KIP-248 - Create New ConfigCommand That Uses The New AdminClient

This page is meant as a template for writing a KIP. To create a KIP choose Tools->Copy on this page and modify with your content and replace the heading with the next KIP number and a description of your issue. Replace anything in italics with your own description.

Status

Current state: Rejected *[One of "Under Discussion", "Accepted", "Rejected"]*

Discussion thread: *here*

JIRA: Unable to render Jira issues macro, execution error.

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

Motivation

Describe the problems you are trying to solve.

- KIP-4 defines the high level motivation for using the admin client and KAFKA-3268 with its subtasks acts as the umbrella JIRA for this KIP.
- The current implementation of the `ConfigCommand` (`kafka.admin.ConfigCommand`) which is used by `kafka-configs.sh` connects to Zookeeper directly. This prevents the tool from being used in deployments where only the brokers are exposed to clients (i.e. where the zookeeper servers are intentionally not exposed).
- There is a general push to refactor/rewrite/replace tools which need Zookeeper access with equivalents which use the `AdminClient` API. Thus it is necessary to change the `ConfigCommand` so that it no longer talks to Zookeeper directly, but via an intermediating broker.
- Makes the ConfigCommand transparent to the authorization mechanism therefore enables higher level of security. The broker will be able to authorize config admin requests.
- The `AdminClient` API currently lacks any functionality for changing broker, user and client configurations which is possible with the current Zookeeper based `ConfigCommand` implementation.
- Changing the ConfigCommand will increase the user experience for KIP-226 for the above mentioned reasons.

Public Interfaces

Command Line Tools And Arguments

The options accepted by `kafka-configs.sh` command will change:

- `--zookeeper` will be removed, it's functionality won't be available. The design decision behind is that the ConfigCommand tool will be rewritten in the tools module which doesn't depend on the core module. This makes it hard to provide backward compatibility with the current ConfigCommand.
- `--bootstrap-server` was added in ⚠ Unable to render Jira issues macro, execution error. and will be used here.
- `--adminclient.config` option will be added and should be used similarly to other tools, such as --producer.config in the console-producer. This parses a config file and initializes the admin client used internally.
- `--adminclient-property` option will be added. It is a key=value list that will be parsed by the command. It initializes the internal admin client.

A new tool, called scram-credentials.sh will be added. The need for this broker is when people use zookeeper as a credentials store for SCRAM (and currently users have no other option), then direct interaction with zookeeper is required to set up the initial credentials with inter-broker communication. The functionality of the tool will cover the following:

- Add, remove and list SCRAM credentials directly with zookeeper
- Will continue to use the `--zookeeper` option to specify the zookeeper host
- Works similarly to the old config command. Examples:
  - Add new credentials:
    `bin/scram-credentials.sh --zookeeper localhost:2181 --add 'SCRAM-SHA-256=[iterations=8192, password=alice-secret],SCRAM-SHA-512=[password=alice-secret]' --username alice`
  - Describe credentials:
    `bin/scram-credentials.sh --zookeeper localhost:2181 --describe --username alice`
  - Remove credentials:
    `bin/scram-credentials.sh --zookeeper localhost:2181 --delete 'SCRAM-SHA-512' --username alice`

Protocol Changes

KIP-133 introduced the describe and alter admin protocols and KIP-140 a wire format representation for ResourceType. We will modify these to accommodate the new requirements.

Wire Format Types

`ResourceType`

- `0: Unknown`
- `1: Any`
- `2: Topic`
- `3: Group`
- `4: Broker`
- `5: User (new)`
- `6: Client (new)`

`QuotaType (new)`

- `0: ProducerByteRate`
- `1: ConsumerByteRate`
- `2: RequestPercentage`

`QuotaSource (new)`

- `0: ClientInUser`
- `1: DefaultClientInUser`
- `2: User`
- `3: ClientInDefaultUser`
- `4: DefaultClientInDefaultUser`
- `5: DefaultUser`
- `6: Client`
- `7: DefaultClient`

Field Types

Double

A new type needs to be added to transfer quota values. Since the protocol classes in Kafka already uses ByteBuffers it is logical to use their functionality for serializing doubles. The serialization is basically a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout. The ByteBuffer serializer writes eight bytes containing the given double value, in Big Endian byte order, into this buffer at the current position, and then increments the position by eight.

The implementation will be defined in `org.apache.kafka.common.protocol.types` with the other protocol types and it will have no default value much like the other types available in the protocol.

Describe Quotas

The justification for a new protocol is that a quota is quite different from a broker or topic config because a quota can sometimes be identified a simple user, client or even a (user,client) tuple while a topic or a broker config can be identified only by the topic's name or the broker's ID. Moreover quotas have their own well defined types.

DescribeQuotas Request

```
DescribeQuotas Request (Version: 0) => [resource]
  resource => [quota_resource] [quota_type]
    quota_resource => type name
      type => INT8
      name => STRING
    quota_type => INT8
```

Request semantics:

1. Can be sent to any broker
2. If the `name` is empty it means that listing the default quota is asked. Responses will be returned the same way for defaults.
3. If the `quota_type` array is empty, all quotas are returned. Otherwise, quotas with the provided types are returned.
4. Authorization: "DescribeQuotas" can only be interpreted on the "Cluster" resource and represented by the DescribeConfigs ACL due to the similarity in use cases. Unauthorized requests will receive an appropriate AuthorizationFailed error code.

DescribeQuotas Response

```
DescribeQuotas Response (Version: 0) => throttle_time_ms [resource]
  throttle_time_ms => INT32
  resource => [quota_resource] [quota]
    quota_resource => type name
      type => INT8
      name => STRING
    quota_collection => error_code error_message [quota_entry]
      error_code => INT16
      error_message => NULLABLE_STRING
      quota_entry => quota_type quota_value quota_source
        quota_type => INT8
        quota_value => DOUBLE
        quota_source => INT8
```

Alter Quotas

AlterQuotas Request

```
AlterQuota Request (Version: 0) => validate_only [resource]
  validate_only => BOOLEAN
  resource => [quota_resource] [quota]
    quota_resource => type name
      type => INT8
      name => STRING
    quota => quota_type quota_value
      quota_type => INT8
      quota_value => DOUBLE
```

Request Semantics

1. Can be sent to any broker

2. If `name` is empty it means that altering a default quota is asked.
3. Authorization: "AlterQuotas" can only be interpreted on the "Cluster" resource and represented by the AlterConfigs ACL due to the similarity in use cases. Unauthorized requests will receive an appropriate AuthorizationFailed error code.
4. For tools that allow users to alter quota configs, a validation/dry-run mode where validation errors are reported but no creation is attempted is available via the `validate_only` parameter.
5. The AlterQuotas protocol has an incremental semantics. By this we mean that the request will update only those quotas which are sent in the request.
6. Removing quotas will be done by sending a NaN as the value.

AlterQuotas Response

```
AlterQuotas Response (Version: 0) => throttle_time_ms [resource]
  throttle_time_ms => INT32
  resource => [quota_resource] [quota]
    quota_resource => type name
      type => INT8
      name => STRING
    quota => error_code error_message quota_type
      error_code => INT16
      error_message => NULLABLE_STRING
      quota_type => INT8
```

AlterConfigs

This request needs some change as currently the --add-config operation of ConfigCommand would do incremental operations in Zookeeper but the AlterConfigs protocol sets the whole properties object. The purpose of this change to add an boolean parameter to the request so that it can specify the behavior (incremental or set) which needs to be executed.

AlterConfigs Request

```
AlterConfigs Request (Version: 1) => [resources] validate_only incremental_update
  validate_only => BOOLEAN
  incremental_update => BOOLEAN                                           // new addition
  resources => resource_type resource_name [configs]
    resource_type => INT8
    resource_name => STRING
    configs => config_name config_value
      config_name => STRING
      config_value => STRING
```

Request Semantics:

1. The default value of `incremental_update` is false. That means that the request will wipe the node's data and sets what is sent in the request.
2. Setting the `incremental_update` flag to true makes sure that existing configs are not deleted.
3. Deleting a config in incremental mode is done by sending an empty string as value.
4. Other existing semantics aren't changed.

AdminClient APIs

org.apache.kafka.clients.admin

```
public static class Quota {
    public QuotaType type();
    public double value();
    public QuotaSource source();
}

public enum QuotaType {
    PRODUCER_BYTE_RATE((byte) 0, "producer_byte_rate"),
    CONSUMER_BYTE_RATE((byte) 1, "consumer_byte_rate"),
    REQUEST_PERCENTAGE((byte) 2, "request_percentage");

    QuotaType(byte id, String name);
    public byte id();
    public String quotaName();
}

public enum QuotaSource {
    CLIENT_OF_USER((byte) 0, "Client of user"),
    DEFAULT_CLIENT_OF_USER((byte) 1, "Default client of user"),
    USER((byte) 2, "User"),
    CLIENT_OF_DEFAULT_USER((byte) 3, "Client of default user"),
    DEFAULT_CLIENT_OF_DEFAULT_USER((byte) 4, "Default client of default user"),
    DEFAULT_USER((byte) 5, "Default user"), CLIENT((byte) 6, "Client"),
    DEFAULT_CLIENT((byte) 7, "Default client");

    QuotaSource(byte id, String description);
    public byte id();
    public String description();
}

/**
 * Makes sure that the list of resources that is used as key in a hashmap is immutable and has a fixed
implementation for the hashCode.
 */
public class ConfigResourceList {
    public List<ConfigResource> getResourceList();

public class AdminClient {
    public DescribeQuotasResult describeQuotas(Map<ConfigResourceList, Collection<QuotaType>>,
DescribeQuotasOptions options);
    public AlterQuotasResult alterQuotas(Map<ConfigResourceList, Collection<Quota>> configs, AlterQuotasOptions
options);
}
public class DescribeQuotasOptions extends AbstractOptions<DescribeQuotasOptions>  {
    public DescribeQuotasOptions timeoutMs(Integer timeout);
}

public class DescribeQuotasResult {
    public Map<List<Resource>, KafkaFuture<Collection<Quota>>> values();
}

public class AlterQuotasOptions extends AbstractOptions<AlterQuotasOptions> {
    public AlterQuotasOptions timeoutMs(Integer timeout);
        public AlterQuotasOptions validateOnly(boolean validateOnly);
}

public class AlterQuotasResult {
    public Map<List<Resource>, KafkaFuture<Void>> results();
}

public class AlterConfigsOptions extends AbstractOptions<AlterConfigsOptions> {
    public AlterConfigsOptions timeoutMs(Integer timeoutMs);
    public AlterConfigsOptions validateOnly(boolean validateOnly);
    public boolean shouldValidateOnly();
    public AlterConfigsOptions incrementalUpdate(boolean incrementalUpdate); // new
    public boolean shouldUpdateIncrementally(); // new
}
```

Request API

org.apache.kafka.common.requests

```
public class QuotaCollection {
        public QuotaCollection(ApiError error, Collection<Quota> entries);

    public QuotaCollection(Collection<Quota> entries);

    public ApiError error();
    public Collection<Quota> entries();
}

public class DescribeQuotasRequest extends AbstractRequest {

        public static Schema[] schemaVersions();
        public static DescribeQuotasRequest parse(ByteBuffer buffer, short version);

        public static class Builder extends AbstractRequest.Builder {
                public Builder(Map<List<Resource>, Collection<QuotaType>> quotaSettings);
                public DescribeQuotasRequest build(short version);
        }

        public DescribeQuotasRequest(short version, Map<List<Resource>, Collection<QuotaType>> quotaSettings);
        public DescribeQuotasRequest(Struct struct, short version);

        public Map<List<Resource>, Collection<QuotaType>> quotaTypes();
}

public class DescribeQuotasResponse extends AbstractResponse {
        public static Schema[] schemaVersions();

        public DescribeQuotasResponse(int throttleTimeMs, Map<ConfigResourceList,
KafkaFuture<Collection<Quota>>>);
        public DescribeQuotasResponse(Struct struct);

        public Map<List<Resource>, QuotaCollection> quotas();
}

public class AlterQuotasRequest extends AbstractRequest {
        public static Schema[] schemaVersions();

        public static class Builder extends AbstractRequest.Builder {
                public Builder(Map<List<Resource>, QuotaCollection> quotaSettings);
                public DescribeQuotasRequest build(short version);
        }

        public AlterQuotasRequest(short version, Map<List<Resource>, QuotaCollection> quotas, boolean
validateOnly);
        public AlterQuotasRequest(Struct struct, short version);

        public Map<List<Resource>, QuotaCollection> quotas();
}

public class AlterQuotasResponse extends AbstractResponse {
        public static Schema[] schemaVersions();

        public AlterQuotasRequest(short version, Map<List<Resource>, ApiError> quotas, boolean validateOnly);
        public AlterQuotasRequest(Struct struct, short version);

        public Map<List<Resource>, ApiError> errors();
    public int throttleTimeMs();
}
```

New Command Line Interface

The `kafka-config.sh` command line interface will change a little bit in terms of help message and response format as we will use argparse4j for parsing arguments.

Help Message

```
usage: config-command [-h] --entity-type {topics,clients,users,brokers}
                      [--force FORCE] [--add-config ADDCONFIG]
                      [--delete-config DELETECONFIG]
                      (--entity-name ENTITYNAME | --entity-default)
                      (--describe | --alter)
                      (--bootstrap-server BOOTSTRAPSERVERS |
                      --adminclient.config CONFIGPROPERTIES |
                      --adminclient-property ADMINCLIENTPROPERTY)

Change configs for topics, clients, users, brokers dynamically.

optional arguments:
  -h, --help             show this help message and exit
  --entity-type {topics,clients,users,brokers}
                         REQUIRED:      the       type       of      entity
                         (topics/clients/users/brokers)
  --force FORCE          Suppresses console prompts
  --add-config ADDCONFIG
                         Key  Value  pairs  of   configs   to  add.  Square
                         brackets  can  be  used   to  group  values  which
                         contain commas: 'k1=v1,k2=[v1,v2,v2],k3=v3'.
  --delete-config DELETECONFIG
                         Config keys to remove in  the following form: 'k1,
                         k2'.

  You can specify only one in --entity-name and --entity-default

  --entity-name ENTITYNAME
                         Name of entity (client id/user principal name)
  --entity-default       Default entity name for  clients/users (applies to
                         corresponding entity type in command line)

  You can specify only one in --alter, --describe

  --describe             List  configs  for  the  given  entity.  (default:
                         false)
  --alter                Alter the configuration for  the entity. (default:
                         false)

  REQUIRED. You can specify only one in --bootstrap-servers, --adminclient.config

  --bootstrap-server BOOTSTRAPSERVER
                         The  broker  list  string  in  the  form
                         HOST1:PORT1,HOST2:PORT2.
  --command-config COMMANDCONFIG
                         The  config  properties  file  for  the
                         Admin Client.

Process finished with exit code 0
```

Output Format

```
                                        CONFIGS FOR TOPIC topicA

                                     Name   Value                                  Sensitive  Read-
only                 Source
                         compression.type = producer                                   false      false
Default config
                   message.format.version = 1.0-IV0                                    false      false
Default config
                     file.delete.delay.ms = 6000                                       false      false   Dynamic
topic config
    leader.replication.throttled.replicas =                                            false      false
Default config
                        max.message.bytes = 1000012                                    false      false
Default config
                    min.compaction.lag.ms = 0                                          false      false
Default config
                   message.timestamp.type = CreateTime                                 false      false
Default config
                      min.insync.replicas = 1                                          false      false
Default config
                        segment.jitter.ms = 0                                          false      false
Default config
                              preallocate = false                                      false      false
Default config
                     index.interval.bytes = 4096                                       false      false
Default config
                001min.cleanable.dirty.ratio = 0.5                                        false      false
Default config
           unclean.leader.election.enable = false                                      false      false
Default config
                          retention.bytes = 10                                         false      false   Dynamic
topic config
                      delete.retention.ms = 86400000                                   false      false
Default config
                           cleanup.policy = delete                                     false      false
Default config
                                 flush.ms = 9223372036854775807                        false      false
Default config
  follower.replication.throttled.replicas =                                            false      false
Default config
                            segment.bytes = 1073741824                                 false      false
Default config
                             retention.ms = 604800000                                  false      false
Default config
                               segment.ms = 604800000                                  false      false
Default config
      message.timestamp.difference.max.ms = 9223372036854775807                        false      false
Default config
                           flush.messages = 9223372036854775807                        false      false
Default config
                      segment.index.bytes = 10485760                                   false      false
Default config
                       producer.byte.rate = 1000                                       false      false
Default user
```

As seen above, the describe format becomes more organized and it will also return default properties (as the protocol currently supports that). In case of alter we will also do an extra describe after executing the alter and print the most fresh state.

Compatibility, Deprecation, And Migration Plan

Compatibility

Firstly, the --zookeeper option will be removed from kafka-configs.sh and the backing code will be replaced. Therefore every user will need to change `--zookeeper` to `--bootstrap-servers`, `--adminclient-property` or `--adminclient.config`. SCRAM update will be done by the newly introduced `scram-credentials.sh` tool. Other existing behavior will be kept.

Secondly, users as of this KIP would be able to describe all topics or brokers in one step but can't do it for clients and users. For those who have this use case would still need to use the old command for a while (see below). The reason for this change is currently MetadataRequest provides enough information about topics and brokers so it's possible to describe all of them in one step but there's no such information about clients and users.

Finally, backward compatibilty (for instance a 2.0 client wants to admin a 1.0 server) will be impacted as some of the protocols are newly created and doesn't exist in old servers. In these cases users should continue using the scala version of the ConfigCommand by putting the core jar on their classpath.

The old command could be launched through kafka-run-class.sh like this:

```
bin/kafka-run-class.sh kafka.admin.ConfigCommand --zookeeper localhost:2181 --describe --entity-type topics --
entity-name topicA
```

Impact

Communicating through the broker instead of Zookeeper allows us to give more protection to Zookeeper as it could be hidden behind a firewall and you can only allow the broker through it. Also this would allow much finer grain authorization and audit of admin operations in the brokers.

From the CLI point of view the impact should be minimal as only the `--zookeeper` option will change but we can assume the Zookeeper is a more protected resource than the `CLIENT` ports of the brokers, therefore we can assume that they have knowledge about it and change with minimal effort.

From the compatibility point of view there might be a bigger impact as mentioned above. Since the command now uses the wire protocols (including some newly introduced ones) the backward compatibility will be impacted. That means that a user can't use a 2.0 client to administrate a 1.0 broker as in the older broker some of the wire protocols don't exist. This again should be acceptable most of the users as most of the admin commands require the core jar on the classpath which means that most of the time the commands are executed from an environment with the same version of the brokers. Therefore the old tool should still be available.

Deprecation

`kafka.admin.ConfigCommand` will print a warning message saying it is deprecated and will be removed in a future version.

Special Migration Tools

There are no tools required.

Removal Of The Existing Behavior

The current `--zookeeper` option will be removed with this change as it has minimal impact on the current users.

Listing multiple users' and clients' quotas at once won't be possible. If this is required, users would need to use the old tool.

Test Plan

Most of the functionality can be covered with end-to-end tests. There will be unit tests also to verify the protocol and the broker side logic.

Future Considerations

At this moment this ConfigCommand can describe all the topics and all the brokers with one command but can't describe all clients or users. The reason for this is that we can gain the required information for topics and brokers by a MetadataRequest but we have no such protocols for getting a list of users or clients.

Therefore we could have a ConfigEntityList protocol tailored to the needs of the admin client. This protocol would send a list of config entities in the request and get a list of entities in the response. For instance requesting (type:USER name:user1, type:CLIENT name:) resources would return all the clients of user1.