KIP-139: Kafka TestKit library

Status

Current state: *Draft*

Discussion thread:

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

Users of Kafka in the JVM often write integration tests involving a Kafka cluster that is launched in the same JVM process. It would be helpful for such users if Kafka were to provide a public Java API as a separate library that would make it simple to start/stop a Kafka cluster in integration tests. Additionally, it should be easy to launch the cluster with TLS and/or SASL enabled.

Public Interfaces

We will introduce a small number of classes in the `org.apache.kafka.testkit` package, which will be considered public API. That is, we will only make incompatible changes in major releases after a suitable deprecation period.

EmbeddedBroker.java

```
package org.apache.kafka.testkit;

/**
 * Runs an in-memory, "embedded" instance of a Kafka broker using a randomly generated port by default.
 * <p>
 * Requires a running ZooKeeper instance to connect to.
 */
public class EmbeddedBroker {

    public EmbeddedBroker(final Map<String, String> configs) throws IOException { ... }

    public EmbeddedBroker(final Map<String, String> configs, Time time) throws IOException { ... }


    /** Return the broker id */
    public int id() { ... }

    /**
     * The ZooKeeper connection string aka `zookeeper.connect`.
     */
    public String zooKeeperConnect() { ... }

    /** Return the broker's log directory */
    public Path logDir() { ... }

    /**
     * Stop the broker.
     */
    public void stop() { ... }
}
```

EmbeddedCluster.java

```
package org.apache.kafka.testkit;

public class EmbeddedCluster extends ExternalResource {

    /** Create a Kafka cluster with the requested number of brokers and with the provided configs */
    public static EmbeddedCluster create(int numBrokers, Map<String, String> configs) { ... }

    /** Create a Kafka cluster with one broker per passed map of configs. */
    public EmbeddedCluster(Collection<Map<String, String>> configs) {
        this.configs = configs;
    }

    public void start() { ... }


    public TestAdmin admin() { ... }

    public boolean stopBroker(int brokerId) { ... }

    public void stop() { ... }

    public List<String> bootstrapServers(String listenerName) { ... }

    public String zooKeeperConnectString() { ... }

    public List<EmbeddedBroker> brokers() { ... }

    public Integer leader(TopicPartition topicPartition) { ... }
}
```

TestAdmin.java

```
package org.apache.kafka.testkit;

public class TestAdmin {

    /**
     * Create a Kafka topic with the given parameters.
     *
     * @param topic       The name of the topic.
     * @param partitions  The number of partitions for this topic.
     * @param replication The replication factor for (the partitions of) this topic.
     */
    public void createTopic(final String topic, final int partitions, final int replication) { ... }

    /**
     * Create a Kafka topic with the given parameters.
     *
     * @param topic       The name of the topic.
     * @param partitions  The number of partitions for this topic.
     * @param replication The replication factor for (partitions of) this topic.
     * @param topicConfig Additional topic-level configuration settings.
     */
    public void createTopic(final String topic,
                            final int partitions,
                            final int replication,
                            final Properties topicConfig) { ... }

    public void deleteTopic(final String topic) { ... }
}
```

TimeUtils.java

```
package org.apache.kafka.testkit;

public class TimeUtils {

    /**
     * Wait for condition to be met for at most {@code maxWaitMs} and throw assertion failure otherwise.
     * This should be used instead of {@code Thread.sleep} whenever possible as it allows a longer timeout to
be used
     * without unnecessarily increasing test time (as the condition is checked frequently). The longer timeout
is needed to
     * avoid transient failures due to slow or overloaded machines.
     */
    public static void waitForCondition(final BooleanSupplier testCondition, final long maxWaitMs, String
conditionDetails) { ... }

}
```

SslConfigBuilder.java

```
package org.apache.kafka.testkit;

public class SslConfigBuilder {

    public static SslConfigBuilder server(Path trustStorePath) { ... }

    public static SslConfigBuilder client(Path trustStorePath) { ... }

    public SslConfigBuilder setHost(String host) { ... }

    public SslConfigBuilder setCertificateAlias(String certificateAlias) { ... }

    public Map<String, String> build() { ... }
}
```

SaslConfigBuilder.java

```
package org.apache.kafka.testkit;

public class SaslConfigBuilder {

    public static SslConfigBuilder server(Path trustStorePath) { ... }

    public static SslConfigBuilder client(Path trustStorePath) { ... }

    public Map<String, String> build() { ... }
}
```

Proposed Changes

We will introduce a new artifact called kafka-testkit that will contain the Java classes for the test library. This module will depend on the clients and core kafka artifacts, but it won't depend on any of the Kafka test artifacts. As such, users won't have our tests on their classpath.

Compatibility, Deprecation, and Migration Plan

Since this is a new library, there is no compatibility issue. Going forward, we will have to evolve the library carefully so that compatibility is maintained.

Rejected Alternatives

1. Continue with the status quo where users who want to write integration tests involving a Kafka cluster have to rely on internal APIs.