can be sent to any broker
2. If there are multiple instructions for the same resource in one request, an InvalidRequestException will be logged on the broker and a single error code for that topic will be returned to the client
    - This is because the list of resources is modeled server side as a map with the resource as the key
3. Valid resource types are "Topic" and "Broker". However, since all broker configs are currently read_only, only the former makes sense until that changes.
4. If an `Alter` operation is attempted on a read-only config, an `UnsupportedOperation` error will be returned for the relevant resource.
5. Authorization is as follows:
    a. Topic configs: "AlterConfigs" on the "Topic" or "Cluster" resource ("AlterConfigs" is also included in the "All" operation). Unauthorized requests will receive an appropriate AuthorizationFailed error code.
    b. Broker configs: "AlterConfigs" on the "Broker" or "Cluster" resource ("AlterConfigs" is also included in the "All" operation). Unauthorized requests will receive an appropriate AuthorizationFailed error code.
6. Replication, User and Client Quota configs are not supported. See "Future work" for more details.
7. The request is not transactional.
    a. If an error occurs for an resource, others could still be updated.
    b. Errors are reported independently per resource.
8. For tools that allow users to alter configs, a validation/dry-run mode where validation errors are reported but no creation is attempted is available via the validate_only parameter.

AlterConfigs Response

```
AlterConfigs Response (Version: 0) => [responses]
  responses => resource_type resource_name error_code error_message
        resource_type => INT8
        resource_name => STRING
    error_code => INT16
    error_message => NULLABLE_STRING
```

Policy

In a similar fashion to KIP-108: Create Topic Policy, we allow users to define a policy class to validate alter configs requests. The config name will be `alter.configs.policy.class.name` and the interface follows:

AlterConfigPolicy

```
package org.apache.kafka.server.policy;

public interface AlterConfigsPolicy extends Configurable, AutoCloseable {

    /**
     * Class containing the create request parameters.
     */
    class RequestMetadata {
        /**
         * Create an instance of this class with the provided parameters.
         *
         * This constructor is public to make testing of <code>AlterConfigPolicy</code> implementations easier.
         */
        public RequestMetadata(ConfigResource resource, Config config) { ... }

        /**
         * Return the Config in the request.
         */
        public Config config() { ... }
    }

    /**
     * Validate the request parameters and throw a <code>PolicyViolationException</code> with a suitable error
     * message if the alter configs request parameters for the provided resource do not satisfy this policy.
     *
     * Clients will receive the POLICY_VIOLATION error code along with the exception's message. Note that
validation
     * failure only affects the relevant resource, other resources in the request will still be processed.
     *
     * @param requestMetadata the alter configs request parameters for the provided resource.
     * @throws PolicyViolationException if the request parameters do not satisfy this policy.
     */
    void validate(RequestMetadata requestMetadata) throws PolicyViolationException;
}
```

Users will have to ensure that the policy implementation code is in the broker's classpath. Implementations should throw the existing `PolicyViolationException` with an appropriate error message if the request does not fulfill the policy requirements. We chose a generic name for the only parameter of the `validate` method in case we decide to add parameters that are not strictly related to the topic (e.g. session information) in the future. The constructor of `RequestMetadata` is public to make testing convenient for users. Under normal circumstances, it should only be instantiated by Kafka. We chose to create separate API classes instead of reusing request classes to make it easier to evolve the latter.

AdminClient APIs

They follow a similar pattern as existing AdminClient APIs:

org.apache.kafka.clients.admin

```
public class AdminClient {
    public DescribeConfigsResult describeConfigs(Collection<ConfigResource> resources, DescribeConfigsOptions
options);
    public AlterConfigsResult alterConfigs(Map<ConfigResource, Config> configs, AlterConfigsOptions options);
}

public class DescribeConfigsOptions {
    public DescribeConfigsOptions timeoutMs(Integer timeout);
}

public class DescribeConfigsResult {
    public Map<ConfigResource, KafkaFuture<Config>> results()
    public KafkaFuture<Map<ConfigResource, Config>> all();
}

public class AlterConfigsOptions {
    public AlterConfigsOptions timeoutMs(Integer timeout);
        public AlterConfigsOptions validateOnly(boolean validateOnly);
}

public class AlterConfigsResult {
    public KafkaFuture<Void> all();
    public Map<ConfigResource, KafkaFuture<Void>> results();
}

public class ConfigResource {
    enum Type {
                BROKER, TOPIC, UNKNOWN;
    }

    public ConfigResource(Type type, String name) { ... }

    public Type type() { ... }
    public String name() { ... }
}


public class Config {
    public Config(Collection<ConfigEntry> configs) { ... }
    public Collection<ConfigEntry> entries() { ... }
    public Config get(String name) { ... }
}
public class ConfigEntry {
    public ConfigEntry(String name, String value) {
        this(name, value, false, false, false);
    }
    public ConfigEntry(String name, String value, boolean isDefault, boolean isSensitive, boolean isReadOnly) {
... }
    public String name() { ... }
    public String value() { ... }
    public boolean isDefault { ... }
    public boolean isSensitive { ... }
    public boolean isReadOnly { ... }
}
```

Proposed Changes

In addition to what is mentioned in the "Public Interfaces" section, it's worth mentioning that the describeConfigs implementation will perform requests to min(1, N) brokers where N is the number of ConfigResources that have a Broker resource type.

Compatibility, Deprecation, and Migration Plan

We are adding new ACL Operations and a new ACL Resource Type, so third-party Authorizer implementations will potentially have to be updated in order to support them. Since this only affects newly introduced protocol and AdminClient APIs, it's not a compatibility issue. It simply means that the new functionality won't be available for users of such Authorizer implementations until they are updated.

With regards to forwards compatibility, ConfigResource.Type has an UNKNOWN element in case it receives data from a newer broker that cannot be mapped to one of the existing enum types.

Rejected Alternatives

1. Allowing sensitive data to be returned: it's good security practice to never expose sensitive data. If necessary, the user can reset the relevant sensitive data (e.g. a password).
2. Introducing a new Configs resource instead of DescribeConfigs and AlterConfigs operations: there is always a one to one mapping between a resource and its configs, so there isn't much value in creating a separate resource for Configs. By adding new operations to existing resources, it's easier to see all the ACLs that affect a given resource.

Future work

1. Forward requests to the relevant brokers in order to return `read-only` broker configs.
2. Support for reading and updating client, user and replication quotas. We initially included that in the KIP, but it subsequently became apparent that a separate protocol and AdminClient API would be more appropriate. The reason is that client/user quotas can be applied on a client id, user or (client id, user) tuple. In the future, the hierarchy may get even more complicated. So, it makes sense to keeping the API simple for the simple cases while introducing a more sophisticated API for the more complex case.