KIP-525 - Return topic metadata and configs in CreateTopics response

Status

Current state: *Accepted*

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

`CreateTopics` response currently only returns success or failure status including any errors. No additional metadata is returned for the topics that are created. When creating topics, configs may be optionally specified in `CreateTopics` request and for configs that are not specified, broker default is used. It will be useful to return the actual configuration of the topic that was created in CreateTopics response to avoid two round trip calls to determine these using `Metadata` and `DescribeConfigs` requests.

KIP-234: add support for getting topic defaults from AdminClient proposed to modify `DescribeConfigs` request to obtain default broker configs used for topic creation to enable users to determine the configs that will be applied before issuing `CreateTopicsRequest`. This is useful, for example, to display default configs in management tools used to create topics. This KIP provides an alternative solution that enables users to obtain this information using `CreateTopics` with `validateOnly=true`. When `CreateTopics` is invoked with `validateOnly=true`, the response will return configs that would have been used if the topic was actually created.

Goals

1. Return topic configs including number of partitions and replication count in CreateTopics response if the user is authorized. This avoids separate `Metadata` and `DescribeConfigs` requests to obtain this after creating the topic.
2. Enable users to obtain default configs that will be applied to new topics using `CreateTopics` with `validateOnly=true`. This is useful for management tools to display default values.
3. Manage access to topic configs using existing ACLs.

Public Interfaces

Protocol changes

CreateTopicsRequest

Version of CreateTopics will be bumped up to 5. Request format will not be changed.

CreateTopicsResponse

Number of partitions, replication factor and topic configs will be included in the response if user has `DescribeConfigs` permission. If user does not have this permission, an error code is returned in the optional tagged field `` `TopicConfigErrorCode` ``.

CreateTopicsResponse version 6

```
{
  "apiKey": 19,
  "type": "response",
  "name": "CreateTopicsResponse",
  // Version 1 adds a per-topic error message string.
  //
  // Version 2 adds the throttle time.
  //
  // Starting in version 3, on quota violation, brokers send out responses before throttling.
  //
  // Version 4 makes partitions/replicationFactor optional even when assignments are not present (KIP-464).
  //
  // Version 5 is the first flexible version.
  // Version 5 also returns topic configs in the response (KIP-525)
  "validVersions": "0-5",
  "flexibleVersions": "5+",
  "fields": [
    { "name": "ThrottleTimeMs", "type": "int32", "versions": "2+", "ignorable": true,
      "about": "The duration in milliseconds for which the request was throttled due to a quota violation, or
zero if the request did not violate any quota." },
    { "name": "Topics", "type": "[]CreatableTopicResult", "versions": "0+",
      "about": "Results for each topic we tried to create.", "fields": [
      { "name": "Name", "type": "string", "versions": "0+", "mapKey": true, "entityType": "topicName",
        "about": "The topic name." },
      { "name": "ErrorCode", "type": "int16", "versions": "0+",
        "about": "The error code, or 0 if there was no error." },
      { "name": "ErrorMessage", "type": "string", "versions": "1+", "nullableVersions": "0+", "ignorable": true,
        "about": "The error message, or null if there was no error." },

      // *************  New fields added by KIP-525 *************//
      { "name": "TopicConfigErrorCode", "type": "int16", "versions": "5+", "tag": 0, "taggedVersions": "5+",
        "about": "Optional topic config error returned if configs are not returned in the response." },
      { "name": "NumPartitions", "type": "int32", "versions": "5+", "default": "-1",
        "about": "Number of partitions of the topic." },
      { "name": "ReplicationFactor", "type": "int16", "versions": "5+", "default": "-1",
        "about": "Replicator factor of the topic." },
      { "name": "Configs", "type": "[]CreatableTopicConfigs", "versions": "5+", "nullableVersions": "5+",
        "about": "Configuration of the topic.", "fields": [
        { "name": "Name", "type": "string", "versions": "5+",
          "about": "The configuration name." },
        { "name": "Value", "type": "string", "versions": "5+", "nullableVersions": "5+",
          "about": "The configuration value." },
        { "name": "ReadOnly", "type": "bool", "versions": "5+",
          "about": "True if the configuration is read-only." },
        { "name": "ConfigSource", "type": "int8", "versions": "5+", "default": "-1", "ignorable": true,
          "about": "The configuration source." },
        { "name": "IsSensitive", "type": "bool", "versions": "5+",
          "about": "True if this configuration is sensitive." }
      ]}
    ]}
  ]
}
```

Admin Client API changes

A new method will be added to CreateTopicsResult to obtain configs. All other methods in CreateTopicsResult will be retained with the same signature.

- `public KafkaFuture<TopicConfig> topicConfig(String topic)`

The updated class is shown below:

CreateTopicsResult.java

```
package org.apache.kafka.clients.admin;

import org.apache.kafka.common.KafkaFuture;
import org.apache.kafka.common.annotation.InterfaceStability;
import org.apache.kafka.common.errors.ApiException;

import java.util.Collection;
import java.util.Map;
import java.util.stream.Collectors;

/**
 * The result of {@link Admin#createTopics(Collection)}.
 *
 * The API of this class is evolving, see {@link Admin} for details.
 */
@InterfaceStability.Evolving
public class CreateTopicsResult {

    private final static int UNKNOWN = -1;

    private final Map<String, KafkaFuture<TopicMetadataAndConfig>> futures;

    CreateTopicsResult(Map<String, KafkaFuture<TopicMetadataAndConfig>> futures) {
        this.futures = futures;
    }

    /**
     * Return a map from topic names to futures, which can be used to check the status of individual
     * topic creations.
     */
    public Map<String, KafkaFuture<Void>> values() {
        return futures.entrySet().stream()
                .collect(Collectors.toMap(Map.Entry::getKey, e -> e.getValue().thenApply(v -> (Void) null)));
    }

    /**TOPIC
     * Return a future which succeeds if all the topic creations succeed.
     */
    public KafkaFuture<Void> all() {
        return KafkaFuture.allOf(futures.values().toArray(new KafkaFuture[0]));
    }

    /**
     * Returns a future that provides topic configs for the topic when the request completes.
     * <p>
     * If broker version doesn't support replication factor in the response, throw
     * {@link org.apache.kafka.common.errors.UnsupportedVersionException}.
     * If broker returned an error for topic configs, throw appropriate exception. For example,
     * {@link org.apache.kafka.common.errors.TopicAuthorizationException} is thrown if user does not
     * have permission to describe topic configs.
     */
    public KafkaFuture<Config> config(String topic) {
        return futures.get(topic).thenApply(TopicMetadataAndConfig::config);
    }

    /**
     * Returns a future that provides number of partitions in the topic when the request completes.
     * <p>
     * If broker version doesn't support replication factor in the response, throw
     * {@link org.apache.kafka.common.errors.UnsupportedVersionException}.
     * If broker returned an error for topic configs, throw appropriate exception. For example,
     * {@link org.apache.kafka.common.errors.TopicAuthorizationException} is thrown if user does not
     * have permission to describe topic configs.
     */
    public KafkaFuture<Integer> numPartitions(String topic) {
        return futures.get(topic).thenApply(TopicMetadataAndConfig::numPartitions);
    }

    /**
     * Returns a future that provides replication factor for the topic when the request completes.
     * <p>
     * If broker version doesn't support replication factor in the response, throw
     * {@link org.apache.kafka.common.errors.UnsupportedVersionException}.
     * If broker returned an error for topic configs, throw appropriate exception. For example,
     * {@link org.apache.kafka.common.errors.TopicAuthorizationException} is thrown if user does not
     * have permission to describe topic configs.
     */
    public KafkaFuture<Integer> replicationFactor(String topic) {
        return futures.get(topic).thenApply(TopicMetadataAndConfig::replicationFactor);
    }

    static class TopicMetadataAndConfig {
        private final ApiException exception;
        private final int numPartitions;
        private final int replicationFactor;
        private final Config config;

        TopicMetadataAndConfig(int numPartitions, int replicationFactor, Config config) {
            this.exception = null;
            this.numPartitions = numPartitions;
            this.replicationFactor = replicationFactor;
            this.config = config;
        }

        TopicMetadataAndConfig(ApiException exception) {
            this.exception = exception;
            this.numPartitions = UNKNOWN;
            this.replicationFactor = UNKNOWN;
            this.config = null;
        }

        public int numPartitions() {
            ensureSuccess();
            return numPartitions;
        }

        public int replicationFactor() {
            ensureSuccess();
            return replicationFactor;
        }

        public Config config() {
            ensureSuccess();
            return config;
        }

        private void ensureSuccess() {
            if (exception != null)
                throw exception;
        }
    }
}
```

Security

Topic configs will be returned only if user has `DescribeConfigs` permissions for the topic. If user doesn't have permissions to describe configs, an error code is returned by the broker. AdminClient method to get topic config will throw `TopicAuthorizationException` in this case,

Proposed Changes

`KafkaApis.handleCreateTopics` will be updated to check `DescribeConfigs` permissions and populate configs in the response if permitted. `Errors.TOPIC_AUTHORIZATION_FAILED` is returned in ``` `topic_config_error` ``` field of the response.

`AdminClient` will be updated to return configs if available or throw appropriate exception.

Compatibility, Deprecation, and Migration Plan

The new config fields will not be populated when new clients talk to older brokers or when old clients talk to newer brokers.

AdminClient will return additional configs in `CreateTopicsResult` using new methods, leaving existing methods as-is. Applications can use the new `topicConfig()` method to retrieve configs from the results. Existing applications can continue to use the existing methods to process success/failure of the requests without any change.

Since configs are authorized using `DescribeConfigs` ACLs, no changes are required in authorizers.

Rejected Alternatives

KIP-234: Return default configs for topics in DescribeConfigsResponse if topic name is null

This KIP proposes to return configs for `CreateTopics` requests regardless of `validateOnly` flag since metadata is useful for created topics as well as before creating topics. KIP-234 was proposing to just return default topic configs. Also, since some configs like replication factor are not returned by `DescribeConfigs`, but are useful for the scenario in KIP-234, it seems better to return these as well as `Config`. Another reason for the different approach in this KIP is security. Even though KIP-234 proposes to return default configs for users about to create topics, ACLs for these need to be based on `DescribeConfigs` ACL for brokers since no topic is specified. Otherwise we will need to change authorizers to authorizer non-literal topics to see if user can create ANY topic. This KIP proposes a different approach that is consistent within the current authorization model and satisfies the requirement in KIP-234.

Return full topic metadata in CreateTopicsResponse

We could return the full topic metadata including replica assignment, but since the requrements we have seen so far has only been for topic configs, this KIP proposes to expose only configs through `CreateTopicsResponse`. This avoids duplicating a lot of fields from `MetadataResponse`. Also, replica assignment generated by brokers is not meaningful with `validateOnly=true` since randomized assignment will change when the topic is subsequently created.