KIP-470: TopologyTestDriver test input and output usability improvements

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

This KIP is inspired by the Discussion in KIP-456: Helper classes to make it simpler to write test logic with TopologyTestDriver

The stream application code is very compact and the test code is a lot of bigger code base than actual implementation of the application, that's why it would be good to get test code easily readable and understandable and that way also maintainable.

The proposal in KIP-456 was to add alternate way to input and output topic, but this KIP enhance those classes and deprecate old functionality to make clear interface for test writer to use.

When using the old TopologyTestDriver you need to call ConsumerRecordFactory to create ConsumerRecord passed into pipeInput method to write to topic. Also when calling readOutput to consume from topic, you need to provide correct Deserializers each time.

You easily end up writing helper methods in your test classes, but this can be avoided when adding generic input and output topic classes to implement the needed functionality.

Also the logic of the old TopologyTestDriver is confusing, when you need to pipe ConsumerRecords to produce record to input topic and receive ProducerRecords when consuming from output topic.

Non-existing topic and no record in the queue scenarious are modified to throw Exception instead of returning null.

Public Interfaces

TopologyTestDriver

```
package org.apache.kafka.streams;

public class TopologyTestDriver {
    public TopologyTestDriver(Topology topology, Properties config); // existing constructor
    @Deprecate public TopologyTestDriver(Topology topology, Properties config, long initialWallClockTimeMs);
    public TopologyTestDriver(Topology topology, Properties config, Instant initialWallClockTime);

    @Deprecate public void advanceWallClockTime(long advanceMs); // can trigger wall-clock-time punctuation
        public void advanceWallClockTime(Duration advance); // can trigger wall-clock-time punctuation

    //Deprecate old pipe and read methods
    @Deprecate public void pipeInput(ConsumerRecord<byte[], byte[]> record); // can trigger event-time
punctuation
    @Deprecate public void pipeInput(List<ConsumerRecord<byte[], byte[]>> records); // can trigger event-time
punctuation
    @Deprecate public ProducerRecord<byte[], byte[]> readOutput(String topic);
    @Deprecate public <K, V> ProducerRecord<K, V> readOutput(String topic, Deserializer<K> keyDeserializer,
Deserializer<V> valueDeserializer);

        // methods for TestTopic object creation
        public final <K, V> TestOutputTopic<K, V> createOutputTopic(final String topicName, final Serializer<K>
keySerializer, final Serializer<V> valueSerializer);
        // Uses current system time as start timestamp. Auto-advance is disabled.
        public final <K, V> TestInputTopic<K, V> createInputTopic(final String topicName, final Deserializer<K>
keyDeserializer, final Deserializer<V> valueDeserializer);
    //Uses provided startTimestamp and autoAdvance duration for timestamp generation
        public final <K, V> TestInputTopic<K, V> createInputTopic(final String topicName, final Deserializer<K>
keyDeserializer, final Deserializer<V> valueDeserializer, final Instant startTimestamp, final Duration
autoAdvance);

    ...
}
```

TestInputTopic

```
package org.apache.kafka.streams;

public class TestInputTopic<K, V> {
    //Create by TopologyTestDriver, Constructors are package private

    //Timestamp handling
    //Record timestamp can be provided when piping input or use internally tracked time configured with
parameters:
    //startTimestamp the initial timestamp for generated records, if not provided uses current system time as
start timestamp.
    //autoAdvance the time increment per generated record, if not provided auto-advance is disabled.

    //Advances the internally tracked time.
    void advanceTime(final Duration advance);

        //Methods to pipe single record
    void pipeInput(final V value);
    void pipeInput(final K key, final V value);

    // Use provided timestamp, does not auto advance internally tracked time.
    void pipeInput(final V value, final Instant timestamp);
    void pipeInput(final K key, final V value, final Instant timestamp);

    // Method with long provided to support easier migration of old tests
   void pipeInput(final K key, final V value, final long timestampMs);

    // If record timestamp set, does not auto advance internally tracked time.
    void pipeInput(final TestRecord<K, V> record);

        //Methods to pipe list of records
    void pipeValueList(final List<V> values);
    void pipeKeyValueList(final List<KeyValue<K, V>> keyValues);

    // Use provided timestamp, does not auto advance internally tracked time.
    void pipeValueList(final List<V> values, final Instant startTimestamp, final Duration advanceMs);
    void pipeKeyValueList(final List<KeyValue<K, V>> keyValues, final Instant startTimestamp, final Duration
advanceMs);

    // If record timestamp set, does not auto advance internally tracked time.
    void pipeRecordList(final List<? extends TestRecord<K, V>> records);
}
```

TestOutputTopic

```
package org.apache.kafka.streams;

public class TestOutputTopic<K, V> {
    //Create by TopologyTestDriver, Constructors are package private

        //Method to check queue size
    final long getQueueSize();
    final boolean isEmpty();

        //Methods to readOutput, throw NoSuchElement if no record in queue
    V readValue();
    KeyValue<K, V> readKeyValue();
    TestRecord<K, V> readRecord();

    //Output as collection
    List<V> readValuesToList();
    List<KeyValue<K, V>> readKeyValuesToList();
    Map<K, V> readKeyValuesToMap();
    List<TestRecord<K, V>> readRecordsToList();
}
```

TestRecord

```
package org.apache.kafka.streams.test;
public class TestRecord<K, V> {
        //Constructors
    public TestRecord(final V value);
    public TestRecord(final K key, final V value);
    public TestRecord(final K key, final V value, final Headers headers);
    public TestRecord(final K key, final V value, final Instant recordTime);
    public TestRecord(final K key, final V value, final Headers headers, final Instant recordTime);
    public TestRecord(final K key, final V value, final Headers headers, final Long timestamp);

        //Constructor based on existing record
    public TestRecord(final ConsumerRecord<K, V> record);
    public TestRecord(final ProducerRecord<K, V> record);

        // Methods like in ProducerRecord / ConsumerRecord
    public Headers headers();
    public K key();
    public V value();
    public Long timestamp();

        // Getters
    public Headers getHeaders();
    public K getKey();
    public V getValue();
    public Instant getRecordTime();

        //Overrides
    public String toString();
    public boolean equals(Object o);
    public int hashCode();
}
```

OutputVerifier

```
package org.apache.kafka.streams.test;

//Recommended to use normal assertion library methods
@Deprecated
public class OutputVerifier {
...
```

ConsumerRecordFactory

```
package org.apache.kafka.streams.test;

//Similar functionality now in TestInputTopic
@Deprecated
public class ConsumerRecordFactory<K, V> {
...
```

Proposed Changes

This improvement adds TestInputTopic class which replaces TopologyTestDriver and ConsumerRecordFactory methods as one class to be used to write to Input Topics and TestOutputTopic class which collects TopologyTestDriver reading methods and provide typesafe read methods.

Example

```
public class SimpleTopicTest {

  private TopologyTestDriver testDriver;
  private TestInputTopic<String, String> inputTopic;
  private TestOutputTopic<String, String> outputTopic;

  @Before
  public void setup() {
    testDriver = new TopologyTestDriver(TestStream.getTopology(), TestStream.getConfig());
        inputTopic = testDriver.createInputTopic(TestStream.INPUT_TOPIC, new StringDeserializer(), new
StringDeserializer());
    outputTopic = testDriver.createOutputTopic(TestStream.OUTPUT_TOPIC, new StringSerializer(), new
LongSerializer());
  }

  @After
  public void tearDown() {
      testDriver.close();
  }

  @Test
  public void testOneWord() {
    //Feed word "Hello" to inputTopic and no kafka key, timestamp is irrelevant in this case
    inputTopic.pipeInput("Hello");
    assertThat(outputTopic.readValue()).isEqualTo("Hello");
    //No more output in topic
    assertThat(outputTopic.isEmpty()).isTrue();
  }
}
```

- New Example utilizing new classes test added to streams/examples/src/test/java/org/apache/kafka/streams/examples/wordcount /WordCountDemoTest.java
- Examples in Testing Kafka Streams https://kafka.apache.org/22/documentation/streams/developer-guide/testing.html updated to use new TopolocyTestDriver, TestInputTopic and TestOutputTopic

Compatibility, Deprecation, and Migration Plan

There are no compatibility issues.

The tests utilizing old TopologyTestDriver can still use deprecated methods.

Rejected Alternatives

- This is replacing KIP-456: Helper classes to make it simpler to write test logic with TopologyTestDriver
- Deprecate current TestTopologyDriver and move new to test package. This would have enabled to keep also TestInputTopic and TestOutputTopic classes in test package, not in very crowded streams root package.
- Add ClientRecord interface to client package and modifiy ProducerRecord (and / or ConsumerRecord) to implement it, to be to utilize OutputVerifier with ProducerRecord and TestRecord