KIP-692: Make AdminClient value object constructors public

- Status
- Motivation
- Public Interfaces / Proposed changes
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives

Status

Current state: *Under Discussion*

Discussion thread: here

JIRA: https://issues.apache.org/jira/browse/KAFKA-10490

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

Motivation

A common strategy in the testing of software components that interact with an external system is to construct a mock representing the behaviour of the external system. There are several JVM based solutions that provide mocking functionality such as Mockito, JMockit, and EasyMock.

When devising test strategies for a system that interacts with a Kafka cluster, employing mocks that behave as the KafkaAdmin has a lot of benefits. It makes tests more robust, simpler and faster compared to interacting with a real cluster. However, constructing the value objects that such mocks would return is currently needlessly complicated because value object classes that the Kafka clients returns lack public constructors.

For example, consider a unit test for a piece of software that uses the `AdminClient.deleteTopics()` method. Setting up a mock that conforms to the contract is easy enough, but creating a `DeleteTopicsResult` instance to return from invoking the method on the mock is not straight forward as the constructor is declared with default access and can not be instantiated by code outside the `org.apache.kafka.clients.admin` package.

While there are ways to work around this limitation, for example by creating a mock of the value object to return, this adds complexity to any user that would want to mock any of the kafka clients while testing their code. As a community, we should strive for making good testing easy.

Mocking AdminClient.deleteTopics() currently might look something like this:

```
DeleteTopicsResult deleteTopicsResult = mock(DeleteTopicsResult.class);
when(deleteTopicsResult.values()).thenReturn(singletonMap(TOPIC_NAME, KafkaFuture.completedFuture
(null)));

AdminClient mockAdminClient = mock(AdminClient.class);
when(mockAdminClient.deleteTopics(singleton(TOPIC_NAME))).thenReturn(deleteTopicsResult);
```

Whereas making constructors public would enable test writers to instead write:

```
AdminClient mockAdminClient = mock(AdminClient.class);
when(mockAdminClient.deleteTopics(singleton(TOPIC_NAME))).thenReturn(
    new DeleteTopicsResult(singletonMap(TOPIC_NAME, KafkaFuture.completedFuture(null))))
);
```

Public Interfaces / Proposed changes

Make the constructors of the following top level classes and their inner classes public:

```
CreateTopicResult
DeleteTopicsResult
ListTopicsResult
DescribeTopicsResult
DescribeClusterResult
DescribeAclsResult
CreateAclsResult
DeleteAclsResult
DescribeConfigsResult
AlterConfigsResult
AlterReplicaLogDirsResult
DescribeLogDirsResult
DescribeReplicaLogsDirsResult
CreatePartitionsResult
DeleteRecordsResult
CreateDelegationTokenResult
RenewDelegationTokenResult
ExpireDelegationTokenResult
DescribeDelegationTokenResult
ListConsumerGroupsResult
ListConsumerGroupOffsetsResult
ElectLeadersResult
AlterPartitionReassignmentResult
ListPartitonReassignmentsResult
RemoveMembersFromConsumerResult
AlterConsumerGroupOffsetsResult
ListOffsetsResult
DescribeClientQuotasResult
AlterClientQuotasResult
DscribeUserScramCredentialsResult
AlterUserScramCredentialsResult
DescribeFeaturesResult
UpdateFeaturesResult
```

This KIP proposes no change in functionality, just a change in the access modifiers for the mentioned constructors to make available the already existing functionality to users outside of the Apache Kafka codebase.

Compatibility, Deprecation, and Migration Plan

There should be no compatibility and migration neccessary for this change. Some Kafka test cases in for example `ConfigCommandTest` could be simplified to not use mock instances of value objects when it makes sense, but this is purely optional.

Rejected Alternatives

Besides leaving this as it is, one might envision some sort of factory setup where construction of value objects would be delegated to a separate class. This would add indirection and complexity with the questionable gain of having a slightly smaller public footprint in the Apache Kafka admin client.