Item 226
git_comments:

1. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
2. * * This class provides access to {@link TopologyTestDriver} protected methods. * It should only be used for internal testing, in the rare occasions where the * necessary functionality is not supported by {@link TopologyTestDriver}.
3. * * Get the processor context, setting the processor whose name is given as current node * * @param processorName processor name to set as current node * @return the processor context
4. * * Get a processor by name * * @param name the name to search for * @return the processor matching the search name

git_commits:

1. **summary:** KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [part 2] (#4986)
   message: KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [part 2] (#4986) * KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [part 2] * Refactor: -KTableFilterTest.java -KTableImplTest.java -KTableMapValuesTest.java -KTableSourceTest.java * Add access to task, processorTopology, and globalTopology in TopologyTestDriver via TopologyTestDriverWrapper * Remove unnecessary constructor in TopologyTestDriver * Change how TopologyTestDriverWrapper#getProcessorContext sets the current node Reviewers: John Roesler <john@confluent.io>, Matthias J. Sax <matthias@confluent.io>, Guozhang Wang <wangguoz@gmail.com>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [part 2]
   body: This PR is a further step towards the complete replacement of `KStreamTestDriver` with `TopologyTestDriver`. * Add task, processorTopology, and globalTopology access to TopologyTestDriverAccessor * Add condition to prevent NPE in ProcessorContextImpl * Refactor: - KTableFilterTest - KTableSourceTest - KTableMapValuesTest - KTableImplTest. edit: To simplify the review process some straightforward changes were moved to another [PR](https://github.com/apache/kafka/pull/5052).

github_pulls_comments:

1. Rebased on current trunk to fix merge conficts.
2. Hey @h314to , Thanks for this latest PR! I'll take a look as soon as I have a chance.
3. Meta comment: as it is easier to review smaller PRs, it might be worth the exclude some classes from this PR and tackle them individually via multiple PRs: - `KTableAggregateTest` to figure out the caching issue - don't introduce `allProcessorNames()` but use `Topology#describe()` -- exclude all classes that use `allProcessorNames()` and don't introduce `allProcessorNames()` in the first place - `MockProcessor` and `OutputVerifier` (this might be a follow up OR as the classes seem to overlap with second bullet point). Whatever works best :) Let us know.
4. **body:** Yes, smaller PR are simpler to review. In order to reduce the size of this one I reverted the changes to the following classes, which will be tackled in subsequent PRs: * Reverted changes to `KTableAggregateTest`. * `allProcessorNames` is gone with my previous commit and there is no further need for it * Reverted changes to `KTableKTableInnerJoinTest`, `KTableKTableOuterJoinTest`, and `KTableKTableLeftJoinTest`. These are the ones `MockProcessor` is used more extensively, and thus, the ones which would benefit the most from being rewritten with `OutputVerifier`.
   label: code-design
5. **body:** I rebased on `trunk` and squashed. New changes are in a separate commit to ease review. I simplified the way the current node is set in `getProcessorContext` which allowed me to tidy up `TopologyTestDriverWrapper` a bit. I also needed to fix a few calls to `ConsumerRecordFactory#create` because the recently added record header support (KAFKA-6850) made calls with `null` value ambiguous.
   label: code-design
6. Merged to trunk, and cherry-picked to 2.0.
7. We still have a bunch test classes that uses KStreamTestDriver. Since the 2.0 code freeze is today I think the final PR for completely getting rid of them will only go into trunk then. @h314to Please ping us once you have the final PR to remove KStreamTestDriver ready.

github_pulls_reviews:

1. A couple of tests in this class fail with `TopologyTestDriver`. My guess is that this is due to some problem with caching. Using `TopologyTestDriver`, regardless of the value I set for `StreamsConfig.CACHE_MAX_BYTES_BUFFERING_CONFIG`, I get the same results out of `MockProcessor.processed` as if the cache was 0. In fact, setting the `KStreamTestDriver` cache to zero makes it yield the exact same results as `TopologyTestDriver`. I'll keep looking into it. If you have any tip about what might be going wrong please let me know.
2. **body:** Removed this constructor after rebasing, since recent commits made it no longer necessary.
   label: code-design
3. **body:** nit: move to next line ("weird" formatting)
   label: code-design
4. as above
5. **body:** nit: add `final`
   label: code-design
6. as above

7. Not entirely sure, but seems to be a bug in `TopologyTestDriver` related to `ThreadCache` -- `KStreamTestDriver` creates a cache and uses it as member variable and passes it into `store.init(...)` -- the new `TopologyTestDriver` also creates a cache and passes it into `GlobalProcessorContextImpl` and the created `StreamTask` -- maybe, internally the cache is not forwarded correctly such that it is not used when stores are initialized. I would need to do a deeper analysis why the cache is not forwarded to the stores correctly. Hope the pointes help. If not sufficient, please let us know and we can dig deeper into it from our side.
8. **body:** nit: fix typo `store[s]` ;)
   label: documentation
9. I am wondering if we should rewrite this using `Topology#describe()` instead of hooking into the test driver? Actually same thought about `driver.getAllStateStores()` ?
10. **body:** Wondering, if we should get rid of `MockProcessor` and update the test setup to pipe the result into a topic instead. Than we would use `OutputVerifier` instead of dealing with all those concatenated Strings.
    label: code-design
11. Yeah, it is wierd. Changed.
12. Done
13. Done
14. **body:** Thanks for the tips! I'll look into it a bit longer, to see if I can find out what's wrong. If I hit a dead end I'll let you know.
    label: code-design
15. :) fixed.
16. **body:** That's a good idea. `allProcessorNames` is only used in this test, and its functionally is easily replicated by using `Topology#describe()`. In the case of `allStateStores`, it provides a bit more functionality than `allProcessorNames` since it returns a `Map<String, StateStore>`. It is also part of the public API. Getting all processor names could be done using `TopologyWrapper` to access the `InternalTopologyBuilder`, so it could in principle be removed. However, it could be useful for end users' testing, which do not have access to `TopologyWrapper`, so it might be nice to keep it around.
    label: code-design
17. **body:** That sounds way cleaner. If it's ok I would like to implement that in a subsequent PR, to avoid cramming even more changes into this one.
    label: code-design
18. **body:** I find this case a little confusing. `currentNode().stateStores == null` would seem to imply also that we don't have access to the store, or any store for that matter. Why do we allow this case to pass though?
    label: code-design
19. **body:** Yes, that's a bit messy. I'm going to change the way we get the processor context, and then we won't need this. Since there are already a few conflicts I'm also going to squash and rebase on current trunk.
    label: code-design
20. Don't understand the comment. What StateStore does it refer to?
21. **body:** nit: remove comment -- clear from the code itself (my believe is, if a test needs comments you need to rewrite the test :))
    label: code-design
22. **body:** nit: as above; remove
    label: code-design
23. once more
24. **body:** This is only used once -- would suggest to inline into `assertTopologyContainsProcessor`. Additionally, the check for `assertNotNull` is not self-expressive. As the return type of `assertTopologyContainsProcessor` is `void` I would suggest to just call `return` (instead of `return node;`) if the node was found and replace `return null;`/`assertNotNull` with `throw new AssertionError(...)` WDYT?
    label: code-design
25. remove
26. remove and rename test to `shouldCreateSourceAndSinkNodesForRepartitioningTopic`
27. +1 from me.
28. **body:** We should connect the state stores with this request processor instead of enforcing us to remember which processor to set while initializing. E.g. here we can do (note I removed the procNames): ``` private void doTestValueGetter(final StreamsBuilder builder, final KTableImpl<String, Integer, Integer> table2, final KTableImpl<String, Integer, Integer> table3, final String topic1) { final Topology topology = builder.build(); KTableValueGetterSupplier<String, Integer> getterSupplier2 = table2.valueGetterSupplier(); KTableValueGetterSupplier<String, Integer> getterSupplier3 = table3.valueGetterSupplier(); InternalTopologyBuilder topologyBuilder = TopologyWrapper.getInternalTopologyBuilder(topology); topologyBuilder.connectProcessorAndStateStores(table2.name, getterSupplier2.storeNames()); topologyBuilder.connectProcessorAndStateStores(table3.name, getterSupplier3.storeNames()); try (final TopologyTestDriverWrapper driver = new TopologyTestDriverWrapper(topology, props)) { KTableValueGetter<String, Integer> getter2 = getterSupplier2.get(); KTableValueGetter<String, Integer> getter3 = getterSupplier3.get(); getter2.init(driver.getProcessorContext(table2.name)); getter3.init(driver.getProcessorContext(table3.name)); // same below ``` Ditto elsewhere for initializing the getters.
    label: code-design
29. Better name as "setCurrentNodeInProcessorContext"? And then in java docs mention that it returns the processor context with current node set.
30. I think this is because this function is only for init the getters, which requires the state stores is connected, hence accessible in `init` to the node.
31. Should we check `globalTopology` as well before give up and throw?
32. Yes, that's it. Thanks for clearing it up @guozhangwang . I tried to make it clearer while addressing your other comments.
33. Good point. Done
34. Done
35. Done
36. **body:** Yes, that makes it much clearer. I changed it.
    label: code-design
37. Done
38. Done
39. **body:** This way is so much better. I wasn't happy with my implementation, but couldn't find a cleaner way to do it. Thank you!
    label: code-design
40. **body:** I agree. Changed the name and cleaned up the docs a bit.
    label: documentation

41. Yes, we should. Added the global topology check.

jira_issues:

1. **summary:** Rewrite test to use new public TopologyTestDriver
   description: With KIP-247 we added public TopologyTestDriver. We should rewrite out own test to use this new test driver and remove the two classes ProcessorTopoogyTestDriver and KStreamTestDriver.

jira_issues_comments:

1. I'd like to work on this. I'm assigning the issue to myself.
2. Hi Filipe, Are you still working on this?
3. Hi. Yes, I'm still working on in. Got a few busy days so I couldn't finish it as soon as I'd like, but I won't take much longer.
4. Hi Filipe, I have been working on a similar task (KAFKA-6473) and discovered something that will probably be a problem for you. This task requires the streams project to have a test dependency on the stream:test-utils project, but streams:test-utils already has a compile dependency on the streams project. I'm no Gradle expert, but as far as I can tell, there's no way to break this circular dependency, at least without doing something exotic in the gradle config. We had a discussion in this mailing list thread: [[DISCUSS] KIP-267|[http://mail-archives.apache.org/mod_mbox/kafka-dev/201803.mbox/%3CCAAyirGsovAzRMLa91nd6rzceQgEcNcLMt7ZrXVN7M1Psj4jCmQ%40mail.gmail.com%3E]|http://mail-archives.apache.org/mod_mbox/kafka-dev/201803.mbox/%3CCAAyirGsovAzRMLa91nd6rzceQgEcNcLMt7ZrXVN7M1Psj4jCmQ%40mail.gmail.com%3E].] . Here's what we settled on: {quote}I would propose we restructure the streams directory thusly: streams/ (artifact name := "streams", the actual streams code lives here) - test-utils/ (this is the current test-utils artifact, depends on "streams") - tests/ (new module, depends on "streams" and "test-utils", *NO published artifact*) This gets us out of the circular dependency without having to engage in any Gradle shenanigans while preserving "test-utils" as a separate artifact. This is good because: 1) the test-utils don't need to be in production code, so it's nice to have a separate artifact, 2) test-utils is already public in 1.1, and it's a bummer to introduce users' code when we can so easily avoid it. {quote}   Another result of the discussion is that I'm actually going to side-step this issue for KAFKA-6473, so I won't be doing any restructuring in the course of my work. I'm just sharing these ideas with you for your context.   Hope you're well! -John
5. Also worth mentioning is [~guozhang]'s reply: {quote}I think I agree with your proposed changes, in fact in order to not scatter the test classes in two places maybe it's better to move all of them to the new module. One caveat is that it will make streams' project hierarchy inconsistent with other projects where the unit test classes are maintained inside the main artifact package, but I think it is a good cost to pay, plus once we start publishing test-util artifacts for other projects like client and connect, we may face the same issue and need to do this refactoring as well. {quote}
6. Hi John, So far I've done most of the refactor and I haven't had any complaints from gradle (just did a clean compile). Now I'm refactoring the last class, KStreamTestDriver. Probably the circular dependency issue will show up now. It's nice to know it is a known issue. Thanks for the heads up!
7. Hi Filipe, I hope you're doing well. I'm trying to figure out why this is a problem for me but not for you... can you share the exact command you're using to run the tests? Thanks, -John
8. **body:** Hi John, I also ran into the cyclic dependency issue. For instance, running: {code:java} ./gradlew streams:test --tests org.apache.kafka.streams.processor.TopologyBuilderTest {code} resulted in: {code:java} Task :streams:compileTestJava FAILED warning: [options] bootstrap class path not set in conjunction with -source 1.7 /Users/agapito/Development/apache/kafka/streams/src/test/java/org/apache/kafka/streams/kstream/internals/GlobalKTableJoinsTest.java:24: error: cannot find symbol import org.apache.kafka.streams.TopologyTestDriver; {code} I added :streams:test-utils as a dependency to :streams in build gradle, which then led to: {code:java} FAILURE: Build failed with an exception. * What went wrong: Circular dependency between the following tasks: :streams:copyDependantLibs +--- :streams:jar | \--- :streams:copyDependantLibs (*) \--- :streams:test-utils:jar +--- :streams:test-utils:classes | \--- :streams:test-utils:compileJava | \--- :streams:jar (*) \--- :streams:test-utils:copyDependantLibs \--- :streams:jar (*) (*) - details omitted (listed previously) {code} I got rid of it by adding the following changes to build.gradle: {code:java} diff --git a/build.gradle b/build.gradle index 5e4c35643..17b49258d 100644 --- a/build.gradle +++ b/build.gradle @@ -921,6 +921,7 @@ project(':streams') { testCompile project(':clients').sourceSets.test.output testCompile project(':core') testCompile project(':core').sourceSets.test.output + testCompile project(':streams:test-utils').sourceSets.main.output testCompile libs.junit testCompile libs.easymock testCompile libs.bcpkix @@ -965,7 +966,7 @@ project(':streams:test-utils') { archivesBaseName = "kafka-streams-test-utils" dependencies { - compile project(':streams') + compile project(':streams').sourceSets.main.output compile project(':clients') testCompile project(':clients').sourceSets.test.output {code} This way both the tests and the jar build run cleanly. I'm not sure if this is the best solution, but this issue entails a large refactor (not too complicated but there are a lot of test classes to change) so it at least allows me to continue refactoring and testing my changes without worrying about the cyclic dependency. I only figured it out today. Previously I was just running the tests in Intellij, where I added test-utils_main as a dependency of streams -> streams_test module, so it was all working fine there. I hope this helps.
   label: code-design
9. Thanks for sharing your experience and the workaround Filipe! This is very helpful. John is going to start a discussion thread in the mailing list for a general proposal to address gradle's circular dependency issue, and we'd love to see you chime in.
10. Yeah, thanks [~h314to]! I had tried it with just half of that change, and it obviously didn't work. I'm really glad you figured that out! FYI, there are other sub-projects in kafka that are going to follow this pattern, so I'm going to start a dev mailing list discussion to share your pattern. Also, for completeness, I had to augment your build.gradle patch slightly to get './gradlew streams:test-utils:test' to pass: {noformat} diff --git a/build.gradle b/build.gradle index 5e4c35643..2f38bf28b 100644 --- a/build.gradle +++ b/build.gradle @@ -921,6 +921,7 @@ project(':streams') { testCompile project(':clients').sourceSets.test.output testCompile project(':core') testCompile project(':core').sourceSets.test.output + testCompile project(':streams:test-utils').sourceSets.main.output testCompile libs.junit testCompile libs.easymock testCompile libs.bcpkix @@ -965,11 +966,12 @@ project(':streams:test-utils') { archivesBaseName = "kafka-streams-test-utils" dependencies { - compile project(':streams') + compile project(':streams').sourceSets.main.output compile project(':clients') testCompile project(':clients').sourceSets.test.output testCompile libs.junit + testCompile libs.rocksDBJni testRuntime libs.slf4jlog4j } {noformat} The reason is that we are skipping :streams:copyDependantLibs during test-utils:compile now (to avoid the circular dependency), so we have to explictly depend in testCompile on any libs that would have been transitively pulled in from :streams (and are in the test's run-time code path). In case you run into a similar error when you run the full test suite, the error I saw was: {noformat} java.lang.ClassNotFoundException: org.rocksdb.RocksDBException{noformat}
11. Filipe would like you to take a look at https://github.com/apache/kafka/pull/4760.
12. Hi John. Great, I'm glad I could help. I haven't bumped into the dependencies error yet, but I certainly will. Thanks for the heads up!

13. Hey [~h314to], I hope you're doing well. FYI, we have just merged [https://github.com/apache/kafka/pull/4760] incorporating your pattern for hooking test-utils into the Streams test dependencies. You may have conflicts to resolve when you rebase on trunk. Thanks, -John
14. Hi John. I hope you are also doing well. Thanks for the warning! Since fixing this issue requires changing a lot of files I've been rebasing frequently to keep conflicts to a minimum, so I already have your commit. Nice to see that it passed all checks and that it is all working fine. By the way, I've already migrated a lot of tests from KStreamTestDriver and ProcessorTopoogyTestDriver to TopologyTestDriver (about 40 classes so far). Should I open a pull request, marked as [WIP], to start getting some feedback, or is it preferable to wait until I finish the refactor (I'm missing about 10 classes) before opening the PR?
15. Definitely :) That will let reviewers to provide some early feedbacks to your changes.
16. Yeah, I agree. Just prefix it as WIP, and no one will judge ;) I personally like to collect a high-level round of feedback asap for my PRs.
17. Hey Filipe, Just to keep you in the loop, we discovered a problem with the dependency strategy we previously discussed, but it's fixed by going back to the regular dependency from `test-utils` to `streams` and declaring a `testCompileOnly` dependency from `streams` to `test-utils`. (See [https://github.com/apache/kafka/pull/4821)] Relevant to you is just that you'll again see a conflict next time you rebase. Hope you're well, -John
18. Hi John. Thanks for the warning. Great work! Breaking the cycle on the other side much cleaner. Unfortunately, I started having some issues with tests after rebasing on the current trunk. For instance, running: {noformat} ./gradlew streams:test -Dtest.single=KStreamKStreamJoinTest {noformat} I would get: {noformat} org.apache.kafka.streams.kstream.internals.KStreamKStreamJoinTest > initializationError FAILED java.lang.NoClassDefFoundError: Lorg/apache/kafka/streams/test/ConsumerRecordFactory; Caused by: java.lang.ClassNotFoundException: org.apache.kafka.streams.test.ConsumerRecordFactory 1 test completed, 1 failed {noformat} So, following your lead, I added the following to build.gradle: {noformat} diff --git a/build.gradle b/build.gradle index 33fa7a750..ca8b0ed4b 100644 --- a/build.gradle +++ b/build.gradle @@ -929,6 +929,8 @@ project(':streams') { testCompile libs.easymock testCompile libs.bcpkix + // testRuntimeOnly makes dependencies available at test runtime, while preventing the cyclic dependency as noted above + testRuntimeOnly project(':streams:test-utils') testRuntime libs.slf4jlog4j } {noformat} Like before I don't know if this is the best solution, but at least it keeps the tests from failing. I'll be creating a [WIP] pull request with my current changes shortly, to test on Jenkins and get some early feedback as advised. Hopefully the tests all pass there too. I'm close to finishing the migration to the new test driver, but I'm still missing about 8 classes. There are also a few tests which still don't work well with the new TopologyTestDriver (I'm still relying on KStreamTestDriver for those), and I'll have to figure what's wrong before I'm done.
19. h314to opened a new pull request #4832: KAFKA-6474: [WIP] Rewrite tests to use new public TopologyTestDriver URL: https://github.com/apache/kafka/pull/4832 **Please do not merge yet. This is still work in progress.** * Remove ProcessorTopologyTestDriver from TopologyTest * Fix ProcessorTopologyTest * Remove ProcessorTopologyTestDriver and InternalTopologyAccessor * Partially refactored StreamsBuilderTest but missing one test * Refactor KStreamBuilderTest * Refactor AbstractStreamTest * Further cleanup of AbstractStreamTest * Refactor GlobalKTableJoinsTest * Refactor InternalStreamsBuilderTest * Fix circular dependency in build.gradle * Refactor KGroupedStreamImplTest * Partial modifications to KGroupedTableImplTest * Refactor KGroupedTableImplTest * Refactor KStreamBranchTest * Refactor KStreamFilterTest * Refactor KStreamFlatMapTest KStreamFlatMapValuesTest * Refactor KStreamForeachTest * Refactor KStreamGlobalKTableJoinTest * Refactor KStreamGlobalKTableLeftJoinTest * Refactor KStreamImplTest * Refactor KStreamImplTest * Refactor KStreamKStreamJoinTest * Refactor KStreamKStreamLeftJoinTest * Refactor KStreamKTableJoinTest * Refactor KStreamKTableLeftJoinTest * Refactor KStreamMapTest and KStreamMapValuesTest * Refactor KStreamPeekTest and KStreamTransformTest * Refactor KStreamSelectKeyTest * Refactor KStreamTransformValuesTest * Refactor KStreamWindowAggregateTest * Add Depercation anotation to KStreamTestDriver and rollback failing tests in StreamsBuilderTest and KTableAggregateTest * Refactor KTableFilterTest * Refactor KTableForeachTest * Add getter for ProcessorTopology, and simplify tests in StreamsBuilderTest * Refactor KTableImplTest * Remove unused imports * Refactor KTableAggregateTest -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
20. Thanks for the PR. I'll review it and comment as soon as I can. -John
21. guozhangwang closed pull request #4832: KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [partial] URL: https://github.com/apache/kafka/pull/4832 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/streams/src/test/java/org/apache/kafka/streams/StreamsBuilderTest.java b/streams/src/test/java/org/apache/kafka/streams/StreamsBuilderTest.java index 4a496b8dd28..d3e01faf32b 100644 --- a/streams/src/test/java/org/apache/kafka/streams/StreamsBuilderTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/StreamsBuilderTest.java @@ -16,7 +16,9 @@ */ package org.apache.kafka.streams; +import org.apache.kafka.common.serialization.LongSerializer; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; import org.apache.kafka.common.utils.Utils; import org.apache.kafka.streams.errors.TopologyException; @@ -28,13 +30,14 @@ import org.apache.kafka.streams.processor.internals.InternalTopologyBuilder; import org.apache.kafka.streams.processor.internals.ProcessorTopology; import org.apache.kafka.streams.state.KeyValueStore; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockMapper; import org.apache.kafka.test.MockPredicate; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; import org.apache.kafka.test.TestUtils; -import org.junit.Rule; +import org.junit.After; +import org.junit.Before; import org.junit.Test; import java.util.Arrays; @@ -43,6 +46,7 @@ import java.util.HashMap; import java.util.Iterator; import java.util.Map; +import java.util.Properties; import java.util.Set; import static org.hamcrest.CoreMatchers.equalTo; @@ -54,9 +58,26 @@ public class StreamsBuilderTest { private final StreamsBuilder builder = new StreamsBuilder(); + private TopologyTestDriver driver; + private final Properties props = new Properties(); + + @Before + public void setup() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "streams-builder-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + @After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; + } @Test(expected = TopologyException.class) public void testFrom() { @@ -70,9 +91,7 @@ public void shouldAllowJoinUnmaterializedFilteredKTable() { final KTable<Bytes, String> filteredKTable = builder.<Bytes, String>table("table-topic").filter(MockPredicate.<Bytes, String>allGoodPredicate()); builder.<Bytes, String>stream("stream-topic").join(filteredKTable, MockValueJoiner.TOSTRING_JOINER); - driver.setUp(builder, TestUtils.tempDirectory()); - - ProcessorTopology topology = builder.internalTopologyBuilder.build(); + final ProcessorTopology topology = builder.internalTopologyBuilder.build(); assertThat(topology.stateStores().size(), equalTo(1));

assertThat(topology.processorConnectedStateStores("KSTREAM-JOIN-0000000005"), equalTo(Collections.singleton(topology.stateStores().get(0).name()))); @@ -85,9 +104,7 @@ public void shouldAllowJoinMaterializedFilteredKTable() { .filter(MockPredicate.<Bytes, String>allGoodPredicate(), Materialized.<Bytes, String, KeyValueStore<Bytes, byte[]>>as("store")); builder.<Bytes, String>stream("stream-topic").join(filteredKTable, MockValueJoiner.TOSTRING_JOINER); - driver.setUp(builder, TestUtils.tempDirectory()); - - ProcessorTopology topology = builder.internalTopologyBuilder.build(); + final ProcessorTopology topology = builder.internalTopologyBuilder.build(); assertThat(topology.stateStores().size(), equalTo(2)); assertThat(topology.processorConnectedStateStores("KSTREAM-JOIN-0000000005"), equalTo(Collections.singleton("store"))); @@ -99,9 +116,7 @@ public void shouldAllowJoinUnmaterializedMapValuedKTable() { final KTable<Bytes, String> mappedKTable = builder.<Bytes, String>table("table-topic").mapValues(MockMapper.<String>noOpValueMapper()); builder.<Bytes, String>stream("stream-topic").join(mappedKTable, MockValueJoiner.TOSTRING_JOINER); - driver.setUp(builder, TestUtils.tempDirectory()); - - ProcessorTopology topology = builder.internalTopologyBuilder.build(); + final ProcessorTopology topology = builder.internalTopologyBuilder.build(); assertThat(topology.stateStores().size(), equalTo(1)); assertThat(topology.processorConnectedStateStores("KSTREAM-JOIN-0000000005"), equalTo(Collections.singleton(topology.stateStores().get(0).name()))); @@ -114,9 +129,7 @@ public void shouldAllowJoinMaterializedMapValuedKTable() { .mapValues(MockMapper.<String>noOpValueMapper(), Materialized.<Bytes, String, KeyValueStore<Bytes, byte[]>>as("store")); builder.<Bytes, String>stream("stream-topic").join(mappedKTable, MockValueJoiner.TOSTRING_JOINER); - driver.setUp(builder, TestUtils.tempDirectory()); - - ProcessorTopology topology = builder.internalTopologyBuilder.build(); + final ProcessorTopology topology = builder.internalTopologyBuilder.build(); assertThat(topology.stateStores().size(), equalTo(2)); assertThat(topology.processorConnectedStateStores("KSTREAM-JOIN-0000000005"), equalTo(Collections.singleton("store"))); @@ -129,14 +142,11 @@ public void shouldAllowJoinUnmaterializedJoinedKTable() { final KTable<Bytes, String> table2 = builder.table("table-topic2"); builder.<Bytes, String>stream("stream-topic").join(table1.join(table2, MockValueJoiner.TOSTRING_JOINER), MockValueJoiner.TOSTRING_JOINER); - driver.setUp(builder, TestUtils.tempDirectory()); - - ProcessorTopology topology = builder.internalTopologyBuilder.build(); + final ProcessorTopology topology = builder.internalTopologyBuilder.build(); assertThat(topology.stateStores().size(), equalTo(2)); assertThat(topology.processorConnectedStateStores("KSTREAM-JOIN-0000000010"), equalTo(Utils.mkSet(topology.stateStores().get(0).name(), topology.stateStores().get(1).name()))); assertThat(topology.processorConnectedStateStores("KTABLE-MERGE-0000000007").isEmpty(), is(true)); - } @Test @@ -145,9 +155,7 @@ public void shouldAllowJoinMaterializedJoinedKTable() { final KTable<Bytes, String> table2 = builder.table("table-topic2"); builder.<Bytes, String>stream("stream-topic").join(table1.join(table2, MockValueJoiner.TOSTRING_JOINER, Materialized.<Bytes, String, KeyValueStore<Bytes, byte[]>>as("store")), MockValueJoiner.TOSTRING_JOINER); - driver.setUp(builder, TestUtils.tempDirectory()); - - ProcessorTopology topology = builder.internalTopologyBuilder.build(); + final ProcessorTopology topology = builder.internalTopologyBuilder.build(); assertThat(topology.stateStores().size(), equalTo(3)); assertThat(topology.processorConnectedStateStores("KSTREAM-JOIN-0000000010"), equalTo(Collections.singleton("store"))); @@ -159,9 +167,7 @@ public void shouldAllowJoinMaterializedSourceKTable() { final KTable<Bytes, String> table = builder.table("table-topic"); builder.<Bytes, String>stream("stream-topic").join(table, MockValueJoiner.TOSTRING_JOINER); - driver.setUp(builder, TestUtils.tempDirectory()); - - ProcessorTopology topology = builder.internalTopologyBuilder.build(); + final ProcessorTopology topology = builder.internalTopologyBuilder.build(); assertThat(topology.stateStores().size(), equalTo(1)); assertThat(topology.processorConnectedStateStores("KTABLE-SOURCE-0000000002"), equalTo(Collections.singleton(topology.stateStores().get(0).name()))); @@ -177,10 +183,10 @@ public void shouldProcessingFromSinkTopic() { source.process(processorSupplier); - driver.setUp(builder); - driver.setTime(0L); + driver = new TopologyTestDriver(builder.build(), props); - driver.process("topic-source", "A", "aa"); + final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + driver.pipeInput(recordFactory.create("topic-source", "A", "aa")); // no exception was thrown assertEquals(Utils.mkList("A:aa"), processorSupplier.processed); @@ -197,10 +203,10 @@ public void shouldProcessViaThroughTopic() { source.process(sourceProcessorSupplier); through.process(throughProcessorSupplier); - driver.setUp(builder); - driver.setTime(0L); + driver = new TopologyTestDriver(builder.build(), props); - driver.process("topic-source", "A", "aa"); + final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + driver.pipeInput(recordFactory.create("topic-source", "A", "aa")); assertEquals(Utils.mkList("A:aa"), sourceProcessorSupplier.processed); assertEquals(Utils.mkList("A:aa"), throughProcessorSupplier.processed); @@ -218,13 +224,13 @@ public void testMerge() { final MockProcessorSupplier<String, String> processorSupplier = new MockProcessorSupplier<>(); merged.process(processorSupplier); - driver.setUp(builder); - driver.setTime(0L); + driver = new TopologyTestDriver(builder.build(), props); - driver.process(topic1, "A", "aa"); - driver.process(topic2, "B", "bb"); - driver.process(topic2, "C", "cc"); - driver.process(topic1, "D", "dd"); + final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + driver.pipeInput(recordFactory.create(topic1, "A", "aa")); + driver.pipeInput(recordFactory.create(topic2, "B", "bb")); + driver.pipeInput(recordFactory.create(topic2, "C", "cc")); + driver.pipeInput(recordFactory.create(topic1, "D", "dd")); assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd"), processorSupplier.processed); } @@ -244,12 +250,13 @@ public void apply(final Long key, final String value) { .withValueSerde(Serdes.String())) .toStream().foreach(action); - driver.setUp(builder, TestUtils.tempDirectory()); - driver.setTime(0L); - driver.process(topic, 1L, "value1"); - driver.process(topic, 2L, "value2"); - driver.flushState(); - final KeyValueStore<Long, String> store = (KeyValueStore<Long, String>) driver.allStateStores().get("store"); + driver = new TopologyTestDriver(builder.build(), props); + + final ConsumerRecordFactory<Long, String> recordFactory = new ConsumerRecordFactory<>(new LongSerializer(), new StringSerializer()); + driver.pipeInput(recordFactory.create(topic, 1L, "value1")); + driver.pipeInput(recordFactory.create(topic, 2L, "value2")); + + final KeyValueStore<Long, String> store = driver.getKeyValueStore("store"); assertThat(store.get(1L), equalTo("value1")); assertThat(store.get(2L), equalTo("value2")); assertThat(results.get(1L), equalTo("value1")); @@ -262,12 +269,14 @@ public void shouldUseSerdesDefinedInMaterializedToConsumeGlobalTable() { builder.globalTable(topic, Materialized.<Long, String, KeyValueStore<Bytes, byte[]>>as("store") .withKeySerde(Serdes.Long()) .withValueSerde(Serdes.String())); - driver.setUp(builder, TestUtils.tempDirectory()); - driver.setTime(0L); - driver.process(topic, 1L, "value1"); - driver.process(topic, 2L, "value2"); - driver.flushState(); - final KeyValueStore<Long, String> store = (KeyValueStore<Long, String>) driver.allStateStores().get("store"); + + driver = new TopologyTestDriver(builder.build(), props); + + final ConsumerRecordFactory<Long, String> recordFactory = new ConsumerRecordFactory<>(new LongSerializer(), new StringSerializer()); + driver.pipeInput(recordFactory.create(topic, 1L, "value1")); + driver.pipeInput(recordFactory.create(topic, 2L, "value2")); + final KeyValueStore<Long, String> store = driver.getKeyValueStore("store"); + assertThat(store.get(1L), equalTo("value1")); assertThat(store.get(2L), equalTo("value2")); } @@ -295,12 +304,12 @@ public void shouldUseDefaultNodeAndStoreNames() { } @Test(expected = TopologyException.class) - public void shouldThrowExceptionWhenNoTopicPresent() throws Exception { + public void shouldThrowExceptionWhenNoTopicPresent() { builder.stream(Collections.<String>emptyList()); } @Test(expected = NullPointerException.class) - public void shouldThrowExceptionWhenTopicNamesAreNull() throws Exception { + public void shouldThrowExceptionWhenTopicNamesAreNull()

```
{ builder.stream(Arrays.<String>asList(null, null)); } diff --git a/streams/src/test/java/org/apache/kafka/streams/TopologyTest.java
b/streams/src/test/java/org/apache/kafka/streams/TopologyTest.java index 68340910c2b..eee33860532 100644 ---
a/streams/src/test/java/org/apache/kafka/streams/TopologyTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/TopologyTest.java @@ -26,7 +26,6 @@ import
org.apache.kafka.streams.state.internals.KeyValueStoreBuilder; import org.apache.kafka.test.MockProcessorSupplier; import
org.apache.kafka.test.MockStateStore; -import org.apache.kafka.test.ProcessorTopologyTestDriver; import org.apache.kafka.test.TestUtils;
import org.easymock.EasyMock; import org.junit.Test; @@ -261,7 +260,6 @@ public void
shouldThrowOnUnassignedStateStoreAccess() throws Exception { config.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG,
"host:1"); config.put(StreamsConfig.APPLICATION_ID_CONFIG, "appId"); config.put(StreamsConfig.STATE_DIR_CONFIG,
TestUtils.tempDirectory().getAbsolutePath()); - final StreamsConfig streamsConfig = new StreamsConfig(config); mockStoreBuilder();
EasyMock.expect(storeBuilder.build()).andReturn(new MockStateStore("store", false)); EasyMock.replay(storeBuilder); @@ -274,7
+272,7 @@ public void shouldThrowOnUnassignedStateStoreAccess() throws Exception { .addProcessor(badNodeName, new
LocalMockProcessorSupplier(), sourceNodeName); try { - new ProcessorTopologyTestDriver(streamsConfig,
topology.internalTopologyBuilder); + new TopologyTestDriver(topology, config); fail("Should have thrown StreamsException"); } catch
(final StreamsException e) { final String error = e.toString(); diff --git
a/streams/src/test/java/org/apache/kafka/streams/InternalTopologyAccessor.java
b/streams/src/test/java/org/apache/kafka/streams/TopologyTestDriverWrapper.java similarity index 58% rename from
streams/src/test/java/org/apache/kafka/streams/InternalTopologyAccessor.java rename to
streams/src/test/java/org/apache/kafka/streams/TopologyTestDriverWrapper.java index c3c45046cbd..fa976a80d68 100644 ---
a/streams/src/test/java/org/apache/kafka/streams/InternalTopologyAccessor.java +++
b/streams/src/test/java/org/apache/kafka/streams/TopologyTestDriverWrapper.java @@ -14,19 +14,24 @@ * See the License for the
specific language governing permissions and * limitations under the License. */ - package org.apache.kafka.streams; import
org.apache.kafka.streams.processor.internals.InternalTopologyBuilder; +import java.util.Properties; /** - * This class is meant for testing
purposes only and allows the testing of - * topologies by using the {@link org.apache.kafka.test.ProcessorTopologyTestDriver} + * This
class allows the instantiation of a {@link TopologyTestDriver} using a + * {@link InternalTopologyBuilder} by exposing a protected
constructor. + * + * It should be used only for testing, and should be removed once the deprecated + * classes {@link
org.apache.kafka.streams.kstream.KStreamBuilder} and + * {@link org.apache.kafka.streams.processor.TopologyBuilder} are removed.
*/ -public class InternalTopologyAccessor { +public class TopologyTestDriverWrapper extends TopologyTestDriver { - public static
InternalTopologyBuilder getInternalTopologyBuilder(final Topology topology) { - return topology.internalTopologyBuilder; + public
TopologyTestDriverWrapper(final InternalTopologyBuilder builder, + final Properties config) { + super(builder, config); } } diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/KStreamBuilderTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/KStreamBuilderTest.java index f9949d3c5b8..81bdb314948 100644 ---
a/streams/src/test/java/org/apache/kafka/streams/kstream/KStreamBuilderTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/KStreamBuilderTest.java @@ -18,7 +18,10 @@ import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Utils; +import
org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriverWrapper; import
org.apache.kafka.streams.errors.TopologyBuilderException; import org.apache.kafka.streams.kstream.internals.KStreamImpl; import
org.apache.kafka.streams.kstream.internals.KTableImpl; @@ -26,19 +29,21 @@ import
org.apache.kafka.streams.processor.TopologyBuilder; import org.apache.kafka.streams.processor.internals.ProcessorTopology; import
org.apache.kafka.streams.processor.internals.SourceNode; -import org.apache.kafka.test.KStreamTestDriver; +import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockMapper; import
org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockTimestampExtractor; import
org.apache.kafka.test.MockValueJoiner; +import org.apache.kafka.test.TestUtils; +import org.junit.After; import org.junit.Before; -import
org.junit.Rule; import org.junit.Test; import java.util.Collections; import java.util.HashSet; import java.util.List; import java.util.Map;
+import java.util.Properties; import java.util.Set; import java.util.regex.Pattern; @@ -55,12 +60,27 @@ private static final String
APP_ID = "app-id"; private final KStreamBuilder builder = new KStreamBuilder(); - @Rule - public final KStreamTestDriver driver =
new KStreamTestDriver(); + private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new
StringSerializer(), new StringSerializer()); + private TopologyTestDriverWrapper driver; + private final Properties props = new
Properties(); @Before - public void setUp() { + public void setup() { builder.setApplicationId(APP_ID); +
props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-builder-test"); +
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + +
@After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; } @Test(expected =
TopologyBuilderException.class) @@ -93,10 +113,8 @@ public void shouldProcessFromSinkTopic() {
source.process(processorSupplier); - driver.setUp(builder); - driver.setTime(0L); - - driver.process("topic-source", "A", "aa"); + driver =
new TopologyTestDriverWrapper(builder.internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create("topic-source", "A",
"aa")); // no exception was thrown assertEquals(Utils.mkList("A:aa"), processorSupplier.processed); @@ -113,10 +131,8 @@ public void
shouldProcessViaThroughTopic() { source.process(sourceProcessorSupplier); through.process(throughProcessorSupplier); -
driver.setUp(builder); - driver.setTime(0L); - - driver.process("topic-source", "A", "aa"); + driver = new
TopologyTestDriverWrapper(builder.internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create("topic-source", "A", "aa"));
assertEquals(Utils.mkList("A:aa"), sourceProcessorSupplier.processed); assertEquals(Utils.mkList("A:aa"),
throughProcessorSupplier.processed); @@ -147,13 +163,12 @@ public void testMerge() { final MockProcessorSupplier<String, String>
processorSupplier = new MockProcessorSupplier<>(); merged.process(processorSupplier); - driver.setUp(builder); - driver.setTime(0L); +
driver = new TopologyTestDriverWrapper(builder.internalTopologyBuilder, props); - driver.process(topic1, "A", "aa"); -
driver.process(topic2, "B", "bb"); - driver.process(topic2, "C", "cc"); - driver.process(topic1, "D", "dd"); +
driver.pipeInput(recordFactory.create(topic1, "A", "aa")); + driver.pipeInput(recordFactory.create(topic2, "B", "bb")); +
driver.pipeInput(recordFactory.create(topic2, "C", "cc")); + driver.pipeInput(recordFactory.create(topic1, "D", "dd"));
assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd"), processorSupplier.processed); } diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/AbstractStreamTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/AbstractStreamTest.java index dcfb9ba5d78..2aa07f356a0 100644 ---
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/AbstractStreamTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/AbstractStreamTest.java @@ -16,20 +16,25 @@ */ package
org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import
org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import
org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig;
+import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import
org.apache.kafka.streams.kstream.ValueTransformerSupplier; import
org.apache.kafka.streams.kstream.ValueTransformerWithKeySupplier; import org.apache.kafka.streams.processor.AbstractProcessor;
import org.apache.kafka.streams.processor.Processor; import org.apache.kafka.streams.processor.ProcessorSupplier; -import
org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import
org.apache.kafka.test.MockProcessorSupplier; -import org.junit.Rule; +import org.apache.kafka.test.TestUtils; import org.junit.Test;
+import java.util.Properties; import java.util.Random; import static org.easymock.EasyMock.createMock; @@ -40,10 +45,6 @@ public
class AbstractStreamTest { - private final String topicName = "topic"; - @Rule - public final KStreamTestDriver driver = new
KStreamTestDriver(); - @Test public void testToInternlValueTransformerSupplierSuppliesNewTransformers() { final
ValueTransformerSupplier vts = createMock(ValueTransformerSupplier.class); @@ -82,9 +83,15 @@ public void
testShouldBeExtensible() { stream.randomFilter().process(processor); - driver.setUp(builder); + final Properties props = new Properties();
+ props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "abstract-stream-test"); +
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + + final
ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new
StringSerializer()); + final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props); for (int expectedKey :
expectedKeys) { - driver.process(topicName, expectedKey, "V" + expectedKey); + driver.pipeInput(recordFactory.create(topicName,
expectedKey, "V" + expectedKey)); } assertTrue(processor.processed.size() <= expectedKeys.length); diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/GlobalKTableJoinsTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/GlobalKTableJoinsTest.java index eebb7d23226..8c50afeda6f 100644 -
-- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/GlobalKTableJoinsTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/GlobalKTableJoinsTest.java @@ -17,22 +17,25 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.Serdes; +import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig; +import
org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.ForeachAction; import
org.apache.kafka.streams.kstream.GlobalKTable; import org.apache.kafka.streams.kstream.KStream; import
org.apache.kafka.streams.kstream.KeyValueMapper; -import org.apache.kafka.test.KStreamTestDriver; +import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockValueJoiner; import
org.apache.kafka.test.TestUtils; +import org.junit.After; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; -import
java.io.File; import java.util.HashMap; import java.util.Map; +import java.util.Properties; import static org.junit.Assert.assertEquals; @@
-43,17 +46,15 @@ private final Map<String, String> results = new HashMap<>(); private final String streamTopic = "stream"; private
final String globalTopic = "global"; - private File stateDir; private GlobalKTable<String, String> global; private KStream<String, String>
stream; private KeyValueMapper<String, String, String> keyValueMapper; private ForeachAction<String, String> action; - @Rule -
public final KStreamTestDriver driver = new KStreamTestDriver(); + private TopologyTestDriver driver; + @Before public void setUp() {
- stateDir = TestUtils.tempDirectory(); final Consumed<String, String> consumed = Consumed.with(Serdes.String(), Serdes.String());
global = builder.globalTable(globalTopic, consumed); stream = builder.stream(streamTopic, consumed); @@ -71,6 +72,14 @@ public
void apply(final String key, final String value) { }; } + @After + public void cleanup() { + if (driver != null) { + driver.close(); + } +
driver = null; + } + @Test public void shouldLeftJoinWithStream() { stream @@ -100,16 +109,22 @@ public void
shouldInnerJoinWithStream() { } private void verifyJoin(final Map<String, String> expected) { - driver.setUp(builder, stateDir); -
driver.setTime(0L); + final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new
StringSerializer(), new StringSerializer()); + + final Properties props = new Properties(); +
props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "global-ktable-joins-test"); +
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + + driver = new
TopologyTestDriver(builder.build(), props); + // write some data to the global table - driver.process(globalTopic, "a", "A"); -
driver.process(globalTopic, "b", "B"); + driver.pipeInput(recordFactory.create(globalTopic, "a", "A")); +
driver.pipeInput(recordFactory.create(globalTopic, "b", "B")); //write some data to the stream - driver.process(streamTopic, "1", "a"); -
driver.process(streamTopic, "2", "b"); - driver.process(streamTopic, "3", "c"); - driver.flushState(); +
driver.pipeInput(recordFactory.create(streamTopic, "1", "a")); + driver.pipeInput(recordFactory.create(streamTopic, "2", "b")); +
driver.pipeInput(recordFactory.create(streamTopic, "3", "c")); assertEquals(expected, results); } diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/InternalStreamsBuilderTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/InternalStreamsBuilderTest.java index b9ba6089497..76ca495c232
100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/InternalStreamsBuilderTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/InternalStreamsBuilderTest.java @@ -30,11 +30,9 @@ import
org.apache.kafka.streams.processor.internals.InternalTopologyBuilder; import
org.apache.kafka.streams.processor.internals.ProcessorTopology; import org.apache.kafka.streams.state.KeyValueStore; -import
org.apache.kafka.test.KStreamTestDriver; import org.apache.kafka.test.MockMapper; import
org.apache.kafka.test.MockTimestampExtractor; import org.apache.kafka.test.MockValueJoiner; -import org.junit.After; import
org.junit.Before; import org.junit.Test; @@ -58,8 +56,6 @@ private static final String APP_ID = "app-id"; private final
InternalStreamsBuilder builder = new InternalStreamsBuilder(new InternalTopologyBuilder()); - - private KStreamTestDriver driver =
null; private final ConsumedInternal<String, String> consumed = new ConsumedInternal<>(); private final String storePrefix = "prefix-";
private MaterializedInternal<String, String, KeyValueStore<Bytes, byte[]>> materialized @@ -70,14 +66,6 @@ public void setUp() {
builder.internalTopologyBuilder.setApplicationId(APP_ID); } - @After - public void cleanup() { - if (driver != null) { - driver.close(); - } -
driver = null; - } - @Test public void testNewName() { assertEquals("X-0000000000", builder.newProcessorName("X-")); @@ -105,7
+93,7 @@ public void testNewStoreName() { } @Test - public void shouldHaveCorrectSourceTopicsForTableFromMergedStream()
throws Exception { + public void shouldHaveCorrectSourceTopicsForTableFromMergedStream() { final String topic1 = "topic-1"; final
String topic2 = "topic-2"; final String topic3 = "topic-3"; @@ -138,7 +126,7 @@ public boolean test(final String key, final String value) {
} @Test - public void shouldStillMaterializeSourceKTableIfMaterializedIsntQueryable() throws Exception { + public void
shouldStillMaterializeSourceKTableIfMaterializedIsntQueryable() { KTable table1 = builder.table("topic2", consumed, new
MaterializedInternal<>( @@ -158,7 +146,7 @@ public void shouldStillMaterializeSourceKTableIfMaterializedIsntQueryable() thro }
@Test - public void shouldBuildGlobalTableWithNonQueryableStoreName() throws Exception { + public void
shouldBuildGlobalTableWithNonQueryableStoreName() { final GlobalKTable<String, String> table1 = builder.globalTable( "topic2",
consumed, @@ -171,7 +159,7 @@ public void shouldBuildGlobalTableWithNonQueryableStoreName() throws Exception { } @Test -
```

```
public void shouldBuildGlobalTableWithQueryaIbleStoreName() throws Exception { + public void
shouldBuildGlobalTableWithQueryaIbleStoreName() { final GlobalKTable<String, String> table1 = builder.globalTable( "topic2",
consumed, @@ -184,7 +172,7 @@ public void shouldBuildGlobalTableWithQueryaIbleStoreName() throws Exception { } @Test -
public void shouldBuildSimpleGlobalTableTopology() throws Exception { + public void shouldBuildSimpleGlobalTableTopology() {
builder.globalTable("table", consumed, new MaterializedInternal<>( @@ -199,7 +187,7 @@ public void
shouldBuildSimpleGlobalTableTopology() throws Exception { assertEquals("globalTable", stateStores.get(0).name()); } - private void
doBuildGlobalTopologyWithAllGlobalTables() throws Exception { + private void doBuildGlobalTopologyWithAllGlobalTables() { final
ProcessorTopology topology = builder.internalTopologyBuilder.buildGlobalStateTopology(); final List<StateStore> stateStores =
topology.globalStateStores(); @@ -210,7 +198,7 @@ private void doBuildGlobalTopologyWithAllGlobalTables() throws Exception { }
@Test - public void shouldBuildGlobalTopologyWithAllGlobalTables() throws Exception { + public void
shouldBuildGlobalTopologyWithAllGlobalTables() { builder.globalTable("table", consumed, new MaterializedInternal<>( @@ -224,7
+212,7 @@ public void shouldBuildGlobalTopologyWithAllGlobalTables() throws Exception { } @Test - public void
shouldAddGlobalTablesToEachGroup() throws Exception { + public void shouldAddGlobalTablesToEachGroup() { final String one =
"globalTable"; final String two = "globalTable2"; @@ -269,7 +257,7 @@ public String apply(final String key, final String value) { }
@Test - public void shouldMapStateStoresToCorrectSourceTopics() throws Exception { + public void
shouldMapStateStoresToCorrectSourceTopics() { final KStream<String, String> playEvents =
builder.stream(Collections.singleton("events"), consumed); final MaterializedInternal<String, String, KeyValueStore<Bytes, byte[]>>
materialized @@ -367,14 +355,14 @@ public void shouldAddRegexTopicToLatestAutoOffsetResetList() { } @Test - public void
shouldHaveNullTimestampExtractorWhenNoneSupplied() throws Exception { + public void
shouldHaveNullTimestampExtractorWhenNoneSupplied() { builder.stream(Collections.singleton("topic"), consumed); final
ProcessorTopology processorTopology = builder.internalTopologyBuilder.build(null);
assertNull(processorTopology.source("topic").getTimestampExtractor()); } @Test - public void shouldUseProvidedTimestampExtractor()
throws Exception { + public void shouldUseProvidedTimestampExtractor() { final ConsumedInternal consumed = new
ConsumedInternal<>(Consumed.with(new MockTimestampExtractor())); builder.stream(Collections.singleton("topic"), consumed); final
ProcessorTopology processorTopology = builder.internalTopologyBuilder.build(null); @@ -382,14 +370,14 @@ public void
shouldUseProvidedTimestampExtractor() throws Exception { } @Test - public void
ktableShouldHaveNullTimestampExtractorWhenNoneSupplied() throws Exception { + public void
ktableShouldHaveNullTimestampExtractorWhenNoneSupplied() { builder.table("topic", consumed, materialized); final
ProcessorTopology processorTopology = builder.internalTopologyBuilder.build(null);
assertNull(processorTopology.source("topic").getTimestampExtractor()); } @Test - public void
ktableShouldUseProvidedTimestampExtractor() throws Exception { + public void ktableShouldUseProvidedTimestampExtractor() { final
ConsumedInternal<String, String> consumed = new ConsumedInternal<>(Consumed.<String, String>with(new
MockTimestampExtractor())); builder.table("topic", consumed, materialized); final ProcessorTopology processorTopology =
builder.internalTopologyBuilder.build(null); diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedStreamImplTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedStreamImplTest.java index c74d0dce449..b9ca30f48cc
100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedStreamImplTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedStreamImplTest.java @@ -20,10 +20,13 @@ import
org.apache.kafka.common.MetricName; import org.apache.kafka.common.errors.InvalidTopicException; import
org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import
org.apache.kafka.common.utils.Bytes; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import
org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig; +import
org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.Aggregator; import
org.apache.kafka.streams.kstream.ForeachAction; import org.apache.kafka.streams.kstream.Initializer; @@ -43,13 +46,13 @@ import
org.apache.kafka.streams.state.KeyValueStore; import org.apache.kafka.streams.state.SessionStore; import
org.apache.kafka.streams.state.WindowStore; -import org.apache.kafka.test.KStreamTestDriver; +import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockAggregator; import
org.apache.kafka.test.MockInitializer; import org.apache.kafka.test.MockReducer; import org.apache.kafka.test.TestUtils; +import
org.junit.After; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; import java.util.ArrayList; @@ -57,6 +60,7 @@
import java.util.HashMap; import java.util.List; import java.util.Map; +import java.util.Properties; import static
org.apache.kafka.test.StreamsTestUtils.getMetricByName; import static org.hamcrest.CoreMatchers.equalTo; @@ -72,13 +76,30 @@
private static final String INVALID_STORE_NAME = "~foo bar~"; private final StreamsBuilder builder = new StreamsBuilder(); private
KGroupedStream<String, String> groupedStream; - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + +
private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new
StringSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); @Before public void before()
{ final KStream<String, String> stream = builder.stream(TOPIC, Consumed.with(Serdes.String(), Serdes.String())); groupedStream =
stream.groupByKey(Serialized.with(Serdes.String(), Serdes.String())); + +
props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kgrouped-stream-impl-test"); +
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + +
@After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; }
@SuppressWarnings("deprecation") @@ -203,20 +224,13 @@ public void shouldNotHaveNullStoreSupplierOnWindowedAggregate() {
} private void doAggregateSessionWindows(final Map<Windowed<String>, Integer> results) { - driver.setUp(builder,
TestUtils.tempDirectory()); - driver.setTime(10); - driver.process(TOPIC, "1", "1"); - driver.setTime(15); - driver.process(TOPIC, "2",
"2"); - driver.setTime(30); - driver.process(TOPIC, "1", "1"); - driver.setTime(70); - driver.process(TOPIC, "1", "1"); - driver.setTime(90);
- driver.process(TOPIC, "1", "1"); - driver.setTime(100); - driver.process(TOPIC, "1", "1"); - driver.flushState(); + driver = new
TopologyTestDriver(builder.build(), props); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 10)); +
driver.pipeInput(recordFactory.create(TOPIC, "2", "2", 15)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 30)); +
driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 70)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 90)); +
driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 100)); assertEquals(Integer.valueOf(2), results.get(new Windowed<>("1", new
SessionWindow(10, 30)))); assertEquals(Integer.valueOf(1), results.get(new Windowed<>("2", new SessionWindow(15, 15))));
assertEquals(Integer.valueOf(3), results.get(new Windowed<>("1", new SessionWindow(70, 100)))); @@ -284,20 +298,13 @@ public
void apply(final Windowed<String> key, final Integer value) { } private void doCountSessionWindows(final Map<Windowed<String>,
Long> results) { - driver.setUp(builder, TestUtils.tempDirectory()); - driver.setTime(10); - driver.process(TOPIC, "1", "1"); -
```

driver.setTime(15); - driver.process(TOPIC, "2", "2"); - driver.setTime(30); - driver.process(TOPIC, "1", "1"); - driver.setTime(70); - driver.process(TOPIC, "1", "1"); - driver.setTime(90); - driver.process(TOPIC, "1", "1"); - driver.setTime(100); - driver.process(TOPIC, "1", "1"); - driver.flushState(); + driver = new TopologyTestDriver(builder.build(), props); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 10)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "2", 15)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 30)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 70)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 90)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 100)); assertEquals(Long.valueOf(2), results.get(new Windowed<>("1", new SessionWindow(10, 30)))); assertEquals(Long.valueOf(1), results.get(new Windowed<>("2", new SessionWindow(15, 15)))); assertEquals(Long.valueOf(3), results.get(new Windowed<>("1", new SessionWindow(70, 100)))); @@ -334,20 +341,13 @@ public void apply(final Windowed<String> key, final Long value) { } private void doReduceSessionWindows(final Map<Windowed<String>, String> results) { - driver.setUp(builder, TestUtils.tempDirectory()); - driver.setTime(10); - driver.process(TOPIC, "1", "A"); - driver.setTime(15); - driver.process(TOPIC, "2", "Z"); - driver.setTime(30); - driver.process(TOPIC, "1", "B"); - driver.setTime(70); - driver.process(TOPIC, "1", "A"); - driver.setTime(90); - driver.process(TOPIC, "1", "B"); - driver.setTime(100); - driver.process(TOPIC, "1", "C"); - driver.flushState(); + driver = new TopologyTestDriver(builder.build(), props); + driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 10)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "Z", 15)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "B", 30)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 70)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "B", 90)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "C", 100)); assertEquals("A:B", results.get(new Windowed<>("1", new SessionWindow(10, 30)))); assertEquals("Z", results.get(new Windowed<>("2", new SessionWindow(15, 15)))); assertEquals("A:B:C", results.get(new Windowed<>("1", new SessionWindow(70, 100)))); @@ -556,8 +556,7 @@ public void shouldCountAndMaterializeResults() { processData(); - @SuppressWarnings("unchecked") final KeyValueStore<String, Long> count = - (KeyValueStore<String, Long>) driver.allStateStores().get("count"); + final KeyValueStore<String, Long> count = driver.getKeyValueStore("count"); assertThat(count.get("1"), equalTo(3L)); assertThat(count.get("2"), equalTo(1L)); @@ -571,10 +570,10 @@ public void shouldLogAndMeasureSkipsInAggregate() { processData(); LogCaptureAppender.unregister(appender); - final Map<MetricName, ? extends Metric> metrics = driver.context().metrics().metrics(); + final Map<MetricName, ? extends Metric> metrics = driver.metrics(); assertEquals(1.0, getMetricByName(metrics, "skipped-records-total", "stream-metrics").metricValue()); assertNotEquals(0.0, getMetricByName(metrics, "skipped-records-rate", "stream-metrics").metricValue()); - assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[3] value=[null] topic=[topic] partition=[-1] offset=[-1]")); + assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[3] value=[null] topic=[topic] partition=[0] offset=[6]")); } @@ -589,7 +588,7 @@ public void shouldReduceAndMaterializeResults() { processData(); - final KeyValueStore<String, String> reduced = (KeyValueStore<String, String>) driver.allStateStores().get("reduce"); + final KeyValueStore<String, String> reduced = driver.getKeyValueStore("reduce"); assertThat(reduced.get("1"), equalTo("A+C+D")); assertThat(reduced.get("2"), equalTo("B")); @@ -609,10 +608,10 @@ public void shouldLogAndMeasureSkipsInReduce() { processData(); LogCaptureAppender.unregister(appender); - final Map<MetricName, ? extends Metric> metrics = driver.context().metrics().metrics(); + final Map<MetricName, ? extends Metric> metrics = driver.metrics(); assertEquals(1.0, getMetricByName(metrics, "skipped-records-total", "stream-metrics").metricValue()); assertNotEquals(0.0, getMetricByName(metrics, "skipped-records-rate", "stream-metrics").metricValue()); - assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[3] value=[null] topic=[topic] partition=[-1] offset=[-1]")); + assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[3] value=[null] topic=[topic] partition=[0] offset=[6]")); } @@ -628,7 +627,7 @@ public void shouldAggregateAndMaterializeResults() { processData(); - final KeyValueStore<String, String> aggregate = (KeyValueStore<String, String>) driver.allStateStores().get("aggregate"); + final KeyValueStore<String, String> aggregate = driver.getKeyValueStore("aggregate"); assertThat(aggregate.get("1"), equalTo("0+A+C+D")); assertThat(aggregate.get("2"), equalTo("0+B")); @@ -658,29 +657,25 @@ public void apply(final String key, final String value) { } private void processData() { - driver.setUp(builder, TestUtils.tempDirectory(), Serdes.String(), Serdes.String(), 0); - driver.setTime(0); - driver.process(TOPIC, "1", "A"); - driver.process(TOPIC, "2", "B"); - driver.process(TOPIC, "1", "C"); - driver.process(TOPIC, "1", "D"); - driver.process(TOPIC, "3", "E"); - driver.process(TOPIC, "3", "F"); - driver.process(TOPIC, "3", null); - driver.flushState(); + driver = new TopologyTestDriver(builder.build(), props); + driver.pipeInput(recordFactory.create(TOPIC, "1", "A")); + driver.pipeInput(recordFactory.create(TOPIC, "2", "B")); + driver.pipeInput(recordFactory.create(TOPIC, "1", "C")); + driver.pipeInput(recordFactory.create(TOPIC, "1", "D")); + driver.pipeInput(recordFactory.create(TOPIC, "3", "E")); + driver.pipeInput(recordFactory.create(TOPIC, "3", "F")); + driver.pipeInput(recordFactory.create(TOPIC, "3", null)); } private void doCountWindowed(final List<KeyValue<Windowed<String>, Long>> results) { - driver.setUp(builder, TestUtils.tempDirectory(), 0); - driver.setTime(0); - driver.process(TOPIC, "1", "A"); - driver.process(TOPIC, "2", "B"); - driver.process(TOPIC, "3", "C"); - driver.setTime(500); - driver.process(TOPIC, "1", "A"); - driver.process(TOPIC, "1", "A"); - driver.process(TOPIC, "2", "B"); - driver.process(TOPIC, "2", "B"); + driver = new TopologyTestDriver(builder.build(), props); + driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 0)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "B", 0)); + driver.pipeInput(recordFactory.create(TOPIC, "3", "C", 0)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 500)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 500)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "B", 500)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "B", 500)); assertThat(results, equalTo(Arrays.asList(KeyValue.pair(new Windowed<>("1", new TimeWindow(0, 500)), 1L), KeyValue.pair(new Windowed<>("2", new TimeWindow(0, 500)), 1L), diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedTableImplTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedTableImplTest.java index 3bbd7e2fa50..742f3496579 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedTableImplTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedTableImplTest.java @@ -17,11 +17,15 @@ package org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.errors.InvalidTopicException; +import org.apache.kafka.common.serialization.DoubleSerializer; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.ForeachAction; import org.apache.kafka.streams.kstream.KGroupedTable; import org.apache.kafka.streams.kstream.KTable; @@ -30,18 +34,19 @@ import org.apache.kafka.streams.kstream.Serialized; import org.apache.kafka.streams.processor.StateStoreSupplier; import org.apache.kafka.streams.state.KeyValueStore; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockAggregator; import org.apache.kafka.test.MockInitializer; import org.apache.kafka.test.MockMapper; import org.apache.kafka.test.MockReducer; import org.apache.kafka.test.TestUtils; +import org.junit.After; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; import java.util.HashMap; import java.util.Map; +import java.util.Properties; import static org.hamcrest.CoreMatchers.equalTo; import static org.hamcrest.MatcherAssert.assertThat; @@ -55,14 +60,29 @@ private final StreamsBuilder builder = new StreamsBuilder(); private static final String INVALID_STORE_NAME = "~foo bar~"; private KGroupedTable<String, String> groupedTable; - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private

```
TopologyTestDriver driver; + private final Properties props = new Properties(); private final String topic = "input"; @Before public void
before() { groupedTable = builder.table("blah", Consumed.with(Serdes.String(), Serdes.String())) .groupBy(MockMapper.<String,
String>selectValueKeyValueMapper()); + + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kgrouped-table-impl-test");
+ props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + } + +
@After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; } @Test @@ -122,6 +142,7
@@ public void shouldNotAllowNullStoreSupplierOnReduce() { private void doShouldReduce(final KTable<String, Integer> reduced,
final String topic) { final Map<String, Integer> results = new HashMap<>(); + final ConsumerRecordFactory<String, Double>
recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new DoubleSerializer()); reduced.foreach(new
ForeachAction<String, Integer>() { @Override public void apply(final String key, final Integer value) { @@ -129,20 +150,17 @@ public
void apply(final String key, final Integer value) { } }); - driver.setUp(builder, TestUtils.tempDirectory(), Serdes.String(), Serdes.Integer());
- driver.setTime(10L); - driver.process(topic, "A", 1.1); - driver.process(topic, "B", 2.2); - driver.flushState(); + driver = new
TopologyTestDriver(builder.build(), props); + driver.pipeInput(recordFactory.create(topic, "A", 1.1, 10)); +
driver.pipeInput(recordFactory.create(topic, "B", 2.2, 10)); assertEquals(Integer.valueOf(1), results.get("A"));
assertEquals(Integer.valueOf(2), results.get("B")); - driver.process(topic, "A", 2.6); - driver.process(topic, "B", 1.3); - driver.process(topic,
"A", 5.7); - driver.process(topic, "B", 6.2); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic, "A", 2.6, 10)); +
driver.pipeInput(recordFactory.create(topic, "B", 1.3, 10)); + driver.pipeInput(recordFactory.create(topic, "A", 5.7, 10)); +
driver.pipeInput(recordFactory.create(topic, "B", 6.2, 10)); assertEquals(Integer.valueOf(5), results.get("A"));
assertEquals(Integer.valueOf(6), results.get("B")); @@ -212,7 +230,7 @@ public void shouldReduceAndMaterializeResults() {
.withValueSerde(Serdes.Integer())); doShouldReduce(reduced, topic); - final KeyValueStore<String, Integer> reduce =
(KeyValueStore<String, Integer>) driver.allStateStores().get("reduce"); + final KeyValueStore<String, Integer> reduce =
(KeyValueStore<String, Integer>) driver.getStateStore("reduce"); assertThat(reduce.get("A"), equalTo(5)); assertThat(reduce.get("B"),
equalTo(6)); } @@ -229,7 +247,7 @@ public void shouldCountAndMaterializeResults() { .withValueSerde(Serdes.Long()));
processData(topic); - final KeyValueStore<String, Long> counts = (KeyValueStore<String, Long>) driver.allStateStores().get("count"); +
final KeyValueStore<String, Long> counts = driver.getKeyValueStore("count"); assertThat(counts.get("1"), equalTo(3L));
assertThat(counts.get("2"), equalTo(2L)); } @@ -249,7 +267,7 @@ public void shouldAggregateAndMaterializeResults() {
.withKeySerde(Serdes.String())); processData(topic); - final KeyValueStore<String, String> aggregate = (KeyValueStore<String, String>)
driver.allStateStores().get("aggregate"); + final KeyValueStore<String, String> aggregate = (KeyValueStore<String, String>)
driver.getStateStore("aggregate"); assertThat(aggregate.get("1"), equalTo("0+1+1+1")); assertThat(aggregate.get("2"), equalTo("0+2+2"));
} @@ -310,13 +328,12 @@ public void shouldThrowNullPointerOnAggregateWhenMaterializedIsNull() { } private void
processData(final String topic) { - driver.setUp(builder, TestUtils.tempDirectory(), Serdes.String(), Serdes.Integer()); - driver.setTime(0L);
- driver.process(topic, "A", "1"); - driver.process(topic, "B", "1"); - driver.process(topic, "C", "1"); - driver.process(topic, "D", "2"); -
driver.process(topic, "E", "2"); - driver.flushState(); + driver = new TopologyTestDriver(builder.build(), props); + final
ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new
StringSerializer()); + driver.pipeInput(recordFactory.create(topic, "A", "1")); + driver.pipeInput(recordFactory.create(topic, "B", "1")); +
driver.pipeInput(recordFactory.create(topic, "C", "1")); + driver.pipeInput(recordFactory.create(topic, "D", "2")); +
driver.pipeInput(recordFactory.create(topic, "E", "2")); } } diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamBranchTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamBranchTest.java index 702631a93e7..a70bc37942e 100644 ---
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamBranchTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamBranchTest.java @@ -16,26 +16,51 @@ */ package
org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import
org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import
org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig;
+import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import
org.apache.kafka.streams.kstream.Predicate; -import org.apache.kafka.test.KStreamTestDriver; +import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.junit.Rule;
+import org.apache.kafka.test.TestUtils; +import org.junit.After; +import org.junit.Before; import org.junit.Test; import
java.lang.reflect.Array; +import java.util.Properties; import static org.junit.Assert.assertEquals; public class KStreamBranchTest { -
private String topicName = "topic"; + private final String topicName = "topic"; + private final ConsumerRecordFactory<Integer, String>
recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); + private TopologyTestDriver driver; +
private final Properties props = new Properties(); + + @Before + public void before() { +
props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-branch-test"); +
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } - @Rule -
public final KStreamTestDriver driver = new KStreamTestDriver(); + @After + public void cleanup() { + props.clear(); + if (driver !=
null) { + driver.close(); + } + driver = null; + } @SuppressWarnings("unchecked") @Test @@ -78,9 +103,9 @@ public boolean
test(Integer key, String value) { branches[i].process(processors[i]); } - driver.setUp(builder); + driver = new
TopologyTestDriver(builder.build(), props); for (int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, "V" +
expectedKey); + driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); } assertEquals(3,
processors[0].processed.size()); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFilterTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFilterTest.java index f1a6152852f..a67d688e278 100644 ---
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFilterTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFilterTest.java @@ -16,25 +16,51 @@ */ package
org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import
org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import
org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig;
+import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import
org.apache.kafka.streams.kstream.Predicate; -import org.apache.kafka.test.KStreamTestDriver; +import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.junit.Rule;
+import org.apache.kafka.test.TestUtils; +import org.junit.After; +import org.junit.Before; import org.junit.Test; +import
java.util.Properties; + import static org.junit.Assert.assertEquals; public class KStreamFilterTest { - private String topicName = "topic"; - -
```

```
@Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private final String topicName = "topic"; + private final
ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new
StringSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); + + @Before + public void
before() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-filter-test"); +
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + @After
+ public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; + } + private Predicate<Integer,
String> isMultipleOfThree = new Predicate<Integer, String>() { @Override public boolean test(Integer key, String value) { @@ -54,9
+80,9 @@ public void testFilter() { stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.String()));
stream.filter(isMultipleOfThree).process(processor); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(), props);
for (int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, "V" + expectedKey); +
driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); } assertEquals(2, processor.processed.size()); @@
-74,9 +100,9 @@ public void testFilterNot() { stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.String()));
stream.filterNot(isMultipleOfThree).process(processor); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(),
props); for (int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, "V" + expectedKey); +
driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); } assertEquals(5, processor.processed.size()); diff --
git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapTest.java index 59dad4706d4..e414218898d 100644 --
- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapTest.java @@ -16,27 +16,52 @@ */ package
org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import
org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import
org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder;
+import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KeyValueMapper; -import
org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import
org.apache.kafka.test.MockProcessorSupplier; -import org.junit.Rule; +import org.apache.kafka.test.TestUtils; +import org.junit.After;
+import org.junit.Before; import org.junit.Test; import java.util.ArrayList; +import java.util.Properties; import static
org.junit.Assert.assertEquals; public class KStreamFlatMapTest { private String topicName = "topic"; + private final
ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new
StringSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); - @Rule - public final
KStreamTestDriver driver = new KStreamTestDriver(); + @Before + public void before() { +
props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-flat-map-test"); +
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + +
@After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; + } @Test public void
testFlatMap() { @@ -63,9 +88,9 @@ public void testFlatMap() { stream = builder.stream(topicName, Consumed.with(Serdes.Integer(),
Serdes.String())); stream.flatMap(mapper).process(processor); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(),
props); for (int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, "V" + expectedKey); +
driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); } assertEquals(6, processor.processed.size()); diff --
git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapValuesTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapValuesTest.java index ecfa3aac13f..14213c99668
100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapValuesTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapValuesTest.java @@ -16,27 +16,51 @@ */ package
org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import
org.apache.kafka.common.serialization.Serdes; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig; +import
org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import
org.apache.kafka.streams.kstream.ValueMapper; import org.apache.kafka.streams.kstream.ValueMapperWithKey; -import
org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import
org.apache.kafka.test.MockProcessorSupplier; -import org.junit.Rule; +import org.apache.kafka.test.TestUtils; +import org.junit.After;
+import org.junit.Before; import org.junit.Test; import java.util.ArrayList; +import java.util.Properties; import static
org.junit.Assert.assertArrayEquals; public class KStreamFlatMapValuesTest { private String topicName = "topic"; + private final
ConsumerRecordFactory<Integer, Integer> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new
IntegerSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); + + @Before + public void
before() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-flat-map-values-test"); +
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } - @Rule -
public final KStreamTestDriver driver = new KStreamTestDriver(); + @After + public void cleanup() { + props.clear(); + if (driver !=
null) { + driver.close(); + } + driver = null; + } @Test public void testFlatMapValues() { @@ -59,9 +83,10 @@ public void
testFlatMapValues() { final MockProcessorSupplier<Integer, String> processor = new MockProcessorSupplier<>();
stream.flatMapValues(mapper).process(processor); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(), props); for
(final int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, expectedKey); + // passing the timestamp to
recordFactory.create to disambiguate the call + driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey, 0L)); }
String[] expected = {"0:v0", "0:V0", "1:v1", "1:V1", "2:v2", "2:V2", "3:v3", "3:V3"}; @@ -92,9 +117,10 @@ public void
testFlatMapValuesWithKeys() { stream.flatMapValues(mapper).process(processor); - driver.setUp(builder); + driver = new
TopologyTestDriver(builder.build(), props); for (final int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey,
expectedKey); + // passing the timestamp to recordFactory.create to disambiguate the call +
driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey, 0L)); } String[] expected = {"0:v0", "0:k0", "1:v1", "1:k1",
"2:v2", "2:k2", "3:v3", "3:k3"}; diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamForeachTest.java
```

b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamForeachTest.java index e8854fc4e41..b975c96e7d1 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamForeachTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamForeachTest.java @@ -16,32 +16,58 @@ */ package org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.ForeachAction; import org.apache.kafka.streams.kstream.KStream; -import org.apache.kafka.test.KStreamTestDriver; -import org.junit.Rule; +import org.apache.kafka.streams.test.ConsumerRecordFactory; +import org.apache.kafka.test.TestUtils; +import org.junit.After; +import org.junit.Before; import org.junit.Test; import java.util.ArrayList; import java.util.Arrays; import java.util.List; import java.util.Locale; +import java.util.Properties; import static org.junit.Assert.assertEquals; public class KStreamForeachTest { - final private String topicName = "topic"; + private final String topicName = "topic"; + private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); + private TopologyTestDriver driver; + private Properties props = new Properties(); - final private Serde<Integer> intSerde = Serdes.Integer(); - final private Serde<String> stringSerde = Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + @Before + public void before() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-foreach-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + + @After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; + } + + private final Serde<Integer> intSerde = Serdes.Integer(); + private final Serde<String> stringSerde = Serdes.String(); @Test public void testForeach() { @@ -75,9 +101,9 @@ public void apply(Integer key, String value) { stream.foreach(action); // Then - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(), props); for (KeyValue<Integer, String> record: inputRecords) { - driver.process(topicName, record.key, record.value); + driver.pipeInput(recordFactory.create(topicName, record.key, record.value)); } assertEquals(expectedRecords.size(), actualRecords.size()); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableJoinTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableJoinTest.java index d1d048bc057..2936f5fb3c4 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableJoinTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableJoinTest.java @@ -16,46 +16,46 @@ */ package org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.GlobalKTable; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KeyValueMapper; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; import org.apache.kafka.test.TestUtils; +import org.junit.After; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; -import java.io.File; import java.io.IOException; import java.util.Collection; +import java.util.Properties; import java.util.Set; import static org.junit.Assert.assertEquals; public class KStreamGlobalKTableJoinTest { - final private String streamTopic = "streamTopic"; - final private String globalTableTopic = "globalTableTopic"; - - final private Serde<Integer> intSerde = Serdes.Integer(); - final private Serde<String> stringSerde = Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); - private File stateDir = null; + private final String streamTopic = "streamTopic"; + private final String globalTableTopic = "globalTableTopic"; + private final Serde<Integer> intSerde = Serdes.Integer(); + private final Serde<String> stringSerde = Serdes.String(); + private TopologyTestDriver driver; private MockProcessorSupplier<Integer, String> processor; private final int[] expectedKeys = {0, 1, 2, 3}; private StreamsBuilder builder; @Before public void setUp() throws IOException { - stateDir = TestUtils.tempDirectory("kafka-test"); builder = new StreamsBuilder(); final KStream<Integer, String> stream; @@ -78,29 +78,46 @@ public String apply(final Integer key, final String value) { }; stream.join(table, keyMapper, MockValueJoiner.TOSTRING_JOINER).process(processor); - driver.setUp(builder, stateDir); - driver.setTime(0L); + final Properties props = new Properties(); + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-global-ktable-join-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + + driver = new TopologyTestDriver(builder.build(), props); + } + + @After + public void cleanup() { + if (driver != null) { + driver.close(); + } + driver = null; } private void pushToStream(final int messageCount, final String valuePrefix, final boolean includeForeignKey) { + final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); for (int i = 0; i < messageCount; i++) { String value = valuePrefix + expectedKeys[i]; if (includeForeignKey) { value = value + ",FKey" + expectedKeys[i]; } - driver.process(streamTopic, expectedKeys[i], value); + driver.pipeInput(recordFactory.create(streamTopic, expectedKeys[i], value)); } } private void pushToGlobalTable(final int messageCount, final String valuePrefix) { + final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); for (int i = 0; i < messageCount; i++) { - driver.process(globalTableTopic, "FKey" + expectedKeys[i], valuePrefix + expectedKeys[i]); + driver.pipeInput(recordFactory.create(globalTableTopic, "FKey" + expectedKeys[i], valuePrefix + expectedKeys[i])); } } private void pushNullValueToGlobalTable(final int messageCount) { + final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); for (int i = 0; i < messageCount; i++) { - driver.process(globalTableTopic, "FKey" + expectedKeys[i], null); + driver.pipeInput(recordFactory.create(globalTableTopic, "FKey" + expectedKeys[i], null)); } } diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableLeftJoinTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableLeftJoinTest.java index 8b7dd427387..88821130d36 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableLeftJoinTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableLeftJoinTest.java @@ -16,25 +16,28 @@ */ package org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.GlobalKTable; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KeyValueMapper; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; import org.apache.kafka.test.TestUtils; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; -import java.io.File; import java.io.IOException; import java.util.Collection; +import java.util.Properties; import java.util.Set; import static org.junit.Assert.assertEquals; @@ -43,19 +46,15 @@ final private String streamTopic = "streamTopic"; final private String globalTableTopic = "globalTableTopic"; - final private Serde<Integer> intSerde = Serdes.Integer(); final private Serde<String> stringSerde = Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); - private File stateDir = null; + private TopologyTestDriver driver; private MockProcessorSupplier<Integer, String> processor; private final int[] expectedKeys = {0, 1, 2, 3}; private StreamsBuilder builder; @Before public void setUp() throws IOException { - stateDir = TestUtils.tempDirectory("kafka-test"); builder = new StreamsBuilder(); final KStream<Integer, String> stream; @@ -78,29 +77,38 @@ public String apply(final Integer key, final String value) { }; stream.leftJoin(table, keyMapper, MockValueJoiner.TOSTRING_JOINER).process(processor); - driver.setUp(builder, stateDir); - driver.setTime(0L); + final Properties props = new Properties(); + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-global-ktable-left-join-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + + driver = new TopologyTestDriver(builder.build(), props); } private void pushToStream(final int messageCount, final String valuePrefix, final boolean includeForeignKey) { + final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); for (int i = 0; i < messageCount; i++) { String value = valuePrefix + expectedKeys[i]; if (includeForeignKey) { value = value + ",FKey" + expectedKeys[i]; } - driver.process(streamTopic, expectedKeys[i], value); + driver.pipeInput(recordFactory.create(streamTopic, expectedKeys[i], value)); } } private void pushToGlobalTable(final int messageCount, final String valuePrefix) { + final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); for (int i = 0; i < messageCount; i++) { - driver.process(globalTableTopic, "FKey" + expectedKeys[i], valuePrefix + expectedKeys[i]); + driver.pipeInput(recordFactory.create(globalTableTopic, "FKey" + expectedKeys[i], valuePrefix + expectedKeys[i])); } } private void pushNullValueToGlobalTable(final int messageCount) { + final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); for (int i = 0; i < messageCount; i++) { - driver.process(globalTableTopic, "FKey" + expectedKeys[i], null); + driver.pipeInput(recordFactory.create(globalTableTopic, "FKey" + expectedKeys[i], null)); } } diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamImplTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamImplTest.java index ef65bb32b36..f397246c7b6 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamImplTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamImplTest.java @@ -18,11 +18,14 @@ import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Utils; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.errors.TopologyException; import org.apache.kafka.streams.kstream.GlobalKTable; import org.apache.kafka.streams.kstream.JoinWindows; @@ -42,16 +45,18 @@ import org.apache.kafka.streams.processor.FailOnInvalidTimestamp; import org.apache.kafka.streams.processor.internals.ProcessorTopology; import org.apache.kafka.streams.processor.internals.SourceNode; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockMapper; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; +import org.apache.kafka.test.TestUtils; +import org.junit.After; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; import java.util.Arrays; import java.util.Collections; +import java.util.Properties; import java.util.concurrent.TimeUnit; import java.util.regex.Pattern; @@ -71,13 +76,29 @@ private StreamsBuilder builder; private final Consumed<String, String> consumed = Consumed.with(stringSerde, stringSerde); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); @Before public void before() { builder = new StreamsBuilder(); testStream = builder.stream("source"); + + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-impl-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + + @After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; } @Test @@ -204,8 +225,8 @@ public void shouldSendDataThroughTopicUsingProduced() { final MockProcessorSupplier<String, String> processorSupplier = new MockProcessorSupplier<>(); stream.through("through-topic", Produced.with(stringSerde, stringSerde)).process(processorSupplier); - driver.setUp(builder); - driver.process(input, "a", "b"); + driver = new TopologyTestDriver(builder.build(), props); + driver.pipeInput(recordFactory.create(input, "a", "b")); assertThat(processorSupplier.processed, equalTo(Collections.singletonList("a:b"))); } @@ -218,8 +239,8 @@ public void shouldSendDataToTopicUsingProduced() { stream.to("to-topic", Produced.with(stringSerde, stringSerde)); builder.stream("to-topic", consumed).process(processorSupplier); - driver.setUp(builder); - driver.process(input, "e", "f"); + driver = new TopologyTestDriver(builder.build(), props); + driver.pipeInput(recordFactory.create(input, "e", "f")); assertThat(processorSupplier.processed, equalTo(Collections.singletonList("e:f"))); } @@ -501,13 +522,12 @@ public void shouldMergeTwoStreams() { final MockProcessorSupplier<String, String> processorSupplier = new MockProcessorSupplier<>(); merged.process(processorSupplier); - driver.setUp(builder); - driver.setTime(0L); + driver = new TopologyTestDriver(builder.build(), props); - driver.process(topic1, "A", "aa"); - driver.process(topic2, "B", "bb"); - driver.process(topic2, "C", "cc"); - driver.process(topic1, "D", "dd"); + driver.pipeInput(recordFactory.create(topic1, "A", "aa")); + driver.pipeInput(recordFactory.create(topic2, "B", "bb")); + driver.pipeInput(recordFactory.create(topic2, "C", "cc")); + driver.pipeInput(recordFactory.create(topic1, "D", "dd")); assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd"), processorSupplier.processed); } @@ -528,17 +548,16 @@ public void shouldMergeMultipleStreams() { final MockProcessorSupplier<String, String> processorSupplier = new MockProcessorSupplier<>(); merged.process(processorSupplier); - driver.setUp(builder); - driver.setTime(0L); + driver = new TopologyTestDriver(builder.build(), props); - driver.process(topic1, "A", "aa"); - driver.process(topic2, "B", "bb"); - driver.process(topic3, "C", "cc"); - driver.process(topic4, "D", "dd"); - driver.process(topic4, "E", "ee"); - driver.process(topic3, "F", "ff"); - driver.process(topic2, "G", "gg"); - driver.process(topic1, "H", "hh"); + driver.pipeInput(recordFactory.create(topic1, "A", "aa")); + driver.pipeInput(recordFactory.create(topic2, "B", "bb")); + driver.pipeInput(recordFactory.create(topic3, "C", "cc")); + driver.pipeInput(recordFactory.create(topic4, "D", "dd")); + driver.pipeInput(recordFactory.create(topic4, "E", "ee")); + driver.pipeInput(recordFactory.create(topic3, "F", "ff")); + driver.pipeInput(recordFactory.create(topic2, "G", "gg")); + driver.pipeInput(recordFactory.create(topic1, "H", "hh")); assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd", "E:ee", "F:ff", "G:gg", "H:hh"), processorSupplier.processed); @@ -551,14 +570,13 @@ public void shouldProcessFromSourceThatMatchPattern() { final MockProcessorSupplier<String, String> processorSupplier = new MockProcessorSupplier<>(); pattern2Source.process(processorSupplier); - driver.setUp(builder); - driver.setTime(0L); + driver = new TopologyTestDriver(builder.build(), props); - driver.process("topic-3", "A", "aa"); - driver.process("topic-4", "B", "bb"); - driver.process("topic-5", "C", "cc"); - driver.process("topic-6", "D", "dd"); - driver.process("topic-7", "E", "ee"); + driver.pipeInput(recordFactory.create("topic-3", "A", "aa")); + driver.pipeInput(recordFactory.create("topic-4", "B", "bb")); + driver.pipeInput(recordFactory.create("topic-5", "C", "cc")); + driver.pipeInput(recordFactory.create("topic-6", "D", "dd")); + driver.pipeInput(recordFactory.create("topic-7", "E", "ee")); assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd", "E:ee"), processorSupplier.processed); @@ -576,14 +594,13 @@ public void shouldProcessFromSourcesThatMatchMultiplePattern() { final MockProcessorSupplier<String, String> processorSupplier = new MockProcessorSupplier<>(); merged.process(processorSupplier); - driver.setUp(builder); - driver.setTime(0L); + driver = new TopologyTestDriver(builder.build(), props); - driver.process("topic-3", "A", "aa"); - driver.process("topic-4", "B", "bb"); - driver.process("topic-A", "C", "cc"); - driver.process("topic-Z", "D", "dd"); - driver.process(topic3, "E", "ee"); + driver.pipeInput(recordFactory.create("topic-3", "A", "aa")); + driver.pipeInput(recordFactory.create("topic-4", "B", "bb")); + driver.pipeInput(recordFactory.create("topic-A", "C", "cc")); + driver.pipeInput(recordFactory.create("topic-Z", "D", "dd")); + driver.pipeInput(recordFactory.create(topic3, "E", "ee")); assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd", "E:ee"), processorSupplier.processed); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamJoinTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamJoinTest.java index 4d2bce5bea0..63a040acd5e 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamJoinTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamJoinTest.java @@ -16,30 +16,32 @@ */ package org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.JoinWindows; import org.apache.kafka.streams.kstream.Joined; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.ValueJoiner; -import org.apache.kafka.streams.processor.internals.ProcessorRecordContext; import org.apache.kafka.streams.processor.internals.testutil.LogCaptureAppender; -import org.apache.kafka.test.InternalMockProcessorContext; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; import org.apache.kafka.test.TestUtils; +import org.junit.After; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; -import java.io.File; import java.util.Arrays; import java.util.Collection; import java.util.HashSet; +import java.util.Properties; import java.util.Set; import static org.apache.kafka.test.StreamsTestUtils.getMetricByName; @@ -55,14 +57,27 @@ final private Serde<Integer> intSerde = Serdes.Integer(); final private Serde<String> stringSerde = Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); - private File stateDir = null; private final Consumed<Integer, String> consumed = Consumed.with(intSerde, stringSerde); + private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); @Before public void setUp() { - stateDir = TestUtils.tempDirectory("kafka-test"); + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-kstream-join-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + + @After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; } @Test @@ -71,6 +86,7 @@ public void shouldLogAndMeterOnSkippedRecordsWithNullValue() { final KStream<String, Integer> left = builder.stream("left", Consumed.with(stringSerde, intSerde)); final KStream<String, Integer> right = builder.stream("right", Consumed.with(stringSerde, intSerde)); + final ConsumerRecordFactory<String, Integer> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new IntegerSerializer()); left.join(right, @@ -85,13 +101,13 @@ public Integer apply(final Integer value1, final Integer value2) {); final LogCaptureAppender appender = LogCaptureAppender.createAndRegister(); - driver.setUp(builder, stateDir); - driver.process("left", "A", null); + driver = new TopologyTestDriver(builder.build(), props); + driver.pipeInput(recordFactory.create("left", "A", null)); LogCaptureAppender.unregister(appender); - assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[A] value=[null] topic=[left] partition=[-1] offset=[-1]")); + assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[A] value=[null] topic=[left] partition=[0] offset=[0]")); - assertEquals(1.0, getMetricByName(driver.context().metrics().metrics(), "skipped-records-total", "stream-metrics").metricValue()); + assertEquals(1.0, getMetricByName(driver.metrics(), "skipped-records-total", "stream-metrics").metricValue()); } @Test @@ -120,8 +136,7 @@ public void testJoin() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver.setUp(builder, stateDir); - driver.setTime(0L); + driver = new TopologyTestDriver(builder.build(), props); // push two items to the primary stream. the other window is empty // w1 = {} @@ -130,7 +145,7 @@ public void testJoin() { // w2 = {} for (int i = 0; i < 2; i++) { - driver.process(topic1, expectedKeys[i], "X" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); } processor.checkAndClearProcessResult(); @@ -142,7 +157,7 @@ public void testJoin() { // w2 = { 0:Y0, 1:Y1 } for (int i = 0; i < 2; i++) { - driver.process(topic2, expectedKeys[i], "Y" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i])); } processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); @@ -153,8 +168,8 @@ public void testJoin() { // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } // w2 = { 0:Y0, 1:Y1 } - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "X" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "X" + expectedKey)); } processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); @@ -165,8 +180,8 @@ public void testJoin() { // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" +

expectedKey)); } processor.checkAndClearProcessResult("0:X0+YY0", "0:X0+YY0", "1:X1+YY1", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); @@ -177,8 +192,8 @@ public void testJoin() { // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey)); } processor.checkAndClearProcessResult("0:XX0+Y0", "0:XX0+YY0", "1:XX1+Y1", "1:XX1+YY1", "2:XX2+YY2", "3:XX3+YY3"); @@ -190,7 +205,7 @@ public void testJoin() { // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3, 0:YYY0, 1:YYY1 } for (int i = 0; i < 2; i++) { - driver.process(topic2, expectedKeys[i], "YYY" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "YYY" + expectedKeys[i])); } processor.checkAndClearProcessResult("0:X0+YYY0", "0:X0+YYY0", "0:XX0+YYY0", "1:X1+YYY1", "1:X1+YYY1", "1:XX1+YYY1"); @@ -222,8 +237,7 @@ public void testOuterJoin() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver.setUp(builder, stateDir); - driver.setTime(0L); + driver = new TopologyTestDriver(builder.build(), props, 0L); // push two items to the primary stream. the other window is empty.this should produce two items // w1 = {} @@ -232,7 +246,7 @@ public void testOuterJoin() { // w2 = {} for (int i = 0; i < 2; i++) { - driver.process(topic1, expectedKeys[i], "X" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); } processor.checkAndClearProcessResult("0:X0+null", "1:X1+null"); @@ -244,7 +258,7 @@ public void testOuterJoin() { // w2 = { 0:Y0, 1:Y1 } for (int i = 0; i < 2; i++) { - driver.process(topic2, expectedKeys[i], "Y" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i])); } processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); @@ -255,8 +269,8 @@ public void testOuterJoin() { // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } // w2 = { 0:Y0, 1:Y1 } - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "X" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "X" + expectedKey)); } processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1", "2:X2+null", "3:X3+null"); @@ -267,8 +281,8 @@ public void testOuterJoin() { // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey)); } processor.checkAndClearProcessResult("0:X0+YY0", "0:X0+YY0", "1:X1+YY1", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); @@ -279,8 +293,8 @@ public void testOuterJoin() { // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey)); } processor.checkAndClearProcessResult("0:XX0+Y0", "0:XX0+YY0", "1:XX1+Y1", "1:XX1+YY1", "2:XX2+YY2", "3:XX3+YY3"); @@ -292,7 +306,7 @@ public void testOuterJoin() { // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3, 0:YYY0, 1:YYY1 } for (int i = 0; i < 2; i++) { - driver.process(topic2, expectedKeys[i], "YYY" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "YYY" + expectedKeys[i])); } processor.checkAndClearProcessResult("0:X0+YYY0", "0:X0+YYY0", "0:XX0+YYY0", "1:X1+YYY1", "1:X1+YYY1", "1:XX1+YYY1"); @@ -327,17 +341,15 @@ public void testWindowing() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver.setUp(builder, stateDir); - + driver = new TopologyTestDriver(builder.build(), props, time); // push two items to the primary stream. the other window is empty. this should produce no items. // w1 = {} // w2 = {} // --> w1 = { 0:X0, 1:X1 } // w2 = {} - setRecordContext(time, topic1); for (int i = 0; i < 2; i++) { - driver.process(topic1, expectedKeys[i], "X" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i], time)); } processor.checkAndClearProcessResult(); @@ -348,58 +360,53 @@ public void testWindowing() { // --> w1 = { 0:X0, 1:X1 } // w2 = { 0:Y0, 1:Y1 } - setRecordContext(time, topic2); for (int i = 0; i < 2; i++) { - driver.process(topic2, expectedKeys[i], "Y" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i], time)); } processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); // clear logically time = 1000L; - setRecordContext(time, topic1); for (int i = 0; i < expectedKeys.length; i++) { - setRecordContext(time + i, topic1); - driver.process(topic1, expectedKeys[i], "X" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i], time + i)); } processor.checkAndClearProcessResult(); // gradually expires items in w1 // w1 = { 0:X0, 1:X1, 2:X2, 3:X3 } - time = 1000 + 100L; - setRecordContext(time, topic2); - - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 100L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("2:X2+YY2", "3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult(); @@ -407,37 +414,36 @@ public void testWindowing() { // go back to the time before expiration time = 1000L - 100L - 1L; - setRecordContext(time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult(); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); @@ -445,8 +451,7 @@ public void testWindowing() { // clear (logically) time = 2000L; for (int i = 0; i < expectedKeys.length; i++) { - setRecordContext(time + i, topic2); -

driver.process(topic2, expectedKeys[i], "Y" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i], time + i)); } processor.checkAndClearProcessResult(); @@ -455,37 +460,36 @@ public void testWindowing() { // w2 = { 0:Y0, 1:Y1, 2:Y2, 3:Y3 } time = 2000L + 100L; - setRecordContext(time, topic1); - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); - setRecordContext(++time, topic1); - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } processor.checkAndClearProcessResult("1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); - setRecordContext(++time, topic1); - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } processor.checkAndClearProcessResult("2:XX2+Y2", "3:XX3+Y3"); - setRecordContext(++time, topic1); - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } processor.checkAndClearProcessResult("3:XX3+Y3"); - setRecordContext(++time, topic1); - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } processor.checkAndClearProcessResult(); @@ -493,37 +497,36 @@ public void testWindowing() { // go back to the time before expiration time = 2000L - 100L - 1L; - setRecordContext(time, topic1); - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } processor.checkAndClearProcessResult(); - setRecordContext(++time, topic1); - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } processor.checkAndClearProcessResult("0:XX0+Y0"); - setRecordContext(++time, topic1); - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1"); - setRecordContext(++time, topic1); - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2"); - setRecordContext(++time, topic1); - for (final int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); @@ -560,85 +563,80 @@ public void testAsymmetricWindowingAfter() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver.setUp(builder, stateDir); + driver = new TopologyTestDriver(builder.build(), props, time); for (int i = 0; i < expectedKeys.length; i++) { - setRecordContext(time + i, topic1); - driver.process(topic1, expectedKeys[i], "X" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i], time + i)); } processor.checkAndClearProcessResult(); time = 1000L - 1L; - setRecordContext(time, topic2); - - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult(); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); time = 1000 + 100L; - setRecordContext(time, topic2); - - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("2:X2+YY2", "3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult(); @@ -673,90 +671,82 @@ public void testAsymmetricWindowingBefore() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver.setUp(builder, stateDir); + driver = new TopologyTestDriver(builder.build(), props, time); for (int i = 0; i < expectedKeys.length; i++) { - setRecordContext(time + i, topic1); - driver.process(topic1, expectedKeys[i], "X" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i], time + i)); } processor.checkAndClearProcessResult(); time = 1000L - 100L - 1L; - - setRecordContext(time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult(); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); }

processor.checkAndClearProcessResult("0:X0+YY0"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - time = 1000L; - setRecordContext(time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time = 1000L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("2:X2+YY2", "3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult("3:X3+YY3"); - setRecordContext(++time, topic2); - for (final int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); } processor.checkAndClearProcessResult(); } - - private void setRecordContext(final long time, final String topic) { - ((InternalMockProcessorContext) driver.context()).setRecordContext(new ProcessorRecordContext(time, 0, 0, topic)); - } } diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamLeftJoinTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamLeftJoinTest.java index 465082b7aa7..cb1aaf16012 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamLeftJoinTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamLeftJoinTest.java @@ -16,29 +16,30 @@ */ package org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.JoinWindows; import org.apache.kafka.streams.kstream.Joined; import org.apache.kafka.streams.kstream.KStream; -import org.apache.kafka.streams.processor.internals.ProcessorRecordContext; -import org.apache.kafka.test.KStreamTestDriver; -import org.apache.kafka.test.InternalMockProcessorContext; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; import org.apache.kafka.test.TestUtils; +import org.junit.After; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; -import java.io.File; -import java.io.IOException; import java.util.Arrays; import java.util.Collection; import java.util.HashSet; +import java.util.Properties; import java.util.Set; import static org.junit.Assert.assertEquals; @@ -50,15 +51,27 @@ final private Serde<Integer> intSerde = Serdes.Integer(); final private Serde<String> stringSerde = Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); - private File stateDir = null; private final Consumed<Integer, String> consumed = Consumed.with(intSerde, stringSerde); - + private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); + private TopologyTestDriver driver; + private Properties props = new Properties(); @Before - public void setUp() throws IOException { - stateDir = TestUtils.tempDirectory("kafka-test"); + public void setUp() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-kstream-left-join-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + + @After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; } @Test @@ -87,8 +100,7 @@ public void testLeftJoin() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver.setUp(builder, stateDir, Serdes.Integer(), Serdes.String()); - driver.setTime(0L); + driver = new TopologyTestDriver(builder.build(), props, 0L); // push two items to the primary stream. the other window is empty // w1 {} @@ -97,9 +109,8 @@ public void testLeftJoin() { // --> w2 = {} for (int i = 0; i < 2; i++) { - driver.process(topic1, expectedKeys[i], "X" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); } - driver.flushState(); processor.checkAndClearProcessResult("0:X0+null", "1:X1+null"); // push two items to the other stream. this should produce two items. @@ -109,9 +120,8 @@ public void testLeftJoin() { // --> w2 = { 0:Y0, 1:Y1 } for (int i = 0; i < 2; i++) { - driver.process(topic2, expectedKeys[i], "Y" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i])); } - driver.flushState(); processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); @@ -122,9 +132,8 @@ public void testLeftJoin() { // --> w2 = { 0:Y0, 1:Y1 } for (int i = 0; i < 3; i++) { - driver.process(topic1, expectedKeys[i], "X" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); } - driver.flushState(); processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1", "2:X2+null"); // push all items to the other stream. this should produce 5 items @@ -134,9 +143,8 @@ public void testLeftJoin() { // --> w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3} for (int expectedKey : expectedKeys) { - driver.process(topic2, expectedKey, "YY" + expectedKey); + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey)); } - driver.flushState(); processor.checkAndClearProcessResult("0:X0+YY0", "0:X0+YY0", "1:X1+YY1", "1:X1+YY1", "2:X2+YY2"); // push all four items to the primary stream. this should produce six items. @@ -146,9 +154,8 @@ public void testLeftJoin() { // --> w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3} for (int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey)); } - driver.flushState(); processor.checkAndClearProcessResult("0:XX0+Y0", "0:XX0+YY0", "1:XX1+Y1", "1:XX1+YY1", "2:XX2+YY2", "3:XX3+YY3"); } @@ -178,7 +185,7 @@ public void testWindowing() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver.setUp(builder, stateDir); + driver = new TopologyTestDriver(builder.build(), props, time); // push two items to the primary stream. the other window is empty. this should produce two items // w1 = {} @@ -186,11 +193,9 @@ public void testWindowing() { // --> w1 =

```
{ 0:X0, 1:X1 } // --> w2 = {} - setRecordContext(time, topic1); for (int i = 0; i < 2; i++) { - driver.process(topic1, expectedKeys[i], "X" +
expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i], time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:X0+null", "1:X1+null"); // push two items to the other stream. this should produce no items.
@@ -199,16 +204,13 @@ public void testWindowing() { // --> w1 = { 0:X0, 1:X1 } // --> w2 = { 0:Y0, 1:Y1 } - setRecordContext(time,
topic2); for (int i = 0; i < 2; i++) { - driver.process(topic2, expectedKeys[i], "Y" + expectedKeys[i]); +
driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i], time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); // clear logically time = 1000L; - setRecordContext(time, topic2); //
push all items to the other stream. this should produce no items. // w1 = {} @@ -216,10 +218,8 @@ public void testWindowing() { // -->
w1 = {} // --> w2 = { 0:Y0, 1:Y1, 2:Y2, 3:Y3 } for (int i = 0; i < expectedKeys.length; i++) { - setRecordContext(time + i, topic2); -
driver.process(topic2, expectedKeys[i], "Y" + expectedKeys[i]); + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" +
expectedKeys[i], time + i)); } - driver.flushState(); processor.checkAndClearProcessResult(); // gradually expire items in window 2. @@
-229,81 +229,65 @@ public void testWindowing() { // --> w2 = { 0:Y0, 1:Y1, 2:Y2, 3:Y3 } time = 1000L + 100L; -
setRecordContext(time, topic1); for (int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); +
driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); - setRecordContext(++time, topic1); +
time += 1L; for (int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); +
driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); - setRecordContext(++time, topic1); +
time += 1L; for (int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); +
driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+Y2", "3:XX3+Y3"); - setRecordContext(++time, topic1);
+ time += 1L; for (int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); +
driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+null", "3:XX3+Y3"); - setRecordContext(++time, topic1);
+ time += 1L; for (int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); +
driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+null", "3:XX3+null"); // go back to the time before
expiration time = 1000L - 100L - 1L; - setRecordContext(time, topic1); for (int expectedKey : expectedKeys) { - driver.process(topic1,
expectedKey, "XX" + expectedKey); + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } -
driver.flushState(); processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+null", "3:XX3+null"); -
setRecordContext(++time, topic1); + time += 1L; for (int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" +
expectedKey); + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+null", "2:XX2+null", "3:XX3+null"); - setRecordContext(++time, topic1);
+ time += 1L; for (int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); +
driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+null", "3:XX3+null"); - setRecordContext(++time, topic1);
+ time += 1L; for (int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); +
driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+null"); - setRecordContext(++time, topic1); +
time += 1L; for (int expectedKey : expectedKeys) { - driver.process(topic1, expectedKey, "XX" + expectedKey); +
driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); } - driver.flushState();
processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); } - - private void
setRecordContext(final long time, final String topic) { - ((InternalMockProcessorContext) driver.context()).setRecordContext(new
ProcessorRecordContext(time, 0, 0, topic)); - } } diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableJoinTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableJoinTest.java index 18003852176..5b2a797b1e2
100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableJoinTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableJoinTest.java @@ -16,26 +16,29 @@ */ package
org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; +import
org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KTable; import
org.apache.kafka.streams.processor.internals.testutil.LogCaptureAppender; -import org.apache.kafka.test.KStreamTestDriver; +import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; import
org.apache.kafka.test.MockValueJoiner; import org.apache.kafka.test.TestUtils; import org.junit.Before; -import org.junit.Rule; import
org.junit.Test; -import java.io.File; import java.util.Arrays; import java.util.Collection; import java.util.HashSet; +import
java.util.Properties; import java.util.Set; import static org.apache.kafka.test.StreamsTestUtils.getMetricByName; @@ -45,24 +48,21 @@
public class KStreamKTableJoinTest { - final private String streamTopic = "streamTopic"; - final private String tableTopic = "tableTopic";
+ private final String streamTopic = "streamTopic"; + private final String tableTopic = "tableTopic"; - final private Serde<Integer>
intSerde = Serdes.Integer(); - final private Serde<String> stringSerde = Serdes.String(); - @Rule - public final KStreamTestDriver driver
= new KStreamTestDriver(); + private final Serde<Integer> intSerde = Serdes.Integer(); + private final Serde<String> stringSerde =
Serdes.String(); + private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new
IntegerSerializer(), new StringSerializer()); + private TopologyTestDriver driver; private MockProcessorSupplier<Integer, String>
processor; private final int[] expectedKeys = {0, 1, 2, 3}; private StreamsBuilder builder; @Before public void setUp() { - final File
stateDir = TestUtils.tempDirectory("kafka-test"); - builder = new StreamsBuilder(); - final KStream<Integer, String> stream; final
KTable<Integer, String> table; @@ -72,25 +72,31 @@ public void setUp() { table = builder.table(tableTopic, consumed);
stream.join(table, MockValueJoiner.TOSTRING_JOINER).process(processor); - driver.setUp(builder, stateDir); - driver.setTime(0L); +
final Properties props = new Properties(); + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-ktable-join-test");
+ props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + + driver =
new TopologyTestDriver(builder.build(), props, 0L); } private void pushToStream(final int messageCount, final String valuePrefix) { for
(int i = 0; i < messageCount; i++) { - driver.process(streamTopic, expectedKeys[i], valuePrefix + expectedKeys[i]); +
```

driver.pipeInput(recordFactory.create(streamTopic, expectedKeys[i], valuePrefix + expectedKeys[i])); } } private void pushToTable(final int messageCount, final String valuePrefix) { for (int i = 0; i < messageCount; i++) { - driver.process(tableTopic, expectedKeys[i], valuePrefix + expectedKeys[i]); + driver.pipeInput(recordFactory.create(tableTopic, expectedKeys[i], valuePrefix + expectedKeys[i])); } } private void pushNullValueToTable() { for (int i = 0; i < 2; i++) { - driver.process(tableTopic, expectedKeys[i], null); + driver.pipeInput(recordFactory.create(tableTopic, expectedKeys[i], null)); } } @@ -188,20 +194,20 @@ public void shouldClearTableEntryOnNullValueUpdates() { @Test public void shouldLogAndMeterWhenSkippingNullLeftKey() { final LogCaptureAppender appender = LogCaptureAppender.createAndRegister(); - driver.process(streamTopic, null, "A"); + driver.pipeInput(recordFactory.create(streamTopic, null, "A")); LogCaptureAppender.unregister(appender); - assertEquals(1.0, getMetricByName(driver.context().metrics().metrics(), "skipped-records-total", "stream-metrics").metricValue()); - assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[null] value=[A] topic=[streamTopic] partition=[-1] offset=[-1]")); + assertEquals(1.0, getMetricByName(driver.metrics(), "skipped-records-total", "stream-metrics").metricValue()); + assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[null] value=[A] topic=[streamTopic] partition=[0] offset=[0]")); } @Test public void shouldLogAndMeterWhenSkippingNullLeftValue() { final LogCaptureAppender appender = LogCaptureAppender.createAndRegister(); - driver.process(streamTopic, 1, null); + driver.pipeInput(recordFactory.create(streamTopic, 1, null)); LogCaptureAppender.unregister(appender); - assertEquals(1.0, getMetricByName(driver.context().metrics().metrics(), "skipped-records-total", "stream-metrics").metricValue()); - assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[1] value=[null] topic=[streamTopic] partition=[-1] offset=[-1]")); + assertEquals(1.0, getMetricByName(driver.metrics(), "skipped-records-total", "stream-metrics").metricValue()); + assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[1] value=[null] topic=[streamTopic] partition=[0] offset=[0]")); } } diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableLeftJoinTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableLeftJoinTest.java index 7507d7a4def..669f4c7712f 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableLeftJoinTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableLeftJoinTest.java @@ -16,26 +16,28 @@ */ package org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KTable; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; import org.apache.kafka.test.TestUtils; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; -import java.io.File; -import java.io.IOException; import java.util.Arrays; import java.util.Collection; import java.util.HashSet; +import java.util.Properties; import java.util.Set; import static org.junit.Assert.assertEquals; @@ -47,17 +49,14 @@ final private Serde<Integer> intSerde = Serdes.Integer(); final private Serde<String> stringSerde = Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); - private File stateDir = null; + private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); + private TopologyTestDriver driver; private MockProcessorSupplier<Integer, String> processor; private final int[] expectedKeys = {0, 1, 2, 3}; private StreamsBuilder builder; @Before - public void setUp() throws IOException { - stateDir = TestUtils.tempDirectory("kafka-test"); - + public void setUp() { builder = new StreamsBuilder(); @@ -70,25 +69,31 @@ public void setUp() throws IOException { table = builder.table(tableTopic, consumed); stream.leftJoin(table, MockValueJoiner.TOSTRING_JOINER).process(processor); - driver.setUp(builder, stateDir); - driver.setTime(0L); + final Properties props = new Properties(); + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-ktable-left-join-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + + driver = new TopologyTestDriver(builder.build(), props, 0L); } private void pushToStream(final int messageCount, final String valuePrefix) { for (int i = 0; i < messageCount; i++) { - driver.process(streamTopic, expectedKeys[i], valuePrefix + expectedKeys[i]); + driver.pipeInput(recordFactory.create(streamTopic, expectedKeys[i], valuePrefix + expectedKeys[i])); } } private void pushToTable(final int messageCount, final String valuePrefix) { for (int i = 0; i < messageCount; i++) { - driver.process(tableTopic, expectedKeys[i], valuePrefix + expectedKeys[i]); + driver.pipeInput(recordFactory.create(tableTopic, expectedKeys[i], valuePrefix + expectedKeys[i])); } } private void pushNullValueToTable(final int messageCount) { for (int i = 0; i < messageCount; i++) { - driver.process(tableTopic, expectedKeys[i], null); + driver.pipeInput(recordFactory.create(tableTopic, expectedKeys[i], null)); } } diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapTest.java index eed7d7dda40..bb222049464 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapTest.java @@ -16,18 +16,26 @@ */ package org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KeyValueMapper; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.junit.Rule; +import org.apache.kafka.test.TestUtils; +import org.junit.After; +import org.junit.Before; import org.junit.Test; +import java.util.Properties; + import static org.junit.Assert.assertEquals; public class KStreamMapTest { @@ -36,8 +44,27 @@ final private Serde<Integer> intSerde = Serdes.Integer(); final private Serde<String> stringSerde = Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); + + @Before + public void setup() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-map-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + +

@After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; + } @Test public void testMap() { @@ -59,9 +86,9 @@ public void testMap() { processor = new MockProcessorSupplier<>(); stream.map(mapper).process(processor); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(), props); for (int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, "V" + expectedKey); + driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); } assertEquals(4, processor.processed.size()); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapValuesTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapValuesTest.java index 2cedfb425bb..17a13e0b82e 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapValuesTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapValuesTest.java @@ -16,18 +16,26 @@ */ package org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.ValueMapper; import org.apache.kafka.streams.kstream.ValueMapperWithKey; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.junit.Rule; +import org.apache.kafka.test.TestUtils; +import org.junit.After; +import org.junit.Before; import org.junit.Test; +import java.util.Properties; + import static org.junit.Assert.assertArrayEquals; public class KStreamMapValuesTest { @@ -36,8 +44,27 @@ final private Serde<Integer> intSerde = Serdes.Integer(); final private Serde<String> stringSerde = Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); + + @Before + public void setup() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-map-values-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + + @After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; + } @Test public void testFlatMapValues() { @@ -58,9 +85,9 @@ public Integer apply(CharSequence value) { stream = builder.stream(topicName, Consumed.with(intSerde, stringSerde)); stream.mapValues(mapper).process(processor); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(), props); for (int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, Integer.toString(expectedKey)); + driver.pipeInput(recordFactory.create(topicName, expectedKey, Integer.toString(expectedKey))); } String[] expected = {"1:1", "10:2", "100:3", "1000:4"}; @@ -86,9 +113,9 @@ public Integer apply(final Integer readOnlyKey, final CharSequence value) { stream = builder.stream(topicName, Consumed.with(intSerde, stringSerde)); stream.mapValues(mapper).process(processor); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(), props); for (int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, Integer.toString(expectedKey)); + driver.pipeInput(recordFactory.create(topicName, expectedKey, Integer.toString(expectedKey))); } String[] expected = {"1:2", "10:12", "100:103", "1000:1004"}; diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamPeekTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamPeekTest.java index 215b0c36a39..2c6ff81b8f6 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamPeekTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamPeekTest.java @@ -16,20 +16,26 @@ */ package org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.ForeachAction; import org.apache.kafka.streams.kstream.KStream; -import org.apache.kafka.test.KStreamTestDriver; - -import org.junit.Rule; +import org.apache.kafka.streams.test.ConsumerRecordFactory; +import org.apache.kafka.test.TestUtils; +import org.junit.After; +import org.junit.Before; import org.junit.Test; import java.util.ArrayList; import java.util.List; +import java.util.Properties; import static org.junit.Assert.assertEquals; import static org.junit.Assert.fail; @@ -39,8 +45,27 @@ private final String topicName = "topic"; private final Serde<Integer> intSerd = Serdes.Integer(); private final Serde<String> stringSerd = Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); + + @Before + public void setup() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-peek-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + + @After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; + } @Test public void shouldObserveStreamElements() { @@ -49,11 +74,11 @@ public void shouldObserveStreamElements() { final List<KeyValue<Integer, String>> peekObserved = new ArrayList<>(), streamObserved = new ArrayList<>(); stream.peek(collect(peekObserved)).foreach(collect(streamObserved)); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(), props); final List<KeyValue<Integer, String>> expected = new ArrayList<>(); for (int key = 0; key < 32; key++) { final String value = "V" + key; - driver.process(topicName, key, value); + driver.pipeInput(recordFactory.create(topicName, key, value)); expected.add(new KeyValue<>(key, value)); } diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamSelectKeyTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamSelectKeyTest.java index f4f340fb6ac..0bf64523f12 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamSelectKeyTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamSelectKeyTest.java @@ -16,20 +16,27 @@ */ package org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.ForeachAction; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KeyValueMapper; -import

```
org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import
org.apache.kafka.test.MockProcessorSupplier; -import org.junit.Rule; +import org.apache.kafka.test.TestUtils; +import org.junit.After;
+import org.junit.Before; import org.junit.Test; import java.util.HashMap; import java.util.Map; +import java.util.Properties; import static
org.junit.Assert.assertEquals; @@ -39,8 +46,27 @@ final private Serde<Integer> integerSerde = Serdes.Integer(); final private
Serde<String> stringSerde = Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private final
ConsumerRecordFactory<String, Integer> recordFactory = new ConsumerRecordFactory<>(topicName, new StringSerializer(), new
IntegerSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); + + @Before + public void
setup() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-select-key-test"); +
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + } + +
@After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; + } @Test public void
testSelectKey() { @@ -68,10 +94,10 @@ public String apply(Object key, Number value) {
stream.selectKey(selector).process(processor); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(), props); for (int
expectedValue : expectedValues) { - driver.process(topicName, null, expectedValue); +
driver.pipeInput(recordFactory.create(expectedValue)); } assertEquals(3, processor.processed.size()); diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformTest.java index df3ceaf3257..aa0cf7ee7a5 100644 --
- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformTest.java @@ -16,20 +16,31 @@ */ package
org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder;
+import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.Transformer; import
org.apache.kafka.streams.kstream.TransformerSupplier; import org.apache.kafka.streams.processor.ProcessorContext; +import
org.apache.kafka.streams.processor.PunctuationType; +import org.apache.kafka.streams.processor.Punctuator; +import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.KStreamTestDriver; import
org.apache.kafka.test.MockProcessorSupplier; +import org.apache.kafka.test.TestUtils; +import org.junit.After; +import org.junit.Before;
import org.junit.Rule; import org.junit.Test; +import java.util.Properties; + import static org.junit.Assert.assertEquals; public class
KStreamTransformTest { @@ -37,8 +48,30 @@ private String topicName = "topic"; final private Serde<Integer> intSerde =
Serdes.Integer(); + private final ConsumerRecordFactory<Integer, Integer> recordFactory = new ConsumerRecordFactory<>(new
IntegerSerializer(), new IntegerSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); +
@Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + public final KStreamTestDriver kstreamDriver = new
KStreamTestDriver(); + + @Before + public void setup() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-
transform-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + } + +
@After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; + } @Test public void
testTransform() { @@ -79,13 +112,77 @@ public void close() { KStream<Integer, Integer> stream = builder.stream(topicName,
Consumed.with(intSerde, intSerde)); stream.transform(transformerSupplier).process(processor); - driver.setUp(builder); +
kstreamDriver.setUp(builder); + for (int expectedKey : expectedKeys) { + kstreamDriver.process(topicName, expectedKey, expectedKey
* 10); + } + + kstreamDriver.punctuate(2); + kstreamDriver.punctuate(3); + + assertEquals(6, processor.processed.size()); + + String[]
expected = {"2:10", "20:110", "200:1110", "2000:11110", "-1:2", "-1:3"}; + + for (int i = 0; i < expected.length; i++) { +
assertEquals(expected[i], processor.processed.get(i)); + } + } + + @Test + public void testTransformWithNewDriverAndPunctuator() { +
StreamsBuilder builder = new StreamsBuilder(); + + TransformerSupplier<Number, Number, KeyValue<Integer, Integer>>
transformerSupplier = + new TransformerSupplier<Number, Number, KeyValue<Integer, Integer>>() { + public Transformer<Number,
Number, KeyValue<Integer, Integer>> get() { + return new Transformer<Number, Number, KeyValue<Integer, Integer>>() { + + private
int total = 0; + + @Override + public void init(final ProcessorContext context) { + context.schedule(1,
PunctuationType.WALL_CLOCK_TIME, new Punctuator() { + @Override + public void punctuate(long timestamp) { +
context.forward(-1, (int) timestamp); + } + }); + } + + @Override + public KeyValue<Integer, Integer> transform(Number key, Number
value) { + total += value.intValue(); + return KeyValue.pair(key.intValue() * 2, total); + } + + @Override + public KeyValue<Integer,
Integer> punctuate(long timestamp) { + return null; + } + + @Override + public void close() { + } + }; + } + }; + + final int[]
expectedKeys = {1, 10, 100, 1000}; + + MockProcessorSupplier<Integer, Integer> processor = new MockProcessorSupplier<>(); +
KStream<Integer, Integer> stream = builder.stream(topicName, Consumed.with(intSerde, intSerde)); +
stream.transform(transformerSupplier).process(processor); + + driver = new TopologyTestDriver(builder.build(), props, 0L); for (int
expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, expectedKey * 10); +
driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey * 10, 0L)); } - driver.punctuate(2); - driver.punctuate(3); + //
This tick will yield yields the "-1:2" result + driver.advanceWallClockTime(2); + // This tick further advances the clock to 3, which leads
to the "-1:3" result + driver.advanceWallClockTime(1); assertEquals(6, processor.processed.size()); diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformValuesTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformValuesTest.java index dc0b886a93c..59a6a212f89
100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformValuesTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformValuesTest.java @@ -16,10 +16,13 @@ */ package
org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig;
+import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.errors.StreamsException; import
org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.ValueTransformer; @@ -29,11 +32,15 @@ import
org.apache.kafka.streams.processor.Processor; import org.apache.kafka.streams.processor.ProcessorContext; import
org.apache.kafka.streams.processor.To; -import org.apache.kafka.test.KStreamTestDriver; +import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.junit.Rule;
+import org.apache.kafka.test.TestUtils; +import org.junit.After; +import org.junit.Before; import org.junit.Test; +import
java.util.Properties; + import static org.junit.Assert.assertArrayEquals; import static org.junit.Assert.fail; @@ -42,8 +49,27 @@ private
```

String topicName = "topic"; final private Serde<Integer> intSerde = Serdes.Integer(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private final ConsumerRecordFactory<Integer, Integer> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new IntegerSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); + + @Before + public void setup() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-transform-values-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + } + + @After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; + } @Test public void testTransform() { @@ -85,9 +111,10 @@ public void close() { stream = builder.stream(topicName, Consumed.with(intSerde, intSerde)); stream.transformValues(valueTransformerSupplier).process(processor); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(), props); + for (int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, expectedKey * 10); + driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey * 10, 0L)); } String[] expected = {"1:10", "10:110", "100:1110", "1000:11110"}; @@ -128,9 +155,10 @@ public void close() { stream = builder.stream(topicName, Consumed.with(intSerde, intSerde)); stream.transformValues(valueTransformerSupplier).process(processor); - driver.setUp(builder); + driver = new TopologyTestDriver(builder.build(), props); + for (int expectedKey : expectedKeys) { - driver.process(topicName, expectedKey, expectedKey * 10); + driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey * 10, 0L)); } String[] expected = {"1:11", "10:121", "100:1221", "1000:12221"}; diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowAggregateTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowAggregateTest.java index 70822518889..fc31db9167b 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowAggregateTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowAggregateTest.java @@ -18,10 +18,13 @@ import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; import org.apache.kafka.common.utils.Utils; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KTable; import org.apache.kafka.streams.kstream.Materialized; @@ -29,20 +32,18 @@ import org.apache.kafka.streams.kstream.TimeWindows; import org.apache.kafka.streams.kstream.ValueJoiner; import org.apache.kafka.streams.kstream.Windowed; -import org.apache.kafka.streams.processor.internals.ProcessorRecordContext; import org.apache.kafka.streams.processor.internals.testutil.LogCaptureAppender; import org.apache.kafka.streams.state.WindowStore; -import org.apache.kafka.test.InternalMockProcessorContext; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockAggregator; import org.apache.kafka.test.MockInitializer; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.TestUtils; +import org.junit.After; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; -import java.io.File; +import java.util.Properties; import static org.apache.kafka.test.StreamsTestUtils.getMetricByName; import static org.hamcrest.CoreMatchers.hasItem; @@ -52,13 +53,26 @@ public class KStreamWindowAggregateTest { final private Serde<String> strSerde = Serdes.String(); - private File stateDir = null; - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); @Before - public void setUp() { - stateDir = TestUtils.tempDirectory("kafka-test"); + public void setup() { + props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-window-aggregate-test"); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); + } + + @After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; } @Test @@ -74,55 +88,24 @@ public void testAggBasic() { final MockProcessorSupplier<Windowed<String>, String> proc2 = new MockProcessorSupplier<>(); table2.toStream().process(proc2); - driver.setUp(builder, stateDir); - - setRecordContext(0, topic1); - driver.process(topic1, "A", "1"); - driver.flushState(); - setRecordContext(1, topic1); - driver.process(topic1, "B", "2"); - driver.flushState(); - setRecordContext(2, topic1); - driver.process(topic1, "C", "3"); - driver.flushState(); - setRecordContext(3, topic1); - driver.process(topic1, "D", "4"); - driver.flushState(); - setRecordContext(4, topic1); - driver.process(topic1, "A", "1"); - driver.flushState(); - - setRecordContext(5, topic1); - driver.process(topic1, "A", "1"); - driver.flushState(); - setRecordContext(6, topic1); - driver.process(topic1, "B", "2"); - driver.flushState(); - setRecordContext(7, topic1); - driver.process(topic1, "D", "4"); - driver.flushState(); - setRecordContext(8, topic1); - driver.process(topic1, "B", "2"); - driver.flushState(); - setRecordContext(9, topic1); - driver.process(topic1, "C", "3"); - driver.flushState(); - setRecordContext(10, topic1); - driver.process(topic1, "A", "1"); - driver.flushState(); - setRecordContext(11, topic1); - driver.process(topic1, "B", "2"); - driver.flushState(); - setRecordContext(12, topic1); - driver.flushState(); - driver.process(topic1, "D", "4"); - driver.flushState(); - setRecordContext(13, topic1); - driver.process(topic1, "B", "2"); - driver.flushState(); - setRecordContext(14, topic1); - driver.process(topic1, "C", "3"); - driver.flushState(); + driver = new TopologyTestDriver(builder.build(), props, 0L); + + driver.pipeInput(recordFactory.create(topic1, "A", "1", 0L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2", 1L)); + driver.pipeInput(recordFactory.create(topic1, "C", "3", 2L)); + driver.pipeInput(recordFactory.create(topic1, "D", "4", 3L)); + driver.pipeInput(recordFactory.create(topic1, "A", "1", 4L)); + + driver.pipeInput(recordFactory.create(topic1, "A", "1", 5L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2", 6L)); + driver.pipeInput(recordFactory.create(topic1, "D", "4", 7L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2", 8L)); + driver.pipeInput(recordFactory.create(topic1, "C", "3", 9L)); + driver.pipeInput(recordFactory.create(topic1, "A", "1", 10L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2", 11L)); + driver.pipeInput(recordFactory.create(topic1, "D", "4", 12L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2", 13L)); + driver.pipeInput(recordFactory.create(topic1, "C", "3", 14L)); assertEquals(@@ -149,10 +132,6 @@ public void testAggBasic() {); } - private void setRecordContext(final long time, @SuppressWarnings("SameParameterValue") final String topic) { - ((InternalMockProcessorContext) driver.context()).setRecordContext(new ProcessorRecordContext(time, 0, 0, topic)); - } - @Test public void testJoin() { final StreamsBuilder builder = new StreamsBuilder(); @@ -183,23 +162,13 @@ public String apply(final String p1, final String p2) { } }).toStream().process(proc3); - driver.setUp(builder, stateDir); - - setRecordContext(0, topic1); - driver.process(topic1, "A", "1"); - driver.flushState(); - setRecordContext(1, topic1); - driver.process(topic1, "B", "2"); - driver.flushState(); - setRecordContext(2, topic1); - driver.process(topic1, "C", "3"); - driver.flushState(); - setRecordContext(3, topic1); - driver.process(topic1, "D", "4"); - driver.flushState(); - setRecordContext(4, topic1); - driver.process(topic1, "A", "1"); - driver.flushState(); + driver = new TopologyTestDriver(builder.build(), props, 0L); + + driver.pipeInput(recordFactory.create(topic1, "A", "1", 0L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2", 1L)); + driver.pipeInput(recordFactory.create(topic1, "C", "3", 2L)); +

driver.pipeInput(recordFactory.create(topic1, "D", "4", 3L)); + driver.pipeInput(recordFactory.create(topic1, "A", "1", 4L));
proc1.checkAndClearProcessResult("[A@0/10]:0+1", @@ -211,21 +180,11 @@ public String apply(final String p1, final String p2) {
proc2.checkAndClearProcessResult(); proc3.checkAndClearProcessResult(); - setRecordContext(5, topic1); - driver.process(topic1, "A",
"1"); - driver.flushState(); - setRecordContext(6, topic1); - driver.process(topic1, "B", "2"); - driver.flushState(); - setRecordContext(7,
topic1); - driver.process(topic1, "D", "4"); - driver.flushState(); - setRecordContext(8, topic1); - driver.process(topic1, "B", "2"); -
driver.flushState(); - setRecordContext(9, topic1); - driver.process(topic1, "C", "3"); - driver.flushState(); +
driver.pipeInput(recordFactory.create(topic1, "A", "1", 5L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2", 6L)); +
driver.pipeInput(recordFactory.create(topic1, "D", "4", 7L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2", 8L)); +
driver.pipeInput(recordFactory.create(topic1, "C", "3", 9L)); proc1.checkAndClearProcessResult("[A@0/10]:0+1+1+1", "
[A@5/15]:0+1", @@ -237,21 +196,11 @@ public String apply(final String p1, final String p2) { proc2.checkAndClearProcessResult();
proc3.checkAndClearProcessResult(); - setRecordContext(0, topic1); - driver.process(topic2, "A", "a"); - driver.flushState(); -
setRecordContext(1, topic1); - driver.process(topic2, "B", "b"); - driver.flushState(); - setRecordContext(2, topic1); - driver.process(topic2,
"C", "c"); - driver.flushState(); - setRecordContext(3, topic1); - driver.process(topic2, "D", "d"); - driver.flushState(); -
setRecordContext(4, topic1); - driver.process(topic2, "A", "a"); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic2, "A",
"a", 0L)); + driver.pipeInput(recordFactory.create(topic2, "B", "b", 1L)); + driver.pipeInput(recordFactory.create(topic2, "C", "c", 2L)); +
driver.pipeInput(recordFactory.create(topic2, "D", "d", 3L)); + driver.pipeInput(recordFactory.create(topic2, "A", "a", 4L));
proc1.checkAndClearProcessResult(); proc2.checkAndClearProcessResult(@@ -268,21 +217,12 @@ public String apply(final String p1,
final String p2) { "[D@0/10]:0+4+4%0+d", "[A@0/10]:0+1+1+1%0+a+a"); - setRecordContext(5, topic1); - driver.process(topic2, "A",
"a"); - driver.flushState(); - setRecordContext(6, topic1); - driver.process(topic2, "B", "b"); - driver.flushState(); - setRecordContext(7,
topic1); - driver.process(topic2, "D", "d"); - driver.flushState(); - setRecordContext(8, topic1); - driver.process(topic2, "B", "b"); -
driver.flushState(); - setRecordContext(9, topic1); - driver.process(topic2, "C", "c"); - driver.flushState(); +
driver.pipeInput(recordFactory.create(topic2, "A", "a", 5L)); + driver.pipeInput(recordFactory.create(topic2, "B", "b", 6L)); +
driver.pipeInput(recordFactory.create(topic2, "D", "d", 7L)); + driver.pipeInput(recordFactory.create(topic2, "B", "b", 8L)); +
driver.pipeInput(recordFactory.create(topic2, "C", "c", 9L)); + proc1.checkAndClearProcessResult(); proc2.checkAndClearProcessResult(
"[A@0/10]:0+a+a+a", "[A@5/15]:0+a", @@ -314,15 +254,13 @@ public void shouldLogAndMeterWhenSkippingNullKey() {
Materialized.<String, String, WindowStore<Bytes, byte[]>>as("topic1-Canonicalized").withValueSerde(strSerde)); - driver.setUp(builder,
stateDir); + driver = new TopologyTestDriver(builder.build(), props, 0L); - setRecordContext(0, topic); final LogCaptureAppender
appender = LogCaptureAppender.createAndRegister(); - driver.process(topic, null, "1"); - driver.flushState(); +
driver.pipeInput(recordFactory.create(topic, null, "1")); LogCaptureAppender.unregister(appender); - assertEquals(1.0,
getMetricByName(driver.context().metrics().metrics(), "skipped-records-total", "stream-metrics").metricValue()); -
assertThat(appender.getMessages(), hasItem("Skipping record due to null key. value=[1] topic=[topic] partition=[-1] offset=[-1]")); +
assertEquals(1.0, getMetricByName(driver.metrics(), "skipped-records-total", "stream-metrics").metricValue()); +
assertThat(appender.getMessages(), hasItem("Skipping record due to null key. value=[1] topic=[topic] partition=[0] offset=[0]")); } } diff -
-git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableForeachTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableForeachTest.java index d8b3a5f3823..30c0a7a51c3 100644 ---
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableForeachTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableForeachTest.java @@ -16,27 +16,31 @@ */ package
org.apache.kafka.streams.kstream.internals; +import org.apache.kafka.common.serialization.IntegerSerializer; import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; import
org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder;
+import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.ForeachAction; import org.apache.kafka.streams.kstream.KTable; import
org.apache.kafka.streams.kstream.Materialized; import org.apache.kafka.streams.state.KeyValueStore; -import
org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import
org.apache.kafka.test.TestUtils; +import org.junit.After; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; -import
java.io.File; import java.util.ArrayList; import java.util.Arrays; import java.util.List; import java.util.Locale; +import java.util.Properties;
import static org.junit.Assert.assertEquals; @@ -44,15 +48,28 @@ public class KTableForeachTest { final private String topicName =
"topic"; - private File stateDir = null; final private Serde<Integer> intSerde = Serdes.Integer(); final private Serde<String> stringSerde =
Serdes.String(); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private final
ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new
StringSerializer()); + private TopologyTestDriver driver; + private final Properties props = new Properties(); @Before - public void
setUp() { - stateDir = TestUtils.tempDirectory("kafka-test"); + public void setup() { +
props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "ktable-foreach-test"); +
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); +
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); +
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); +
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); + } + +
@After + public void cleanup() { + props.clear(); + if (driver != null) { + driver.close(); + } + driver = null; } @Test @@ -91,11 +108,11
@@ public void apply(Integer key, String value) { table.foreach(action); // Then - driver.setUp(builder, stateDir); + driver = new
TopologyTestDriver(builder.build(), props); + for (KeyValue<Integer, String> record: inputRecords) { - driver.process(topicName,
record.key, record.value); + driver.pipeInput(recordFactory.create(topicName, record.key, record.value)); } - driver.flushState();
assertEquals(expectedRecords.size(), actualRecords.size()); for (int i = 0; i < expectedRecords.size(); i++) { diff --git
a/streams/src/test/java/org/apache/kafka/streams/processor/TopologyBuilderTest.java
b/streams/src/test/java/org/apache/kafka/streams/processor/TopologyBuilderTest.java index f67b6341f8d..f948c2c7fae 100644 ---
a/streams/src/test/java/org/apache/kafka/streams/processor/TopologyBuilderTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/processor/TopologyBuilderTest.java @@ -16,9 +16,11 @@ */ package
org.apache.kafka.streams.processor; +import org.apache.kafka.clients.consumer.ConsumerRecord; import
org.apache.kafka.common.config.TopicConfig; import org.apache.kafka.common.serialization.Serde; import
org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.streams.TopologyTestDriverWrapper; import
org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.errors.StreamsException; import
org.apache.kafka.streams.errors.TopologyBuilderException; @@ -34,7 +36,6 @@ import org.apache.kafka.test.MockProcessorSupplier;
import org.apache.kafka.test.MockStateStoreSupplier; import org.apache.kafka.test.MockTimestampExtractor; -import
org.apache.kafka.test.ProcessorTopologyTestDriver; import org.apache.kafka.test.TestUtils; import org.junit.Test; @@ -605,16 +606,12
@@ public void shouldThroughOnUnassignedStateStoreAccess() throws Exception { final TopologyBuilder builder = new
TopologyBuilder(); builder.addSource(sourceNodeName, "topic") - .addProcessor(goodNodeName, new LocalMockProcessorSupplier(), -

sourceNodeName) - .addStateStore(- Stores.create(LocalMockProcessorSupplier.STORE_NAME) - .withStringKeys().withStringValues().inMemory() - .build(), goodNodeName) - .addProcessor(badNodeName, new LocalMockProcessorSupplier(), - sourceNodeName); + .addProcessor(goodNodeName, new LocalMockProcessorSupplier(), sourceNodeName) + .addStateStore(Stores.create(LocalMockProcessorSupplier.STORE_NAME).withStringKeys().withStringValues().inMemory().build(), goodNodeName) + .addProcessor(badNodeName, new LocalMockProcessorSupplier(), sourceNodeName); try { - final ProcessorTopologyTestDriver driver = new ProcessorTopologyTestDriver(streamsConfig, builder.internalTopologyBuilder); + final TopologyTestDriverWrapper driver = new TopologyTestDriverWrapper(builder.internalTopologyBuilder, config); + driver.pipeInput(new ConsumerRecord<>("topic", 0, 0L, new byte[] {}, new byte[] {})); fail("Should have thrown StreamsException"); } catch (final StreamsException e) { final String error = e.toString(); diff --git a/streams/src/test/java/org/apache/kafka/streams/processor/internals/InternalTopologyBuilderTest.java b/streams/src/test/java/org/apache/kafka/streams/processor/internals/InternalTopologyBuilderTest.java index 901fc4b7e0a..25468c0d449 100644 --- a/streams/src/test/java/org/apache/kafka/streams/processor/internals/InternalTopologyBuilderTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/processor/internals/InternalTopologyBuilderTest.java @@ -19,6 +19,7 @@ import org.apache.kafka.common.config.TopicConfig; import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.streams.TopologyTestDriverWrapper; import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.Topology; import org.apache.kafka.streams.TopologyDescription; @@ -34,7 +35,6 @@ import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockStateStoreSupplier; import org.apache.kafka.test.MockTimestampExtractor; -import org.apache.kafka.test.ProcessorTopologyTestDriver; import org.apache.kafka.test.TestUtils; import org.junit.Test; @@ -581,7 +581,7 @@ public void shouldThrowOnUnassignedStateStoreAccess() { builder.addProcessor(badNodeName, new LocalMockProcessorSupplier(), sourceNodeName); try { - new ProcessorTopologyTestDriver(streamsConfig, builder); + new TopologyTestDriverWrapper(builder, config); fail("Should have throw StreamsException"); } catch (final StreamsException e) { final String error = e.toString(); diff --git a/streams/src/test/java/org/apache/kafka/streams/processor/internals/ProcessorTopologyTest.java b/streams/src/test/java/org/apache/kafka/streams/processor/internals/ProcessorTopologyTest.java index d07274a32a9..a80b25d0271 100644 --- a/streams/src/test/java/org/apache/kafka/streams/processor/internals/ProcessorTopologyTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/processor/internals/ProcessorTopologyTest.java @@ -24,6 +24,7 @@ import org.apache.kafka.common.serialization.StringDeserializer; import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.StreamsConfig; +import org.apache.kafka.streams.TopologyTestDriverWrapper; import org.apache.kafka.streams.processor.AbstractProcessor; import org.apache.kafka.streams.processor.Processor; import org.apache.kafka.streams.processor.ProcessorContext; @@ -36,8 +37,8 @@ import org.apache.kafka.streams.state.KeyValueIterator; import org.apache.kafka.streams.state.KeyValueStore; import org.apache.kafka.streams.state.Stores; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.ProcessorTopologyTestDriver; import org.apache.kafka.test.TestUtils; import org.junit.After; import org.junit.Before; @@ -66,26 +67,26 @@ private final TopologyBuilder builder = new TopologyBuilder(); private final MockProcessorSupplier mockProcessorSupplier = new MockProcessorSupplier(); + private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(STRING_SERIALIZER, STRING_SERIALIZER, 0L); - private ProcessorTopologyTestDriver driver; - private StreamsConfig config; + private TopologyTestDriverWrapper driver; + private final Properties props = new Properties(); @Before public void setup() { // Create a new directory in which we'll put all of the state for this test, enabling running tests in parallel ... final File localState = TestUtils.tempDirectory(); - final Properties props = new Properties(); props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "processor-topology-test"); props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); props.setProperty(StreamsConfig.STATE_DIR_CONFIG, localState.getAbsolutePath()); props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); props.setProperty(StreamsConfig.DEFAULT_TIMESTAMP_EXTRACTOR_CLASS_CONFIG, CustomTimestampExtractor.class.getName()); - this.config = new StreamsConfig(props); } @After public void cleanup() { + props.clear(); if (driver != null) { driver.close(); } @@ -122,19 +123,19 @@ public void testTopologyMetadata() { @Test public void testDrivingSimpleTopology() { - final int partition = 10; - driver = new ProcessorTopologyTestDriver(config, createSimpleTopology(partition).internalTopologyBuilder); - driver.process(INPUT_TOPIC_1, "key1", "value1", STRING_SERIALIZER, STRING_SERIALIZER); + int partition = 10; + driver = new TopologyTestDriverWrapper(createSimpleTopology(partition).internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value1")); assertNextOutputRecord(OUTPUT_TOPIC_1, "key1", "value1", partition); assertNoOutputRecord(OUTPUT_TOPIC_2); - driver.process(INPUT_TOPIC_1, "key2", "value2", STRING_SERIALIZER, STRING_SERIALIZER); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key2", "value2")); assertNextOutputRecord(OUTPUT_TOPIC_1, "key2", "value2", partition); assertNoOutputRecord(OUTPUT_TOPIC_2); - driver.process(INPUT_TOPIC_1, "key3", "value3", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key4", "value4", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key5", "value5", STRING_SERIALIZER, STRING_SERIALIZER); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key3", "value3")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key4", "value4")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key5", "value5")); assertNoOutputRecord(OUTPUT_TOPIC_2); assertNextOutputRecord(OUTPUT_TOPIC_1, "key3", "value3", partition); assertNextOutputRecord(OUTPUT_TOPIC_1, "key4", "value4", partition); @@ -144,18 +145,18 @@ public void testDrivingSimpleTopology() { @Test public void testDrivingMultiplexingTopology() { - driver = new ProcessorTopologyTestDriver(config, createMultiplexingTopology().internalTopologyBuilder); - driver.process(INPUT_TOPIC_1, "key1", "value1", STRING_SERIALIZER, STRING_SERIALIZER); + driver = new TopologyTestDriverWrapper(createMultiplexingTopology().internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value1")); assertNextOutputRecord(OUTPUT_TOPIC_1, "key1", "value1(1)"); assertNextOutputRecord(OUTPUT_TOPIC_2, "key1", "value1(2)"); - driver.process(INPUT_TOPIC_1, "key2", "value2", STRING_SERIALIZER, STRING_SERIALIZER); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key2", "value2")); assertNextOutputRecord(OUTPUT_TOPIC_1, "key2", "value2(1)"); assertNextOutputRecord(OUTPUT_TOPIC_2, "key2", "value2(2)"); - driver.process(INPUT_TOPIC_1, "key3", "value3", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key4", "value4", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key5", "value5", STRING_SERIALIZER, STRING_SERIALIZER); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key3", "value3")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key4", "value4")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key5", "value5"));

assertNextOutputRecord(OUTPUT_TOPIC_1, "key3", "value3(1)"); assertNextOutputRecord(OUTPUT_TOPIC_1, "key4", "value4(1)"); assertNextOutputRecord(OUTPUT_TOPIC_1, "key5", "value5(1)"); @@ -166,18 +167,18 @@ public void testDrivingMultiplexingTopology() { @Test public void testDrivingMultiplexByNameTopology() { - driver = new ProcessorTopologyTestDriver(config, createMultiplexByNameTopology().internalTopologyBuilder); - driver.process(INPUT_TOPIC_1, "key1", "value1", STRING_SERIALIZER, STRING_SERIALIZER); + driver = new TopologyTestDriverWrapper(createMultiplexByNameTopology().internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value1")); assertNextOutputRecord(OUTPUT_TOPIC_1, "key1", "value1(1)"); assertNextOutputRecord(OUTPUT_TOPIC_2, "key1", "value1(2)"); - driver.process(INPUT_TOPIC_1, "key2", "value2", STRING_SERIALIZER, STRING_SERIALIZER); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key2", "value2")); assertNextOutputRecord(OUTPUT_TOPIC_1, "key2", "value2(1)"); assertNextOutputRecord(OUTPUT_TOPIC_2, "key2", "value2(2)"); - driver.process(INPUT_TOPIC_1, "key3", "value3", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key4", "value4", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key5", "value5", STRING_SERIALIZER, STRING_SERIALIZER); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key3", "value3")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key4", "value4")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key5", "value5")); assertNextOutputRecord(OUTPUT_TOPIC_1, "key3", "value3(1)"); assertNextOutputRecord(OUTPUT_TOPIC_1, "key4", "value4(1)"); assertNextOutputRecord(OUTPUT_TOPIC_1, "key5", "value5(1)"); @@ -188,12 +189,12 @@ public void testDrivingMultiplexByNameTopology() { @Test public void testDrivingStatefulTopology() { - final String storeName = "entries"; - driver = new ProcessorTopologyTestDriver(config, createStatefulTopology(storeName).internalTopologyBuilder); - driver.process(INPUT_TOPIC_1, "key1", "value1", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key2", "value2", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key3", "value3", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key1", "value4", STRING_SERIALIZER, STRING_SERIALIZER); + String storeName = "entries"; + driver = new TopologyTestDriverWrapper(createStatefulTopology(storeName).internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value1")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key2", "value2")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key3", "value3")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value4")); assertNoOutputRecord(OUTPUT_TOPIC_1); final KeyValueStore<String, String> store = driver.getKeyValueStore(storeName); @@ -214,10 +215,10 @@ public void shouldDriveGlobalStore() { final TopologyBuilder topologyBuilder = this.builder .addGlobalStore(storeSupplier, global, STRING_DESERIALIZER, STRING_DESERIALIZER, topic, "processor", define(new StatefulProcessor(storeName))); - driver = new ProcessorTopologyTestDriver(config, topologyBuilder.internalTopologyBuilder); - final KeyValueStore<String, String> globalStore = (KeyValueStore<String, String>) topologyBuilder.globalStateStores().get(storeName); - driver.process(topic, "key1", "value1", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(topic, "key2", "value2", STRING_SERIALIZER, STRING_SERIALIZER); + driver = new TopologyTestDriverWrapper(topologyBuilder.internalTopologyBuilder, props); + final KeyValueStore<String, String> globalStore = (KeyValueStore<String, String>) topologyBuilder.globalStateStores().get("my-store"); + driver.pipeInput(recordFactory.create(topic, "key1", "value1")); + driver.pipeInput(recordFactory.create(topic, "key2", "value2")); assertEquals("value1", globalStore.get("key1")); assertEquals("value2", globalStore.get("key2")); } @@ -225,23 +226,23 @@ public void shouldDriveGlobalStore() { @Test public void testDrivingSimpleMultiSourceTopology() { final int partition = 10; - driver = new ProcessorTopologyTestDriver(config, createSimpleMultiSourceTopology(partition).internalTopologyBuilder); + driver = new TopologyTestDriverWrapper(createSimpleMultiSourceTopology(partition).internalTopologyBuilder, props); - driver.process(INPUT_TOPIC_1, "key1", "value1", STRING_SERIALIZER, STRING_SERIALIZER); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value1")); assertNextOutputRecord(OUTPUT_TOPIC_1, "key1", "value1", partition); assertNoOutputRecord(OUTPUT_TOPIC_2); - driver.process(INPUT_TOPIC_2, "key2", "value2", STRING_SERIALIZER, STRING_SERIALIZER); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_2, "key2", "value2")); assertNextOutputRecord(OUTPUT_TOPIC_2, "key2", "value2", partition); assertNoOutputRecord(OUTPUT_TOPIC_1); } @Test public void testDrivingForwardToSourceTopology() { - driver = new ProcessorTopologyTestDriver(config, createForwardToSourceTopology().internalTopologyBuilder); - driver.process(INPUT_TOPIC_1, "key1", "value1", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key2", "value2", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key3", "value3", STRING_SERIALIZER, STRING_SERIALIZER); + driver = new TopologyTestDriverWrapper(createForwardToSourceTopology().internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value1")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key2", "value2")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key3", "value3")); assertNextOutputRecord(OUTPUT_TOPIC_2, "key1", "value1"); assertNextOutputRecord(OUTPUT_TOPIC_2, "key2", "value2"); assertNextOutputRecord(OUTPUT_TOPIC_2, "key3", "value3"); @@ -249,10 +250,10 @@ public void testDrivingForwardToSourceTopology() { @Test public void testDrivingInternalRepartitioningTopology() { - driver = new ProcessorTopologyTestDriver(config, createInternalRepartitioningTopology().internalTopologyBuilder); - driver.process(INPUT_TOPIC_1, "key1", "value1", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key2", "value2", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key3", "value3", STRING_SERIALIZER, STRING_SERIALIZER); + driver = new TopologyTestDriverWrapper(createInternalRepartitioningTopology().internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value1")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key2", "value2")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key3", "value3")); assertNextOutputRecord(OUTPUT_TOPIC_1, "key1", "value1"); assertNextOutputRecord(OUTPUT_TOPIC_1, "key2", "value2"); assertNextOutputRecord(OUTPUT_TOPIC_1, "key3", "value3"); @@ -260,10 +261,10 @@ public void testDrivingInternalRepartitioningTopology() { @Test public void testDrivingInternalRepartitioningForwardingTimestampTopology() { - driver = new ProcessorTopologyTestDriver(config, createInternalRepartitioningWithValueTimestampTopology().internalTopologyBuilder); - driver.process(INPUT_TOPIC_1, "key1", "value1@1000", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key2", "value2@2000", STRING_SERIALIZER, STRING_SERIALIZER); - driver.process(INPUT_TOPIC_1, "key3", "value3@3000", STRING_SERIALIZER, STRING_SERIALIZER); + driver = new TopologyTestDriverWrapper(createInternalRepartitioningWithValueTimestampTopology().internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value1@1000")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key2", "value2@2000")); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key3", "value3@3000")); assertThat(driver.readOutput(OUTPUT_TOPIC_1, STRING_DESERIALIZER, STRING_DESERIALIZER), equalTo(new ProducerRecord<>(OUTPUT_TOPIC_1, null, 1000L, "key1", "value1"))); assertThat(driver.readOutput(OUTPUT_TOPIC_1, STRING_DESERIALIZER, STRING_DESERIALIZER), @@ -319,10

+320,10 @@ public void shouldRecursivelyPrintChildren() { @Test public void shouldConsiderTimeStamps() { final int partition = 10; - driver = new ProcessorTopologyTestDriver(config, createSimpleTopology(partition).internalTopologyBuilder); - driver.process(INPUT_TOPIC_1, "key1", "value1", STRING_SERIALIZER, STRING_SERIALIZER, 10L); - driver.process(INPUT_TOPIC_1, "key2", "value2", STRING_SERIALIZER, STRING_SERIALIZER, 20L); - driver.process(INPUT_TOPIC_1, "key3", "value3", STRING_SERIALIZER, STRING_SERIALIZER, 30L); + driver = new TopologyTestDriverWrapper(createSimpleTopology(partition).internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value1", 10L)); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key2", "value2", 20L)); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key3", "value3", 30L)); assertNextOutputRecord(OUTPUT_TOPIC_1, "key1", "value1", partition, 10L); assertNextOutputRecord(OUTPUT_TOPIC_1, "key2", "value2", partition, 20L); assertNextOutputRecord(OUTPUT_TOPIC_1, "key3", "value3", partition, 30L); @@ -331,10 +332,10 @@ public void shouldConsiderTimeStamps() { @Test public void shouldConsiderModifiedTimeStamps() { final int partition = 10; - driver = new ProcessorTopologyTestDriver(config, createTimestampTopology(partition).internalTopologyBuilder); - driver.process(INPUT_TOPIC_1, "key1", "value1", STRING_SERIALIZER, STRING_SERIALIZER, 10L); - driver.process(INPUT_TOPIC_1, "key2", "value2", STRING_SERIALIZER, STRING_SERIALIZER, 20L); - driver.process(INPUT_TOPIC_1, "key3", "value3", STRING_SERIALIZER, STRING_SERIALIZER, 30L); + driver = new TopologyTestDriverWrapper(createTimestampTopology(partition).internalTopologyBuilder, props); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key1", "value1", 10L)); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key2", "value2", 20L)); + driver.pipeInput(recordFactory.create(INPUT_TOPIC_1, "key3", "value3", 30L)); assertNextOutputRecord(OUTPUT_TOPIC_1, "key1", "value1", partition, 20L); assertNextOutputRecord(OUTPUT_TOPIC_1, "key2", "value2", partition, 30L); assertNextOutputRecord(OUTPUT_TOPIC_1, "key3", "value3", partition, 40L); diff --git a/streams/src/test/java/org/apache/kafka/test/KStreamTestDriver.java b/streams/src/test/java/org/apache/kafka/test/KStreamTestDriver.java index 7313414981b..1ef79da8b6f 100644 --- a/streams/src/test/java/org/apache/kafka/test/KStreamTestDriver.java +++ b/streams/src/test/java/org/apache/kafka/test/KStreamTestDriver.java @@ -44,6 +44,7 @@ import java.util.Set; import java.util.regex.Pattern; +@Deprecated public class KStreamTestDriver extends ExternalResource { private static final long DEFAULT_CACHE_SIZE_BYTES = 1024 * 1024L; diff --git a/streams/src/test/java/org/apache/kafka/test/ProcessorTopologyTestDriver.java b/streams/src/test/java/org/apache/kafka/test/ProcessorTopologyTestDriver.java deleted file mode 100644 index afd2bb20d72..00000000000 --- a/streams/src/test/java/org/apache/kafka/test/ProcessorTopologyTestDriver.java +++ /dev/null @@ -1,490 +0,0 @@ -/* - * Licensed to the Apache Software Foundation (ASF) under one or more - * contributor license agreements. See the NOTICE file distributed with - * this work for additional information regarding copyright ownership. - * The ASF licenses this file to You under the Apache License, Version 2.0 - * (the "License"); you may not use this file except in compliance with - * the License. You may obtain a copy of the License at - * - * http://www.apache.org/licenses/LICENSE-2.0 - * - * Unless required by applicable law or agreed to in writing, software - * distributed under the License is distributed on an "AS IS" BASIS, - * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. - * See the License for the specific language governing permissions and - * limitations under the License. - */ -package org.apache.kafka.test; - -import org.apache.kafka.clients.consumer.Consumer; -import org.apache.kafka.clients.consumer.ConsumerRecord; -import org.apache.kafka.clients.consumer.MockConsumer; -import org.apache.kafka.clients.consumer.OffsetResetStrategy; -import org.apache.kafka.clients.producer.MockProducer; -import org.apache.kafka.clients.producer.ProducerRecord; -import org.apache.kafka.common.PartitionInfo; -import org.apache.kafka.common.TopicPartition; -import org.apache.kafka.common.metrics.Metrics; -import org.apache.kafka.common.record.TimestampType; -import org.apache.kafka.common.serialization.ByteArraySerializer; -import org.apache.kafka.common.serialization.Deserializer; -import org.apache.kafka.common.serialization.Serializer; -import org.apache.kafka.common.utils.LogContext; -import org.apache.kafka.common.utils.MockTime; -import org.apache.kafka.common.utils.Time; -import org.apache.kafka.streams.StreamsConfig; -import org.apache.kafka.streams.Topology; -import org.apache.kafka.streams.errors.LogAndContinueExceptionHandler; -import org.apache.kafka.streams.processor.StateStore; -import org.apache.kafka.streams.processor.TaskId; -import org.apache.kafka.streams.processor.internals.GlobalProcessorContextImpl; -import org.apache.kafka.streams.processor.internals.GlobalStateManagerImpl; -import org.apache.kafka.streams.processor.internals.GlobalStateUpdateTask; -import org.apache.kafka.streams.processor.internals.InternalProcessorContext; -import org.apache.kafka.streams.processor.internals.InternalTopologyBuilder; -import org.apache.kafka.streams.processor.internals.MockStreamsMetrics; -import org.apache.kafka.streams.processor.internals.ProcessorContextImpl; -import org.apache.kafka.streams.processor.internals.ProcessorRecordContext; -import org.apache.kafka.streams.processor.internals.ProcessorTopology; -import org.apache.kafka.streams.processor.internals.StateDirectory; -import org.apache.kafka.streams.processor.internals.StoreChangelogReader; -import org.apache.kafka.streams.processor.internals.StreamTask; -import org.apache.kafka.streams.processor.internals.metrics.StreamsMetricsImpl; -import org.apache.kafka.streams.state.KeyValueStore; -import org.apache.kafka.streams.state.internals.ThreadCache; - -import java.io.IOException; -import java.util.ArrayList; -import java.util.Collection; -import java.util.Collections; -import java.util.HashMap; -import java.util.HashSet; -import java.util.LinkedList; -import java.util.List; -import java.util.Map; -import java.util.Queue; -import java.util.Set; -import java.util.concurrent.atomic.AtomicLong; - -/** - * This class makes it easier to write tests to verify the behavior of topologies created with a {@link Topology}. - * You can test simple topologies that have a single processor, or very complex topologies that have multiple sources, processors, - * and sinks. And because it starts with a {@link Topology}, you can create topologies specific to your tests or you - * can use and test code you already have that uses a builder to create topologies. Best of all, the class works without a real - * Kafka broker, so the tests execute very quickly with very little overhead. - * <p> - * Using the ProcessorTopologyTestDriver in tests is easy: simply instantiate the driver with a {@link StreamsConfig} and a - * TopologyBuilder, use the driver to supply an input message to the topology, and then use the driver to read and verify any - * messages output by the topology. - * <p> - * Although the driver doesn't use a real Kafka broker, it does simulate Kafka {@link org.apache.kafka.clients.consumer.Consumer}s - * and {@link org.apache.kafka.clients.producer.Producer}s that read and write raw {@code byte[]} messages. You can either deal - * with messages that have {@code byte[]} keys and values, or you can supply the {@link Serializer}s and {@link Deserializer}s - * that the driver can use to convert the keys and values into objects. - * - * <h2>Driver setup</h2> - * <p> - * In order to create a ProcessorTopologyTestDriver instance, you need a TopologyBuilder and a {@link StreamsConfig}. The - * configuration needs to be representative of what you'd supply to the real topology, so that means including several key - * properties. For example, the following code fragment creates a configuration that specifies a local Kafka broker list - * (which is needed but not used), a timestamp extractor, and default serializers and deserializers for string keys and values: - * - * <pre> - * StringSerializer strSerializer = new StringSerializer(); - * StringDeserializer

```
strDeserializer = new StringDeserializer(); - * Properties props = new Properties(); - *
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - *
props.setProperty(StreamsConfig.DEFAULT_TIMESTAMP_EXTRACTOR_CLASS_CONFIG,
CustomTimestampExtractor.class.getName()); - * props.setProperty(StreamsConfig.KEY_SERIALIZER_CLASS_CONFIG,
strSerializer.getClass().getName()); - * props.setProperty(StreamsConfig.KEY_DESERIALIZER_CLASS_CONFIG,
strDeserializer.getClass().getName()); - * props.setProperty(StreamsConfig.VALUE_SERIALIZER_CLASS_CONFIG,
strSerializer.getClass().getName()); - * props.setProperty(StreamsConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
strDeserializer.getClass().getName()); - * StreamsConfig config = new StreamsConfig(props); - * TopologyBuilder builder = ... - *
ProcessorTopologyTestDriver driver = new ProcessorTopologyTestDriver(config, builder); - * </pre> - * - * <h2>Processing
messages</h2> - * <p> - * Your test can supply new input records on any of the topics that the topology's sources consume. Here's an
example of an - * input message on the topic named {@code input-topic}: - * - * <pre> - * driver.process("input-topic", "key1", "value1",
strSerializer, strSerializer); - * </pre> - * - * Immediately, the driver will pass the input message through to the appropriate source that
consumes the named topic, - * and will invoke the processor(s) downstream of the source. If your topology's processors forward messages
to sinks, - * your test can then consume these output messages to verify they match the expected outcome. For example, if our topology - *
should have generated 2 messages on {@code output-topic-1} and 1 message on {@code output-topic-2}, then our test can - * obtain
these messages using the {@link #readOutput(String, Deserializer, Deserializer)} method: - * - * <pre> - * ProducerRecord<String,
String> record1 = driver.readOutput("output-topic-1", strDeserializer, strDeserializer); - * ProducerRecord<String, String> record2 =
driver.readOutput("output-topic-1", strDeserializer, strDeserializer); - * ProducerRecord<String, String> record3 =
driver.readOutput("output-topic-2", strDeserializer, strDeserializer); - * </pre> - * - * Again, our example topology generates messages
with string keys and values, so we supply our string deserializer instance - * for use on both the keys and values. Your test logic can then
verify whether these output records are correct. - * <p> - * Finally, when completed, make sure your tests {@link #close()} the driver to
release all resources and - * {@link org.apache.kafka.streams.processor.Processor}s. - * - * <h2>Processor state</h2> - * <p> - * Some
processors use Kafka {@link StateStore state storage}, so this driver class provides the {@link #getStateStore(String)} - * and {@link
#getKeyValueStore(String)} methods so that your tests can check the underlying state store(s) used by your - * topology's processors. In
our previous example, after we supplied a single input message and checked the three output messages, - * our test could also check the
key value store to verify the processor correctly added, removed, or updated internal state. - * Or, our test might have pre-populated some
state <em>before</em> submitting the input message, and verified afterward that the - * processor(s) correctly updated the state. - */ -
public class ProcessorTopologyTestDriver { - - private final static String APPLICATION_ID = "test-driver-application"; - private final
static int PARTITION_ID = 0; - private final static TaskId TASK_ID = new TaskId(0, PARTITION_ID); - - private final
ProcessorTopology topology; - private final MockProducer<byte[], byte[]> producer; - private final Map<String, TopicPartition>
partitionsByTopic = new HashMap<>(); - private final Map<TopicPartition, AtomicLong> offsetsByTopicPartition = new HashMap<>();
- private final Map<String, Queue<ProducerRecord<byte[], byte[]>>> outputRecordsByTopic = new HashMap<>(); - private final
Set<String> internalTopics = new HashSet<>(); - private final Map<String, TopicPartition> globalPartitionsByTopic = new HashMap<>
(); - private StreamTask task; - private GlobalStateUpdateTask globalStateTask; - - /** - * Create a new test driver instance. - * - *
@param config the stream configuration for the topology - * @param builder the topology builder that will be used to create the topology
instance - */ - public ProcessorTopologyTestDriver(final StreamsConfig config, - final InternalTopologyBuilder builder) { - topology =
builder.setApplicationId(APPLICATION_ID).build(null); - final ProcessorTopology globalTopology =
builder.buildGlobalStateTopology(); - - // Set up the consumer and producer ... - final Consumer<byte[], byte[]> consumer = new
MockConsumer<>(OffsetResetStrategy.EARLIEST); - final Serializer<byte[]> bytesSerializer = new ByteArraySerializer(); - producer =
new MockProducer<byte[], byte[]>(true, bytesSerializer, bytesSerializer) { - @Override - public List<PartitionInfo> partitionsFor(final
String topic) { - return Collections.singletonList(new PartitionInfo(topic, PARTITION_ID, null, null, null)); - } - }; - - // Identify internal
topics for forwarding in process ... - for (final InternalTopologyBuilder.TopicsInfo topicsInfo : builder.topicGroups().values()) { -
internalTopics.addAll(topicsInfo.repartitionSourceTopics.keySet()); - } - - // Set up all of the topic+partition information and subscribe the
consumer to each ... - for (final String topic : topology.sourceTopics()) { - final TopicPartition tp = new TopicPartition(topic,
PARTITION_ID); - partitionsByTopic.put(topic, tp); - offsetsByTopicPartition.put(tp, new AtomicLong()); - } - -
consumer.assign(offsetsByTopicPartition.keySet()); - - final StateDirectory stateDirectory = new StateDirectory(config, Time.SYSTEM); -
final StreamsMetricsImpl streamsMetrics = new MockStreamsMetrics(new Metrics()); - final ThreadCache cache = new
ThreadCache(new LogContext("mock "), 1024 * 1024, streamsMetrics); - - if (globalTopology != null) { - final MockConsumer<byte[],
byte[]> globalConsumer = createGlobalConsumer(); - final MockStateRestoreListener stateRestoreListener = new
MockStateRestoreListener(); - for (final String topicName : globalTopology.sourceTopics()) { - final List<PartitionInfo> partitionInfos =
new ArrayList<>(); - partitionInfos.add(new PartitionInfo(topicName, 1, null, null, null)); - globalConsumer.updatePartitions(topicName,
partitionInfos); - final TopicPartition partition = new TopicPartition(topicName, 1); -
globalConsumer.updateEndOffsets(Collections.singletonMap(partition, 0L)); - globalPartitionsByTopic.put(topicName, partition); -
offsetsByTopicPartition.put(partition, new AtomicLong()); - } - final GlobalStateManagerImpl stateManager = new
GlobalStateManagerImpl( - new LogContext("mock "), - globalTopology, - globalConsumer, - stateDirectory, - stateRestoreListener, -
config); - final GlobalProcessorContextImpl globalProcessorContext = new GlobalProcessorContextImpl(config, stateManager,
streamsMetrics, cache); - stateManager.setGlobalProcessorContext(globalProcessorContext); - globalStateTask = new
GlobalStateUpdateTask( - globalTopology, - globalProcessorContext, - stateManager, - new LogAndContinueExceptionHandler(), - new
LogContext()); - globalStateTask.initialize(); - } - - if (!partitionsByTopic.isEmpty()) { - task = new StreamTask( - TASK_ID, -
partitionsByTopic.values(), - topology, - consumer, - new StoreChangelogReader( -
createRestoreConsumer(topology.storeToChangelogTopic()), - new MockStateRestoreListener(), - new LogContext("topology-test-driver
") - ), - config, - streamsMetrics, - stateDirectory, - cache, - new MockTime(), - producer - ); - task.initializeStateStores(); -
task.initializeTopology(); - } - } - - /** - * Send an input message with the given key, value and timestamp on the specified topic to the
topology, and then commit the messages. - * - * @param topicName the name of the topic on which the message is to be sent - * @param
key the raw message key - * @param value the raw message value - * @param timestamp the raw message timestamp - */ - public void
process(final String topicName, - final byte[] key, - final byte[] value, - final long timestamp) { - - final TopicPartition tp =
partitionsByTopic.get(topicName); - if (tp != null) { - // Add the record ... - final long offset =
offsetsByTopicPartition.get(tp).incrementAndGet(); - task.addRecords(tp, records(new ConsumerRecord<>(tp.topic(), tp.partition(),
offset, timestamp, TimestampType.CREATE_TIME, 0L, 0, 0, key, value))); - producer.clear(); - - // Process the record ... - task.process(); -
((InternalProcessorContext) task.context()).setRecordContext(new ProcessorRecordContext(timestamp, offset, tp.partition(), topicName));
- task.commit(); - - // Capture all the records sent to the producer ... - for (final ProducerRecord<byte[], byte[]> record : producer.history())
{ - Queue<ProducerRecord<byte[], byte[]>> outputRecords = outputRecordsByTopic.get(record.topic()); - if (outputRecords == null) { -
outputRecords = new LinkedList<>(); - outputRecordsByTopic.put(record.topic(), outputRecords); - } - outputRecords.add(record); - - //
Forward back into the topology if the produced record is to an internal or a source topic ... - if (internalTopics.contains(record.topic()) ||
topology.sourceTopics().contains(record.topic())) { - process(record.topic(), record.key(), record.value(), record.timestamp()); - } - } - }
else { - final TopicPartition global = globalPartitionsByTopic.get(topicName); - if (global == null) { - throw new
```

IllegalArgumentException("Unexpected topic: " + topicName); - } - final long offset = offsetsByTopicPartition.get(global).incrementAndGet(); - globalStateTask.update(new ConsumerRecord<>(global.topic(), global.partition(), offset, timestamp, TimestampType.CREATE_TIME, 0L, 0, 0, key, value)); - globalStateTask.flushState(); - } - } - - /** - * Send an input message with the given key and value on the specified topic to the topology. - * - * @param topicName the name of the topic on which the message is to be sent - * @param key the raw message key - * @param value the raw message value - */ - public void process(final String topicName, - final byte[] key, - final byte[] value) { - process(topicName, key, value, 0L); - } - - /** - * Send an input message with the given key and value on the specified topic to the topology. - * - * @param topicName the name of the topic on which the message is to be sent - * @param key the raw message key - * @param value the raw message value - * @param keySerializer the serializer for the key - * @param valueSerializer the serializer for the value - */ - public <K, V> void process(final String topicName, - final K key, - final V value, - final Serializer<K> keySerializer, - final Serializer<V> valueSerializer) { - process(topicName, key, value, keySerializer, valueSerializer, 0L); - } - - /** - * Send an input message with the given key and value and timestamp on the specified topic to the topology. - * - * @param topicName the name of the topic on which the message is to be sent - * @param key the raw message key - * @param value the raw message value - * @param keySerializer the serializer for the key - * @param valueSerializer the serializer for the value - * @param timestamp the raw message timestamp - */ - public <K, V> void process(final String topicName, - final K key, - final V value, - final Serializer<K> keySerializer, - final Serializer<V> valueSerializer, - final long timestamp) { - process(topicName, keySerializer.serialize(topicName, key), valueSerializer.serialize(topicName, value), timestamp); - } - - /** - * Read the next record from the given topic. These records were output by the topology during the previous calls to - * {@link #process(String, byte[], byte[])}. - * - * @param topic the name of the topic - * @return the next record on that topic, or null if there is no record available - */ - public ProducerRecord<byte[], byte[]> readOutput(final String topic) { - final Queue<ProducerRecord<byte[], byte[]>> outputRecords = outputRecordsByTopic.get(topic); - if (outputRecords == null) { - return null; - } - return outputRecords.poll(); - } - - /** - * Read the next record from the given topic. These records were output by the topology during the previous calls to - * {@link #process(String, byte[], byte[])}. - * - * @param topic the name of the topic - * @param keyDeserializer the deserializer for the key type - * @param valueDeserializer the deserializer for the value type - * @return the next record on that topic, or null if there is no record available - */ - public <K, V> ProducerRecord<K, V> readOutput(final String topic, - final Deserializer<K> keyDeserializer, - final Deserializer<V> valueDeserializer) { - final ProducerRecord<byte[], byte[]> record = readOutput(topic); - if (record == null) { - return null; - } - final K key = keyDeserializer.deserialize(record.topic(), record.key()); - final V value = valueDeserializer.deserialize(record.topic(), record.value()); - return new ProducerRecord<>(record.topic(), record.partition(), record.timestamp(), key, value); - } - - private Iterable<ConsumerRecord<byte[], byte[]>> records(final ConsumerRecord<byte[], byte[]> record) { - return Collections.singleton(record); - } - - /** - * Get the {@link StateStore} with the given name. The name should have been supplied via - * {@link #ProcessorTopologyTestDriver(StreamsConfig, InternalTopologyBuilder) this object's constructor}, and is - * presumed to be used by a Processor within the topology. - * <p> - * This is often useful in test cases to pre-populate the store before the test case instructs the topology to - * {@link #process(String, byte[], byte[]) process an input message}, and/or to check the store afterward. - * - * @param name the name of the store - * @return the state store, or null if no store has been registered with the given name - * @see #getKeyValueStore(String) - */ - public StateStore getStateStore(final String name) { - return ((ProcessorContextImpl) task.context()).getStateMgr().getStore(name); - } - - /** - * Get the {@link KeyValueStore} with the given name. The name should have been supplied via - * {@link #ProcessorTopologyTestDriver(StreamsConfig, InternalTopologyBuilder) this object's constructor}, and is - * presumed to be used by a Processor within the topology. - * <p> - * This is often useful in test cases to pre-populate the store before the test case instructs the topology to - * {@link #process(String, byte[], byte[]) process an input message}, and/or to check the store afterward. - * <p> - * - * @param name the name of the store - * @return the key value store, or null if no {@link KeyValueStore} has been registered with the given name - * @see #getStateStore(String) - */ - @SuppressWarnings("unchecked") - public <K, V> KeyValueStore<K, V> getKeyValueStore(final String name) { - final StateStore store = getStateStore(name); - return store instanceof KeyValueStore ? (KeyValueStore<K, V>) getStateStore(name) : null; - } - - /** - * Close the driver, its topology, and all processors. - */ - public void close() { - if (task != null) { - task.close(true, false); - } - if (globalStateTask != null) { - try { - globalStateTask.close(); - } catch (final IOException e) { - // ignore - } - } - } - - /** - * Utility method that creates the {@link MockConsumer} used for restoring state, which should not be done by this - * driver object unless this method is overwritten with a functional consumer. - * - * @param storeToChangelogTopic the map of the names of the stores to the changelog topics - * @return the mock consumer; never null - */ - private MockConsumer<byte[], byte[]> createRestoreConsumer(final Map<String, String> storeToChangelogTopic) { - final MockConsumer<byte[], byte[]> consumer = new MockConsumer<byte[], byte[]>(OffsetResetStrategy.LATEST) { - @Override - public synchronized void seekToEnd(final Collection<TopicPartition> partitions) {} - - @Override - public synchronized void seekToBeginning(final Collection<TopicPartition> partitions) {} - - @Override - public synchronized long position(final TopicPartition partition) { - return 0L; - } - }; - // For each store ... - for (final Map.Entry<String, String> storeAndTopic : storeToChangelogTopic.entrySet()) { - final String topicName = storeAndTopic.getValue(); - // Set up the restore-state topic ... - // consumer.subscribe(new TopicPartition(topicName, 1)); - // Set up the partition that matches the ID (which is what ProcessorStateManager expects) ... - final List<PartitionInfo> partitionInfos = new ArrayList<>(); - partitionInfos.add(new PartitionInfo(topicName, PARTITION_ID, null, null, null)); - consumer.updatePartitions(topicName, partitionInfos); - consumer.updateEndOffsets(Collections.singletonMap(new TopicPartition(topicName, PARTITION_ID), 0L)); - } - return consumer; - } - - private MockConsumer<byte[], byte[]> createGlobalConsumer() { - return new MockConsumer<byte[], byte[]>(OffsetResetStrategy.LATEST) { - @Override - public synchronized void seekToEnd(final Collection<TopicPartition> partitions) {} - - @Override - public synchronized void seekToBeginning(final Collection<TopicPartition> partitions) {} - - @Override - public synchronized long position(final TopicPartition partition) { - return 0L; - } - }; - } - -} diff --git a/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java b/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java index bde70b4c6d4..7343d59e013 100644 --- a/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java +++ b/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java @@ -216,10 +216,36 @@ public TopologyTestDriver(final Topology topology, public TopologyTestDriver(final Topology topology, final Properties config, final long initialWallClockTimeMs) { + + this(topology.internalTopologyBuilder, config, initialWallClockTimeMs); + } + + /** + * Create a new test diver instance. + * + * @param builder builder for the topology to be tested + * @param config the configuration for the topology + */ + protected TopologyTestDriver(final InternalTopologyBuilder builder, + final Properties config) { + this(builder, config, System.currentTimeMillis()); + + } + + /** + * Create a new test diver instance. + * + * @param builder builder for the topology to be tested + * @param config the configuration for the topology + * @param initialWallClockTimeMs the initial value of internally mocked wall-clock time + */ + private TopologyTestDriver(final InternalTopologyBuilder builder, + final Properties config, + final long initialWallClockTimeMs) { final StreamsConfig streamsConfig = new StreamsConfig(config); mockTime = new MockTime(initialWallClockTimeMs); - internalTopologyBuilder = topology.internalTopologyBuilder; + internalTopologyBuilder = builder; internalTopologyBuilder.setApplicationId(streamsConfig.getString(StreamsConfig.APPLICATION_ID_CONFIG)); processorTopology = internalTopologyBuilder.build(null); -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

22. h314to opened a new pull request #4939: KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [cleanup] URL: https://github.com/apache/kafka/pull/4939 This implements the suggestions made after the previous [PR] (https://github.com/apache/kafka/pull/4832) for KAFKA-6474 was merged. The majority of changes deals with using try-with-resources and a new method in `StreamsTestUtils` to set the test properties and instantiate the `TopologyTestDriver`, thus allowing the removal of the cumbersome `@Before` and `@After` methods. I also replaced `stringSerde` and `intSerde` variables with (almost equally succinct) inline calls to `Serdes.String()` and `Serdes.Integer()`. * Add method to create test properties to StreamsTestUtils * Make TopologyTestDriver protected constructor package-private * Add comment suggesting the use of TopologyTestDriver to KStreamTestDriver * Cleanup: - GlobalKTableJoinsTest - KGroupedStreamImplTest - KGroupedTableImplTest - KStreamBranchTest - KStreamFilterTest - KStreamFlatMapTest - KStreamFlatMapValuesTest - KStreamForeachTest - KStreamGlobalKTableJoinTest - KStreamGlobalKTableLeftJoinTest - KStreamImplTest - KStreamKStreamJoinTest - KStreamKStreamLeftJoinTest - KStreamGlobalKTableLeftJoinTest - KStreamKTableJoinTest - KStreamKTableLeftJoinTest - KStreamMapTest - KStreamMapValuesTest - KStreamPeekTest - StreamsBuilderTest - KStreamSelectKeyTest - KStreamTransformTest - KStreamTransformValuesTest - KStreamWindowAggregateTest - KTableForeachTest -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
23. guozhangwang closed pull request #4939: KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [cleanup] URL: https://github.com/apache/kafka/pull/4939 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/streams/src/test/java/org/apache/kafka/streams/StreamsBuilderTest.java b/streams/src/test/java/org/apache/kafka/streams/StreamsBuilderTest.java index 15e55d873a4..7c2bfa6b16a 100644 --- a/streams/src/test/java/org/apache/kafka/streams/StreamsBuilderTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/StreamsBuilderTest.java @@ -35,9 +35,7 @@ import org.apache.kafka.test.MockPredicate; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; -import org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import java.util.Arrays; @@ -58,26 +56,7 @@ public class StreamsBuilderTest { private final StreamsBuilder builder = new StreamsBuilder(); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - - @Before - public void setup() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "streams-builder-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After - public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); @Test(expected = TopologyException.class) public void testFrom() { @@ -183,10 +162,10 @@ public void shouldProcessingFromSinkTopic() { source.process(processorSupplier); - driver = new TopologyTestDriver(builder.build(), props); - - final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); - driver.pipeInput(recordFactory.create("topic-source", "A", "aa")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + driver.pipeInput(recordFactory.create("topic-source", "A", "aa")); + } // no exception was thrown assertEquals(Utils.mkList("A:aa"), processorSupplier.theCapturedProcessor().processed); @@ -203,10 +182,10 @@ public void shouldProcessViaThroughTopic() { source.process(sourceProcessorSupplier); through.process(throughProcessorSupplier); - driver = new TopologyTestDriver(builder.build(), props); - - final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); - driver.pipeInput(recordFactory.create("topic-source", "A", "aa")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + driver.pipeInput(recordFactory.create("topic-source", "A", "aa")); + } assertEquals(Utils.mkList("A:aa"), sourceProcessorSupplier.theCapturedProcessor().processed); assertEquals(Utils.mkList("A:aa"), throughProcessorSupplier.theCapturedProcessor().processed); @@ -224,13 +203,13 @@ public void testMerge() { final MockProcessorSupplier<String, String> processorSupplier = new MockProcessorSupplier<>(); merged.process(processorSupplier); - driver = new TopologyTestDriver(builder.build(), props); - final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); - driver.pipeInput(recordFactory.create(topic1, "A", "aa")); - driver.pipeInput(recordFactory.create(topic2, "B", "bb")); - driver.pipeInput(recordFactory.create(topic2, "C", "cc")); - driver.pipeInput(recordFactory.create(topic1, "D", "dd")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(topic1, "A", "aa")); + driver.pipeInput(recordFactory.create(topic2, "B", "bb")); + driver.pipeInput(recordFactory.create(topic2, "C", "cc")); + driver.pipeInput(recordFactory.create(topic1, "D", "dd")); + } assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd"), processorSupplier.theCapturedProcessor().processed); } @@ -250,17 +229,17 @@ public void apply(final Long key, final String value) { .withValueSerde(Serdes.String())) .toStream().foreach(action); - driver = new TopologyTestDriver(builder.build(), props); - final ConsumerRecordFactory<Long, String> recordFactory = new ConsumerRecordFactory<>(new LongSerializer(), new StringSerializer()); - driver.pipeInput(recordFactory.create(topic, 1L, "value1")); - driver.pipeInput(recordFactory.create(topic, 2L, "value2")); - - final KeyValueStore<Long, String> store = driver.getKeyValueStore("store"); - assertThat(store.get(1L), equalTo("value1")); - assertThat(store.get(2L), equalTo("value2")); - assertThat(results.get(1L), equalTo("value1")); - assertThat(results.get(2L), equalTo("value2")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(topic, 1L, "value1")); + driver.pipeInput(recordFactory.create(topic, 2L, "value2")); + + final KeyValueStore<Long, String> store = driver.getKeyValueStore("store"); + assertThat(store.get(1L), equalTo("value1")); + assertThat(store.get(2L), equalTo("value2")); + assertThat(results.get(1L), equalTo("value1")); + assertThat(results.get(2L), equalTo("value2")); + } } @Test @@ -270,15 +249,15 @@ public void shouldUseSerdesDefinedInMaterializedToConsumeGlobalTable() { .withKeySerde(Serdes.Long()) .withValueSerde(Serdes.String())); - driver = new TopologyTestDriver(builder.build(), props); - final ConsumerRecordFactory<Long, String> recordFactory = new ConsumerRecordFactory<>(new LongSerializer(), new StringSerializer()); - driver.pipeInput(recordFactory.create(topic, 1L, "value1")); - driver.pipeInput(recordFactory.create(topic, 2L, "value2")); - final KeyValueStore<Long, String> store = driver.getKeyValueStore("store"); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(topic, 1L, "value1")); + driver.pipeInput(recordFactory.create(topic, 2L, "value2")); + final KeyValueStore<Long, String> store = driver.getKeyValueStore("store"); - assertThat(store.get(1L), equalTo("value1")); - assertThat(store.get(2L), equalTo("value2")); +

```
assertThat(store.get(1L), equalTo("value1")); + assertThat(store.get(2L), equalTo("value2")); + } } @Test diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/GlobalKTableJoinsTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/GlobalKTableJoinsTest.java index 8c50afeda6f..da8b102fde6 100644 -
-- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/GlobalKTableJoinsTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/GlobalKTableJoinsTest.java @@ -20,7 +20,6 @@ import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig; import
org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.ForeachAction; import
org.apache.kafka.streams.kstream.GlobalKTable; @@ -28,8 +27,7 @@ import org.apache.kafka.streams.kstream.KeyValueMapper;
import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockValueJoiner; -import
org.apache.kafka.test.TestUtils; -import org.junit.After; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Before; import
org.junit.Test; @@ -50,7 +48,6 @@ private KStream<String, String> stream; private KeyValueMapper<String, String, String>
keyValueMapper; private ForeachAction<String, String> action; - private TopologyTestDriver driver; @Before @@ -72,14 +69,6 @@
public void apply(final String key, final String value) { }; } - @After - public void cleanup() { - if (driver != null) { - driver.close(); - } -
driver = null; - } - @Test public void shouldLeftJoinWithStream() { stream @@ -110,21 +99,17 @@ public void
shouldInnerJoinWithStream() { private void verifyJoin(final Map<String, String> expected) { final ConsumerRecordFactory<String,
String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); - - final Properties props = new
Properties(); - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "global-ktable-joins-test"); -
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - - driver = new
TopologyTestDriver(builder.build(), props); - - // write some data to the global table - driver.pipeInput(recordFactory.create(globalTopic,
"a", "A")); - driver.pipeInput(recordFactory.create(globalTopic, "b", "B")); - //write some data to the stream -
driver.pipeInput(recordFactory.create(streamTopic, "1", "a")); - driver.pipeInput(recordFactory.create(streamTopic, "2", "b")); -
driver.pipeInput(recordFactory.create(streamTopic, "3", "c")); + final Properties props =
StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); + + try (final TopologyTestDriver driver = new
TopologyTestDriver(builder.build(), props)) { + // write some data to the global table + driver.pipeInput(recordFactory.create(globalTopic,
"a", "A")); + driver.pipeInput(recordFactory.create(globalTopic, "b", "B")); + //write some data to the stream +
driver.pipeInput(recordFactory.create(streamTopic, "1", "a")); + driver.pipeInput(recordFactory.create(streamTopic, "2", "b")); +
driver.pipeInput(recordFactory.create(streamTopic, "3", "c")); + } assertEquals(expected, results); } diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedStreamImplTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedStreamImplTest.java index b9ca30f48cc..e7a9226d89b
100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedStreamImplTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedStreamImplTest.java @@ -25,7 +25,6 @@ import
org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; -
import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.Aggregator; import org.apache.kafka.streams.kstream.ForeachAction; @@ -50,8 +49,7 @@ import
org.apache.kafka.test.MockAggregator; import org.apache.kafka.test.MockInitializer; import org.apache.kafka.test.MockReducer; -import
org.apache.kafka.test.TestUtils; -import org.junit.After; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Before; import
org.junit.Test; @@ -78,28 +76,13 @@ private KGroupedStream<String, String> groupedStream; private final
ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new
StringSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); + private final Properties props
= StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); @Before public void before() { final KStream<String, String>
stream = builder.stream(TOPIC, Consumed.with(Serdes.String(), Serdes.String())); groupedStream =
stream.groupByKey(Serialized.with(Serdes.String(), Serdes.String())); - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG,
"kgrouped-stream-impl-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); -
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After -
public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; } @SuppressWarnings("deprecation") @@
-224,13 +207,14 @@ public void shouldNotHaveNullStoreSupplierOnWindowedAggregate() { } private void
doAggregateSessionWindows(final Map<Windowed<String>, Integer> results) { - driver = new TopologyTestDriver(builder.build(),
props); - driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 10)); - driver.pipeInput(recordFactory.create(TOPIC, "2", "2", 15)); -
driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 30)); - driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 70)); -
driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 90)); - driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 100)); + try (final
TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(TOPIC, "1", "1",
10)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "2", 15)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 30)); +
driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 70)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 90)); +
driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 100)); + } assertEquals(Integer.valueOf(2), results.get(new Windowed<>("1", new
SessionWindow(10, 30)))); assertEquals(Integer.valueOf(1), results.get(new Windowed<>("2", new SessionWindow(15, 15))));
assertEquals(Integer.valueOf(3), results.get(new Windowed<>("1", new SessionWindow(70, 100)))); @@ -298,13 +282,14 @@ public
void apply(final Windowed<String> key, final Integer value) { } private void doCountSessionWindows(final Map<Windowed<String>,
Long> results) { - driver = new TopologyTestDriver(builder.build(), props); - driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 10));
- driver.pipeInput(recordFactory.create(TOPIC, "2", "2", 15)); - driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 30)); -
driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 70)); - driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 90)); -
driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 100)); + try (final TopologyTestDriver driver = new
TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 10)); +
driver.pipeInput(recordFactory.create(TOPIC, "2", "2", 15)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 30)); +
driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 70)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 90)); +
driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 100)); + } assertEquals(Long.valueOf(2), results.get(new Windowed<>("1", new
SessionWindow(10, 30)))); assertEquals(Long.valueOf(1), results.get(new Windowed<>("2", new SessionWindow(15, 15))));
assertEquals(Long.valueOf(3), results.get(new Windowed<>("1", new SessionWindow(70, 100)))); @@ -341,13 +326,14 @@ public
void apply(final Windowed<String> key, final Long value) { } private void doReduceSessionWindows(final Map<Windowed<String>,
String> results) { - driver = new TopologyTestDriver(builder.build(), props); - driver.pipeInput(recordFactory.create(TOPIC, "1", "A",
10)); - driver.pipeInput(recordFactory.create(TOPIC, "2", "Z", 15)); - driver.pipeInput(recordFactory.create(TOPIC, "1", "B", 30)); -
driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 70)); - driver.pipeInput(recordFactory.create(TOPIC, "1", "B", 90)); -
driver.pipeInput(recordFactory.create(TOPIC, "1", "C", 100)); + try (final TopologyTestDriver driver = new
```

TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 10)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "Z", 15)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "B", 30)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 70)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "B", 90)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "C", 100)); + } assertEquals("A:B", results.get(new Windowed<>("1", new SessionWindow(10, 30)))); assertEquals("Z", results.get(new Windowed<>("2", new SessionWindow(15, 15)))); assertEquals("A:B:C", results.get(new Windowed<>("1", new SessionWindow(70, 100)))); @@ -554,26 +540,30 @@ public void shouldThrowNullPointerOnCountWhenMaterializedIsNull() { public void shouldCountAndMaterializeResults() { groupedStream.count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("count").withKeySerde(Serdes.String())); - processData(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + processData(driver); - final KeyValueStore<String, Long> count = driver.getKeyValueStore("count"); + final KeyValueStore<String, Long> count = driver.getKeyValueStore("count"); - assertThat(count.get("1"), equalTo(3L)); - assertThat(count.get("2"), equalTo(1L)); - assertThat(count.get("3"), equalTo(2L)); + assertThat(count.get("1"), equalTo(3L)); + assertThat(count.get("2"), equalTo(1L)); + assertThat(count.get("3"), equalTo(2L)); + } } @Test public void shouldLogAndMeasureSkipsInAggregate() { groupedStream.count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("count").withKeySerde(Serdes.String())); final LogCaptureAppender appender = LogCaptureAppender.createAndRegister(); - processData(); - LogCaptureAppender.unregister(appender); - - final Map<MetricName, ? extends Metric> metrics = driver.metrics(); - assertEquals(1.0, getMetricByName(metrics, "skipped-records-total", "stream-metrics").metricValue()); - assertNotEquals(0.0, getMetricByName(metrics, "skipped-records-rate", "stream-metrics").metricValue()); - assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[3] value=[null] topic=[topic] partition=[0] offset=[6]")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + processData(driver); + LogCaptureAppender.unregister(appender); + + final Map<MetricName, ? extends Metric> metrics = driver.metrics(); + assertEquals(1.0, getMetricByName(metrics, "skipped-records-total", "stream-metrics").metricValue()); + assertNotEquals(0.0, getMetricByName(metrics, "skipped-records-rate", "stream-metrics").metricValue()); + assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[3] value=[null] topic=[topic] partition=[0] offset=[6]")); + } } @@ -586,13 +576,15 @@ public void shouldReduceAndMaterializeResults() { .withKeySerde(Serdes.String()) .withValueSerde(Serdes.String())); - processData(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + processData(driver); - final KeyValueStore<String, String> reduced = driver.getKeyValueStore("reduce"); + final KeyValueStore<String, String> reduced = driver.getKeyValueStore("reduce"); - assertThat(reduced.get("1"), equalTo("A+C+D")); - assertThat(reduced.get("2"), equalTo("B")); - assertThat(reduced.get("3"), equalTo("E+F")); + assertThat(reduced.get("1"), equalTo("A+C+D")); + assertThat(reduced.get("2"), equalTo("B")); + assertThat(reduced.get("3"), equalTo("E+F")); + } } @Test @@ -605,13 +597,15 @@ public void shouldLogAndMeasureSkipsInReduce() {); final LogCaptureAppender appender = LogCaptureAppender.createAndRegister(); - processData(); - LogCaptureAppender.unregister(appender); - - final Map<MetricName, ? extends Metric> metrics = driver.metrics(); - assertEquals(1.0, getMetricByName(metrics, "skipped-records-total", "stream-metrics").metricValue()); - assertNotEquals(0.0, getMetricByName(metrics, "skipped-records-rate", "stream-metrics").metricValue()); - assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[3] value=[null] topic=[topic] partition=[0] offset=[6]")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + processData(driver); + LogCaptureAppender.unregister(appender); + + final Map<MetricName, ? extends Metric> metrics = driver.metrics(); + assertEquals(1.0, getMetricByName(metrics, "skipped-records-total", "stream-metrics").metricValue()); + assertNotEquals(0.0, getMetricByName(metrics, "skipped-records-rate", "stream-metrics").metricValue()); + assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=[3] value=[null] topic=[topic] partition=[0] offset=[6]")); + } } @@ -625,13 +619,15 @@ public void shouldAggregateAndMaterializeResults() { .withKeySerde(Serdes.String()) .withValueSerde(Serdes.String())); - processData(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + processData(driver); - final KeyValueStore<String, String> aggregate = driver.getKeyValueStore("aggregate"); + final KeyValueStore<String, String> aggregate = driver.getKeyValueStore("aggregate"); - assertThat(aggregate.get("1"), equalTo("0+A+C+D")); - assertThat(aggregate.get("2"), equalTo("0+B")); - assertThat(aggregate.get("3"), equalTo("0+E+F")); + assertThat(aggregate.get("1"), equalTo("0+A+C+D")); + assertThat(aggregate.get("2"), equalTo("0+B")); + assertThat(aggregate.get("3"), equalTo("0+E+F")); + } } @SuppressWarnings("unchecked") @@ -649,15 +645,16 @@ public void apply(final String key, final String value) { } }); - processData(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + processData(driver); - assertThat(results.get("1"), equalTo("0+A+C+D")); - assertThat(results.get("2"), equalTo("0+B")); - assertThat(results.get("3"), equalTo("0+E+F")); + assertThat(results.get("1"), equalTo("0+A+C+D")); + assertThat(results.get("2"), equalTo("0+B")); + assertThat(results.get("3"), equalTo("0+E+F")); + } } - private void processData() { - driver = new TopologyTestDriver(builder.build(), props); + private void processData(final TopologyTestDriver driver) { driver.pipeInput(recordFactory.create(TOPIC, "1", "A")); driver.pipeInput(recordFactory.create(TOPIC, "2", "B")); driver.pipeInput(recordFactory.create(TOPIC, "1", "C")); @@ -668,22 +665,23 @@ private void processData() { } private void doCountWindowed(final List<KeyValue<Windowed<String>, Long>> results) { - driver = new TopologyTestDriver(builder.build(), props); - driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 0)); - driver.pipeInput(recordFactory.create(TOPIC, "2", "B", 0)); - driver.pipeInput(recordFactory.create(TOPIC, "3", "C", 0)); - driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 500)); - driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 500)); - driver.pipeInput(recordFactory.create(TOPIC, "2", "B", 500)); - driver.pipeInput(recordFactory.create(TOPIC, "2", "B", 500)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 0)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "B", 0)); + driver.pipeInput(recordFactory.create(TOPIC, "3", "C", 0)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 500)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "A", 500)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "B", 500)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "B", 500)); + } assertThat(results, equalTo(Arrays.asList(- KeyValue.pair(new Windowed<>("1", new TimeWindow(0, 500)), 1L), - KeyValue.pair(new Windowed<>("2", new TimeWindow(0, 500)), 1L), - KeyValue.pair(new Windowed<>("3", new TimeWindow(0, 500)), 1L), - KeyValue.pair(new Windowed<>("1", new TimeWindow(500, 1000)), 1L), - KeyValue.pair(new Windowed<>("1", new TimeWindow(500, 1000)), 2L), - KeyValue.pair(new Windowed<>("2", new TimeWindow(500, 1000)), 1L), - KeyValue.pair(new Windowed<>("2", new TimeWindow(500, 1000)), 2L) + KeyValue.pair(new Windowed<>("1", new TimeWindow(0, 500)), 1L), + KeyValue.pair(new Windowed<>("2", new TimeWindow(0, 500)), 1L), + KeyValue.pair(new Windowed<>("3", new TimeWindow(0, 500)), 1L), + KeyValue.pair(new Windowed<>("1", new TimeWindow(500, 1000)), 1L), + KeyValue.pair(new Windowed<>("1", new TimeWindow(500, 1000)), 2L), + KeyValue.pair(new Windowed<>("2", new TimeWindow(500, 1000)), 1L), + KeyValue.pair(new Windowed<>("2", new TimeWindow(500, 1000)), 2L)))); } diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedTableImplTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedTableImplTest.java index 742f3496579..05d339f9ad9 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedTableImplTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KGroupedTableImplTest.java @@ -24,7 +24,6 @@ import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; -

import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.ForeachAction; import org.apache.kafka.streams.kstream.KGroupedTable; @@ -39,8 +38,7 @@ import org.apache.kafka.test.MockInitializer; import org.apache.kafka.test.MockMapper; import org.apache.kafka.test.MockReducer; -import org.apache.kafka.test.TestUtils; -import org.junit.After; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Before; import org.junit.Test; @@ -60,29 +58,13 @@ private final StreamsBuilder builder = new StreamsBuilder(); private static final String INVALID_STORE_NAME = "~foo bar~"; private KGroupedTable<String, String> groupedTable; - private TopologyTestDriver driver; - private final Properties props = new Properties(); + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.Integer()); private final String topic = "input"; @Before public void before() { groupedTable = builder.table("blah", Consumed.with(Serdes.String(), Serdes.String())) .groupBy(MockMapper.<String, String>selectValueKeyValueMapper()); - - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kgrouped-table-impl-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - } - - @After - public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; } @Test @@ -140,30 +122,31 @@ public void shouldNotAllowNullStoreSupplierOnReduce() { groupedTable.reduce(MockReducer.STRING_ADDER, MockReducer.STRING_REMOVER, (StateStoreSupplier<KeyValueStore>) null); } - private void doShouldReduce(final KTable<String, Integer> reduced, final String topic) { - final Map<String, Integer> results = new HashMap<>(); - final ConsumerRecordFactory<String, Double> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new DoubleSerializer()); - reduced.foreach(new ForeachAction<String, Integer>() { + private Map<String, Integer> getReducedResults(final KTable<String, Integer> inputKTable) { + final Map<String, Integer> reducedResults = new HashMap<>(); + inputKTable.foreach(new ForeachAction<String, Integer>() { @Override public void apply(final String key, final Integer value) { - results.put(key, value); + reducedResults.put(key, value); } }); - - driver = new TopologyTestDriver(builder.build(), props); + return reducedResults; + } + private void assertReduced(final Map<String, Integer> reducedResults, final String topic, final TopologyTestDriver driver) { + final ConsumerRecordFactory<String, Double> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new DoubleSerializer()); driver.pipeInput(recordFactory.create(topic, "A", 1.1, 10)); driver.pipeInput(recordFactory.create(topic, "B", 2.2, 10)); - assertEquals(Integer.valueOf(1), results.get("A")); - assertEquals(Integer.valueOf(2), results.get("B")); + assertEquals(Integer.valueOf(1), reducedResults.get("A")); + assertEquals(Integer.valueOf(2), reducedResults.get("B")); driver.pipeInput(recordFactory.create(topic, "A", 2.6, 10)); driver.pipeInput(recordFactory.create(topic, "B", 1.3, 10)); driver.pipeInput(recordFactory.create(topic, "A", 5.7, 10)); driver.pipeInput(recordFactory.create(topic, "B", 6.2, 10)); - assertEquals(Integer.valueOf(5), results.get("A")); - assertEquals(Integer.valueOf(6), results.get("B")); + assertEquals(Integer.valueOf(5), reducedResults.get("A")); + assertEquals(Integer.valueOf(6), reducedResults.get("B")); } @Test @@ -184,8 +167,11 @@ public void shouldReduce() { .groupBy(intProjection) .reduce(MockReducer.INTEGER_ADDER, MockReducer.INTEGER_SUBTRACTOR, "reduced"); - doShouldReduce(reduced, topic); - assertEquals(reduced.queryableStoreName(), "reduced"); + final Map<String, Integer> results = getReducedResults(reduced); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + assertReduced(results, topic, driver); + assertEquals(reduced.queryableStoreName(), "reduced"); + } } @Test @@ -206,8 +192,11 @@ public void shouldReduceWithInternalStoreName() { .groupBy(intProjection) .reduce(MockReducer.INTEGER_ADDER, MockReducer.INTEGER_SUBTRACTOR); - doShouldReduce(reduced, topic); - assertNull(reduced.queryableStoreName()); + final Map<String, Integer> results = getReducedResults(reduced); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + assertReduced(results, topic, driver); + assertNull(reduced.queryableStoreName()); + } } @SuppressWarnings("unchecked") @@ -229,10 +218,13 @@ public void shouldReduceAndMaterializeResults() { .withKeySerde(Serdes.String()) .withValueSerde(Serdes.Integer())); - doShouldReduce(reduced, topic); - final KeyValueStore<String, Integer> reduce = (KeyValueStore<String, Integer>) driver.getStateStore("reduce"); - assertThat(reduce.get("A"), equalTo(5)); - assertThat(reduce.get("B"), equalTo(6)); + final Map<String, Integer> results = getReducedResults(reduced); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + assertReduced(results, topic, driver); + final KeyValueStore<String, Integer> reduce = (KeyValueStore<String, Integer>) driver.getStateStore("reduce"); + assertThat(reduce.get("A"), equalTo(5)); + assertThat(reduce.get("B"), equalTo(6)); + } } @SuppressWarnings("unchecked") @@ -246,10 +238,12 @@ public void shouldCountAndMaterializeResults() { .withKeySerde(Serdes.String()) .withValueSerde(Serdes.Long())); - processData(topic); - final KeyValueStore<String, Long> counts = driver.getKeyValueStore("count"); - assertThat(counts.get("1"), equalTo(3L)); - assertThat(counts.get("2"), equalTo(2L)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + processData(topic, driver); + final KeyValueStore<String, Long> counts = driver.getKeyValueStore("count"); + assertThat(counts.get("1"), equalTo(3L)); + assertThat(counts.get("2"), equalTo(2L)); + } } @SuppressWarnings("unchecked") @@ -266,10 +260,12 @@ public void shouldAggregateAndMaterializeResults() { .withValueSerde(Serdes.String()) .withKeySerde(Serdes.String())); - processData(topic); - final KeyValueStore<String, String> aggregate = (KeyValueStore<String, String>) driver.getStateStore("aggregate"); - assertThat(aggregate.get("1"), equalTo("0+1+1+1")); - assertThat(aggregate.get("2"), equalTo("0+2+2")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + processData(topic, driver); + final KeyValueStore<String, String> aggregate = (KeyValueStore<String, String>) driver.getStateStore("aggregate"); + assertThat(aggregate.get("1"), equalTo("0+1+1+1")); + assertThat(aggregate.get("2"), equalTo("0+2+2")); + } } @SuppressWarnings("unchecked") @@ -327,8 +323,7 @@ public void shouldThrowNullPointerOnAggregateWhenMaterializedIsNull() { (Materialized) null); } - private void processData(final String topic) { - driver = new TopologyTestDriver(builder.build(), props); + private void processData(final String topic, final TopologyTestDriver driver) { final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); driver.pipeInput(recordFactory.create(topic, "A", "1")); driver.pipeInput(recordFactory.create(topic, "B", "1")); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamBranchTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamBranchTest.java index bd3d60be8b5..2aa8aaceae0 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamBranchTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamBranchTest.java @@ -21,16 +21,13 @@ import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.Predicate; import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessor; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import java.util.List; @@ -42,26 +39,7 @@ private final String topicName = "topic"; private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - - @Before - public void before() { -

props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-branch-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After - public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); @SuppressWarnings("unchecked") @Test @@ -102,9 +80,10 @@ public boolean test(Integer key, String value) { branches[i].process(supplier); } - driver = new TopologyTestDriver(builder.build(), props); - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); + } } final List<MockProcessor<Integer, String>> processors = supplier.capturedProcessors(3); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFilterTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFilterTest.java index d338fe3f99b..51a994b6add 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFilterTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFilterTest.java @@ -21,15 +21,12 @@ import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.Predicate; import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import java.util.Properties; @@ -40,28 +37,9 @@ private final String topicName = "topic"; private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - - @Before - public void before() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-filter-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); - @After - public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } - - private Predicate<Integer, String> isMultipleOfThree = new Predicate<Integer, String>() { + private final Predicate<Integer, String> isMultipleOfThree = new Predicate<Integer, String>() { @Override public boolean test(Integer key, String value) { return (key % 3) == 0; @@ -80,9 +58,10 @@ public void testFilter() { stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.String())); stream.filter(isMultipleOfThree).process(supplier); - driver = new TopologyTestDriver(builder.build(), props); - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); + } } assertEquals(2, supplier.theCapturedProcessor().processed.size()); @@ -100,9 +79,10 @@ public void testFilterNot() { stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.String())); stream.filterNot(isMultipleOfThree).process(supplier); - driver = new TopologyTestDriver(builder.build(), props); - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); + } } assertEquals(5, supplier.theCapturedProcessor().processed.size()); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapTest.java index 9ce24b556af..3173dcfca5f 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapTest.java @@ -22,15 +22,12 @@ import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KeyValueMapper; import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import java.util.ArrayList; @@ -42,26 +39,7 @@ private String topicName = "topic"; private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - - @Before - public void before() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-flat-map-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After - public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); @Test public void testFlatMap() { @@ -88,9 +66,10 @@ public void testFlatMap() { stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.String())); stream.flatMap(mapper).process(supplier); - driver = new TopologyTestDriver(builder.build(), props); - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); + } } assertEquals(6, supplier.theCapturedProcessor().processed.size()); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapValuesTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapValuesTest.java index 221b02b22ef..471b1279b87 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapValuesTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamFlatMapValuesTest.java @@ -20,16 +20,13 @@ import org.apache.kafka.common.serialization.Serdes; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig; import

```
org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import
org.apache.kafka.streams.kstream.ValueMapper; import org.apache.kafka.streams.kstream.ValueMapperWithKey; import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import
org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import
org.junit.Test; import java.util.ArrayList; @@ -41,26 +38,7 @@ private String topicName = "topic"; private final
ConsumerRecordFactory<Integer, Integer> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new
IntegerSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - - @Before - public void
before() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-flat-map-values-test"); -
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); -
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After -
public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props =
StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); @Test public void testFlatMapValues() { @@ -83,10 +61,11
@@ public void testFlatMapValues() { final MockProcessorSupplier<Integer, String> supplier = new MockProcessorSupplier<>();
stream.flatMapValues(mapper).process(supplier); - driver = new TopologyTestDriver(builder.build(), props); - for (final int expectedKey :
expectedKeys) { - // passing the timestamp to recordFactory.create to disambiguate the call -
driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey, 0L)); + try (final TopologyTestDriver driver = new
TopologyTestDriver(builder.build(), props)) { + for (final int expectedKey : expectedKeys) { + // passing the timestamp to
recordFactory.create to disambiguate the call + driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey, 0L)); + } }
String[] expected = {"0:v0", "0:V0", "1:v1", "1:V1", "2:v2", "2:V2", "3:v3", "3:V3"}; @@ -117,10 +96,11 @@ public void
testFlatMapValuesWithKeys() { stream.flatMapValues(mapper).process(supplier); - driver = new TopologyTestDriver(builder.build(),
props); - for (final int expectedKey : expectedKeys) { - // passing the timestamp to recordFactory.create to disambiguate the call -
driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey, 0L)); + try (final TopologyTestDriver driver = new
TopologyTestDriver(builder.build(), props)) { + for (final int expectedKey : expectedKeys) { + // passing the timestamp to
recordFactory.create to disambiguate the call + driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey, 0L)); + } }
String[] expected = {"0:v0", "0:k0", "1:v1", "1:k1", "2:v2", "2:k2", "3:v3", "3:k3"}; diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamForeachTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamForeachTest.java index b975c96e7d1..83a20a60346 100644 --
- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamForeachTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamForeachTest.java @@ -17,20 +17,16 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig;
import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.ForeachAction; import
org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.test.ConsumerRecordFactory; -import
org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import
org.junit.Test; import java.util.ArrayList; @@ -45,29 +41,7 @@ private final String topicName = "topic"; private final
ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new
StringSerializer()); - private TopologyTestDriver driver; - private Properties props = new Properties(); - - @Before - public void before() {
- props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-foreach-test"); -
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); -
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After -
public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } - - private final Serde<Integer> intSerde
= Serdes.Integer(); - private final Serde<String> stringSerde = Serdes.String(); + private final Properties props =
StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); @Test public void testForeach() { @@ -97,13 +71,14 @@ public
void apply(Integer key, String value) { // When StreamsBuilder builder = new StreamsBuilder(); - KStream<Integer, String> stream =
builder.stream(topicName, Consumed.with(intSerde, stringSerde)); + KStream<Integer, String> stream = builder.stream(topicName,
Consumed.with(Serdes.Integer(), Serdes.String())); stream.foreach(action); // Then - driver = new TopologyTestDriver(builder.build(),
props); - for (KeyValue<Integer, String> record: inputRecords) { - driver.pipeInput(recordFactory.create(topicName, record.key,
record.value)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + for (KeyValue<Integer,
String> record : inputRecords) { + driver.pipeInput(recordFactory.create(topicName, record.key, record.value)); + } }
assertEquals(expectedRecords.size(), actualRecords.size()); diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableJoinTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableJoinTest.java index 6e5b816f5ed..c37e8a954ea
100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableJoinTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableJoinTest.java @@ -17,22 +17,20 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; -import
org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.GlobalKTable; import org.apache.kafka.streams.kstream.KStream; import
org.apache.kafka.streams.kstream.KeyValueMapper; -import org.apache.kafka.test.MockProcessor; import
org.apache.kafka.streams.test.ConsumerRecordFactory; +import org.apache.kafka.test.MockProcessor; import
org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; -import org.apache.kafka.test.TestUtils;
+import org.apache.kafka.test.StreamsTestUtils; import org.junit.After; import org.junit.Before; import org.junit.Test; @@ -47,8 +45,6
@@ private final String streamTopic = "streamTopic"; private final String globalTableTopic = "globalTableTopic"; - private final
Serde<Integer> intSerde = Serdes.Integer(); - private final Serde<String> stringSerde = Serdes.String(); private TopologyTestDriver
driver; private MockProcessor<Integer, String> processor; private final int[] expectedKeys = {0, 1, 2, 3}; @@ -63,8 +59,8 @@ public
void setUp() { final KeyValueMapper<Integer, String, String> keyMapper; final MockProcessorSupplier<Integer, String> supplier = new
MockProcessorSupplier<>(); - final Consumed<Integer, String> streamConsumed = Consumed.with(intSerde, stringSerde); - final
Consumed<String, String> tableConsumed = Consumed.with(stringSerde, stringSerde); + final Consumed<Integer, String>
```

streamConsumed = Consumed.with(Serdes.Integer(), Serdes.String()); + final Consumed<String, String> tableConsumed = Consumed.with(Serdes.String(), Serdes.String()); stream = builder.stream(streamTopic, streamConsumed); table = builder.globalTable(globalTableTopic, tableConsumed); keyMapper = new KeyValueMapper<Integer, String, String>() { @@ -78,13 +74,7 @@ public String apply(final Integer key, final String value) { }; stream.join(table, keyMapper, MockValueJoiner.TOSTRING_JOINER).process(supplier); - final Properties props = new Properties(); - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-global-ktable-join-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - + final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); driver = new TopologyTestDriver(builder.build(), props); processor = supplier.theCapturedProcessor(); @@ -92,10 +82,7 @@ public String apply(final Integer key, final String value) { @After public void cleanup() { - if (driver != null) { - driver.close(); - } - driver = null; + driver.close(); } private void pushToStream(final int messageCount, final String valuePrefix, final boolean includeForeignKey) { diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableLeftJoinTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableLeftJoinTest.java index b3551baf961..eb0775a0847 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableLeftJoinTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamGlobalKTableLeftJoinTest.java @@ -17,13 +17,11 @@ package org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; -import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.GlobalKTable; import org.apache.kafka.streams.kstream.KStream; @@ -32,7 +30,8 @@ import org.apache.kafka.test.MockProcessor; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; -import org.apache.kafka.test.TestUtils; +import org.apache.kafka.test.StreamsTestUtils; +import org.junit.After; import org.junit.Before; import org.junit.Test; @@ -46,8 +45,6 @@ final private String streamTopic = "streamTopic"; final private String globalTableTopic = "globalTableTopic"; - final private Serde<Integer> intSerde = Serdes.Integer(); - final private Serde<String> stringSerde = Serdes.String(); private MockProcessor<Integer, String> processor; private TopologyTestDriver driver; @@ -64,8 +61,8 @@ public void setUp() { final KeyValueMapper<Integer, String, String> keyMapper; final MockProcessorSupplier<Integer, String> supplier = new MockProcessorSupplier<>(); - final Consumed<Integer, String> streamConsumed = Consumed.with(intSerde, stringSerde); - final Consumed<String, String> tableConsumed = Consumed.with(stringSerde, stringSerde); + final Consumed<Integer, String> streamConsumed = Consumed.with(Serdes.Integer(), Serdes.String()); + final Consumed<String, String> tableConsumed = Consumed.with(Serdes.String(), Serdes.String()); stream = builder.stream(streamTopic, streamConsumed); table = builder.globalTable(globalTableTopic, tableConsumed); keyMapper = new KeyValueMapper<Integer, String, String>() { @@ -79,18 +76,17 @@ public String apply(final Integer key, final String value) { }; stream.leftJoin(table, keyMapper, MockValueJoiner.TOSTRING_JOINER).process(supplier); - final Properties props = new Properties(); - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-global-ktable-left-join-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - + final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); driver = new TopologyTestDriver(builder.build(), props); processor = supplier.theCapturedProcessor(); } + @After + public void cleanup() { + driver.close(); + } + private void pushToStream(final int messageCount, final String valuePrefix, final boolean includeForeignKey) { final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); for (int i = 0; i < messageCount; i++) { diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamImplTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamImplTest.java index 797575d8bb1..49e8aaae026 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamImplTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamImplTest.java @@ -16,7 +16,6 @@ */ package org.apache.kafka.streams.kstream.internals; -import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Utils; @@ -24,7 +23,6 @@ import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; -import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.errors.TopologyException; import org.apache.kafka.streams.kstream.GlobalKTable; @@ -49,8 +47,7 @@ import org.apache.kafka.test.MockMapper; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; -import org.apache.kafka.test.TestUtils; -import org.junit.After; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Before; import org.junit.Test; @@ -69,48 +66,28 @@ public class KStreamImplTest { - private final Serde<String> stringSerde = Serdes.String(); - private final Serde<Integer> intSerde = Serdes.Integer(); - private final Consumed<String, String> consumed = Consumed.with(stringSerde, stringSerde); private final Consumed<String, String> stringConsumed = Consumed.with(Serdes.String(), Serdes.String()); - private final MockProcessorSupplier<String, String> processorSupplier = new MockProcessorSupplier<>(); private KStream<String, String> testStream; private StreamsBuilder builder; private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); @Before public void before() { builder = new StreamsBuilder(); testStream = builder.stream("source"); - - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-impl-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After - public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; } @Test public void testNumProcesses() { final StreamsBuilder builder = new StreamsBuilder(); - KStream<String, String> source1 = builder.stream(Arrays.asList("topic-1", "topic-2"), consumed); + KStream<String, String> source1 = builder.stream(Arrays.asList("topic-1", "topic-2"), stringConsumed); -

KStream<String, String> source2 = builder.stream(Arrays.asList("topic-3", "topic-4"), consumed); + KStream<String, String> source2 = builder.stream(Arrays.asList("topic-3", "topic-4"), stringConsumed); KStream<String, String> stream1 = source1.filter(new Predicate<String, String>() { @@ -170,7 +147,7 @@ public boolean test(String key, Integer value) {); final int anyWindowSize = 1; - final Joined<String, Integer, Integer> joined = Joined.with(stringSerde, intSerde, intSerde); + final Joined<String, Integer, Integer> joined = Joined.with(Serdes.String(), Serdes.Integer(), Serdes.Integer()); KStream<String, Integer> stream4 = streams2[0].join(streams3[0], new ValueJoiner<Integer, Integer, Integer>() { @Override public Integer apply(Integer value1, Integer value2) { @@ -205,9 +182,8 @@ public Integer apply(Integer value1, Integer value2) { @Test public void shouldUseRecordMetadataTimestampExtractorWithThrough() { final StreamsBuilder builder = new StreamsBuilder(); - final Consumed<String, String> consumed = Consumed.with(stringSerde, stringSerde); - KStream<String, String> stream1 = builder.stream(Arrays.asList("topic-1", "topic-2"), consumed); - KStream<String, String> stream2 = builder.stream(Arrays.asList("topic-3", "topic-4"), consumed); + KStream<String, String> stream1 = builder.stream(Arrays.asList("topic-1", "topic-2"), stringConsumed); + KStream<String, String> stream2 = builder.stream(Arrays.asList("topic-3", "topic-4"), stringConsumed); stream1.to("topic-5"); stream2.through("topic-6"); @@ -224,11 +200,12 @@ public void shouldUseRecordMetadataTimestampExtractorWithThrough() { public void shouldSendDataThroughTopicUsingProduced() { final StreamsBuilder builder = new StreamsBuilder(); final String input = "topic"; - final KStream<String, String> stream = builder.stream(input, consumed); - stream.through("through-topic", Produced.with(stringSerde, stringSerde)).process(processorSupplier); + final KStream<String, String> stream = builder.stream(input, stringConsumed); + stream.through("through-topic", Produced.with(Serdes.String(), Serdes.String())).process(processorSupplier); - driver = new TopologyTestDriver(builder.build(), props); - driver.pipeInput(recordFactory.create(input, "a", "b")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(input, "a", "b")); + } assertThat(processorSupplier.theCapturedProcessor().processed, equalTo(Collections.singletonList("a:b"))); } @@ -236,12 +213,13 @@ public void shouldSendDataThroughTopicUsingProduced() { public void shouldSendDataToTopicUsingProduced() { final StreamsBuilder builder = new StreamsBuilder(); final String input = "topic"; - final KStream<String, String> stream = builder.stream(input, consumed); - stream.to("to-topic", Produced.with(stringSerde, stringSerde)); - builder.stream("to-topic", consumed).process(processorSupplier); + final KStream<String, String> stream = builder.stream(input, stringConsumed); + stream.to("to-topic", Produced.with(Serdes.String(), Serdes.String())); + builder.stream("to-topic", stringConsumed).process(processorSupplier); - driver = new TopologyTestDriver(builder.build(), props); - driver.pipeInput(recordFactory.create(input, "e", "f")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(input, "e", "f")); + } assertThat(processorSupplier.theCapturedProcessor().processed, equalTo(Collections.singletonList("e:f"))); } @@ -249,7 +227,7 @@ public void shouldSendDataToTopicUsingProduced() { // TODO: this test should be refactored when we removed KStreamBuilder so that the created Topology contains internal topics as well public void shouldUseRecordMetadataTimestampExtractorWhenInternalRepartitioningTopicCreated() { final KStreamBuilder builder = new KStreamBuilder(); - KStream<String, String> kStream = builder.stream(stringSerde, stringSerde, "topic-1"); + KStream<String, String> kStream = builder.stream(Serdes.String(), Serdes.String(), "topic-1"); ValueJoiner<String, String, String> valueJoiner = MockValueJoiner.instance(":"); long windowSize = TimeUnit.MILLISECONDS.convert(1, TimeUnit.DAYS); final KStream<String, String> stream = kStream @@ -282,9 +260,8 @@ public void shouldUseRecordMetadataTimestampExtractorWhenInternalRepartitioningT @Test public void testToWithNullValueSerdeDoesntNPE() { final StreamsBuilder builder = new StreamsBuilder(); - final Consumed<String, String> consumed = Consumed.with(stringSerde, stringSerde); - final KStream<String, String> inputStream = builder.stream(Collections.singleton("input"), consumed); - inputStream.to(stringSerde, null, "output"); + final KStream<String, String> inputStream = builder.stream(Collections.singleton("input"), stringConsumed); + inputStream.to(Serdes.String(), null, "output"); } @Test(expected = NullPointerException.class) @@ -477,7 +454,7 @@ public void shouldThrowNullPointerOnToWhenProducedIsNull() { @Test public void shouldThrowNullPointerOnLeftJoinWithTableWhenJoinedIsNull() { - final KTable<String, String> table = builder.table("blah", consumed); + final KTable<String, String> table = builder.table("blah", stringConsumed); try { testStream.leftJoin(table, MockValueJoiner.TOSTRING_JOINER, @@ -490,7 +467,7 @@ public void shouldThrowNullPointerOnLeftJoinWithTableWhenJoinedIsNull() { @Test public void shouldThrowNullPointerOnJoinWithTableWhenJoinedIsNull() { - final KTable<String, String> table = builder.table("blah", consumed); + final KTable<String, String> table = builder.table("blah", stringConsumed); try { testStream.join(table, MockValueJoiner.TOSTRING_JOINER, @@ -522,12 +499,12 @@ public void shouldMergeTwoStreams() { merged.process(processorSupplier); - driver = new TopologyTestDriver(builder.build(), props); - - driver.pipeInput(recordFactory.create(topic1, "A", "aa")); - driver.pipeInput(recordFactory.create(topic2, "B", "bb")); - driver.pipeInput(recordFactory.create(topic2, "C", "cc")); - driver.pipeInput(recordFactory.create(topic1, "D", "dd")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(topic1, "A", "aa")); + driver.pipeInput(recordFactory.create(topic2, "B", "bb")); + driver.pipeInput(recordFactory.create(topic2, "C", "cc")); + driver.pipeInput(recordFactory.create(topic1, "D", "dd")); + } assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd"), processorSupplier.theCapturedProcessor().processed); } @@ -547,16 +524,16 @@ public void shouldMergeMultipleStreams() { merged.process(processorSupplier); - driver = new TopologyTestDriver(builder.build(), props); - - driver.pipeInput(recordFactory.create(topic1, "A", "aa")); - driver.pipeInput(recordFactory.create(topic2, "B", "bb")); - driver.pipeInput(recordFactory.create(topic3, "C", "cc")); - driver.pipeInput(recordFactory.create(topic4, "D", "dd")); - driver.pipeInput(recordFactory.create(topic4, "E", "ee")); - driver.pipeInput(recordFactory.create(topic3, "F", "ff")); - driver.pipeInput(recordFactory.create(topic2, "G", "gg")); - driver.pipeInput(recordFactory.create(topic1, "H", "hh")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(topic1, "A", "aa")); + driver.pipeInput(recordFactory.create(topic2, "B", "bb")); + driver.pipeInput(recordFactory.create(topic3, "C", "cc")); + driver.pipeInput(recordFactory.create(topic4, "D", "dd")); + driver.pipeInput(recordFactory.create(topic4, "E", "ee")); + driver.pipeInput(recordFactory.create(topic3, "F", "ff")); + driver.pipeInput(recordFactory.create(topic2, "G", "gg")); + driver.pipeInput(recordFactory.create(topic1, "H", "hh")); + } assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd", "E:ee", "F:ff", "G:gg", "H:hh"), processorSupplier.theCapturedProcessor().processed); @@ -568,13 +545,13 @@ public void shouldProcessFromSourceThatMatchPattern() { pattern2Source.process(processorSupplier); - driver = new TopologyTestDriver(builder.build(), props); - - driver.pipeInput(recordFactory.create("topic-3", "A", "aa")); - driver.pipeInput(recordFactory.create("topic-4", "B", "bb")); - driver.pipeInput(recordFactory.create("topic-5", "C", "cc")); - driver.pipeInput(recordFactory.create("topic-6", "D", "dd")); - driver.pipeInput(recordFactory.create("topic-7", "E", "ee")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create("topic-3", "A", "aa")); + driver.pipeInput(recordFactory.create("topic-4", "B", "bb")); + driver.pipeInput(recordFactory.create("topic-5", "C", "cc")); + driver.pipeInput(recordFactory.create("topic-6", "D", "dd")); + driver.pipeInput(recordFactory.create("topic-7", "E", "ee")); + } assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd", "E:ee"), processorSupplier.theCapturedProcessor().processed); @@ -591,13 +568,13 @@ public void shouldProcessFromSourcesThatMatchMultiplePattern() { merged.process(processorSupplier); - driver = new TopologyTestDriver(builder.build(), props); - - driver.pipeInput(recordFactory.create("topic-3", "A", "aa")); -

```
driver.pipeInput(recordFactory.create("topic-4", "B", "bb")); - driver.pipeInput(recordFactory.create("topic-A", "C", "cc")); -
driver.pipeInput(recordFactory.create("topic-Z", "D", "dd")); - driver.pipeInput(recordFactory.create(topic3, "E", "ee")); + try (final
TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create("topic-3", "A",
"aa")); + driver.pipeInput(recordFactory.create("topic-4", "B", "bb")); + driver.pipeInput(recordFactory.create("topic-A", "C", "cc")); +
driver.pipeInput(recordFactory.create("topic-Z", "D", "dd")); + driver.pipeInput(recordFactory.create(topic3, "E", "ee")); + }
assertEquals(Utils.mkList("A:aa", "B:bb", "C:cc", "D:dd", "E:ee"), processorSupplier.theCapturedProcessor().processed); diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamJoinTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamJoinTest.java index 5d849eee5d4..de3446c1a08
100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamJoinTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamJoinTest.java @@ -17,26 +17,22 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; -import
org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.JoinWindows; import org.apache.kafka.streams.kstream.Joined; import
org.apache.kafka.streams.kstream.KStream; -import org.apache.kafka.test.MockProcessor; import
org.apache.kafka.streams.kstream.ValueJoiner; import org.apache.kafka.streams.processor.internals.testutil.LogCaptureAppender; import
org.apache.kafka.streams.test.ConsumerRecordFactory; +import org.apache.kafka.test.MockProcessor; import
org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; -import org.apache.kafka.test.TestUtils; -
import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import
java.util.Arrays; @@ -55,38 +51,16 @@ final private String topic1 = "topic1"; final private String topic2 = "topic2"; - final private
Serde<Integer> intSerde = Serdes.Integer(); - final private Serde<String> stringSerde = Serdes.String(); - - private final
Consumed<Integer, String> consumed = Consumed.with(intSerde, stringSerde); + private final Consumed<Integer, String> consumed =
Consumed.with(Serdes.Integer(), Serdes.String()); private final ConsumerRecordFactory<Integer, String> recordFactory = new
ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); - private TopologyTestDriver driver; - private final Properties
props = new Properties(); - - @Before - public void setUp() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG,
"kstream-kstream-join-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); -
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After -
public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props =
StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); @Test public void
shouldLogAndMeterOnSkippedRecordsWithNullValue() { final StreamsBuilder builder = new StreamsBuilder(); - final KStream<String,
Integer> left = builder.stream("left", Consumed.with(stringSerde, intSerde)); - final KStream<String, Integer> right =
builder.stream("right", Consumed.with(stringSerde, intSerde)); + final KStream<String, Integer> left = builder.stream("left",
Consumed.with(Serdes.String(), Serdes.Integer())); + final KStream<String, Integer> right = builder.stream("right",
Consumed.with(Serdes.String(), Serdes.Integer())); final ConsumerRecordFactory<String, Integer> recordFactory = new
ConsumerRecordFactory<>(new StringSerializer(), new IntegerSerializer()); left.join( @@ -98,17 +72,18 @@ public Integer apply(final
Integer value1, final Integer value2) { } }, JoinWindows.of(100), - Joined.with(stringSerde, intSerde, intSerde) +
Joined.with(Serdes.String(), Serdes.Integer(), Serdes.Integer()) ); final LogCaptureAppender appender =
LogCaptureAppender.createAndRegister(); - driver = new TopologyTestDriver(builder.build(), props); -
driver.pipeInput(recordFactory.create("left", "A", null)); - LogCaptureAppender.unregister(appender); + try (final TopologyTestDriver
driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create("left", "A", null)); +
LogCaptureAppender.unregister(appender); - assertThat(appender.getMessages(), hasItem("Skipping record due to null key or value. key=
[A] value=[null] topic=[left] partition=[0] offset=[0]")); + assertThat(appender.getMessages(), hasItem("Skipping record due to null key
or value. key=[A] value=[null] topic=[left] partition=[0] offset=[0]")); - assertEquals(1.0, getMetricByName(driver.metrics(), "skipped-
records-total", "stream-metrics").metricValue()); + assertEquals(1.0, getMetricByName(driver.metrics(), "skipped-records-total", "stream-
metrics").metricValue()); + } } @Test @@ -127,7 +102,7 @@ public void testJoin() { stream2, MockValueJoiner.TOSTRING_JOINER,
JoinWindows.of(100), - Joined.with(intSerde, stringSerde, stringSerde)); + Joined.with(Serdes.Integer(), Serdes.String(), Serdes.String()));
joined.process(supplier); final Collection<Set<String>> copartitionGroups = StreamsBuilderTest.getCopartitionedGroups(builder); @@
-135,81 +110,82 @@ public void testJoin() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>
(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver = new TopologyTestDriver(builder.build(), props); + try (final
TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { - final MockProcessor<Integer, String> processor =
supplier.theCapturedProcessor(); + final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); - // push two items
to the primary stream. the other window is empty - // w1 = {} - // w2 = {} - // --> w1 = { 0:X0, 1:X1 } - // w2 = {} + // push two items to
the primary stream. the other window is empty + // w1 = {} + // w2 = {} + // --> w1 = { 0:X0, 1:X1 } + // w2 = {} - for (int i = 0; i < 2;
i++) { - driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); - } + for (int i = 0; i < 2; i++) { +
driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); + } - processor.checkAndClearProcessResult(); +
processor.checkAndClearProcessResult(); - // push two items to the other stream. this should produce two items. - // w1 = { 0:X0, 1:X1 } -
// w2 = {} - // --> w1 = { 0:X0, 1:X1 } - // w2 = { 0:Y0, 1:Y1 } + // push two items to the other stream. this should produce two items. + //
w1 = { 0:X0, 1:X1 } + // w2 = {} + // --> w1 = { 0:X0, 1:X1 } + // w2 = { 0:Y0, 1:Y1 } - for (int i = 0; i < 2; i++) { -
driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i])); - } + for (int i = 0; i < 2; i++) { +
driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i])); + } -
processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); + processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); - //
push all four items to the primary stream. this should produce two items. - // w1 = { 0:X0, 1:X1 } - // w2 = { 0:Y0, 1:Y1 } - // --> w1 = {
0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } - // w2 = { 0:Y0, 1:Y1 } + // push all four items to the primary stream. this should produce two
items. + // w1 = { 0:X0, 1:X1 } + // w2 = { 0:Y0, 1:Y1 } + // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } + // w2 = { 0:Y0, 1:Y1 } -
for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "X" + expectedKey)); - } + for (int
expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "X" + expectedKey)); + } -
processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); + processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); - //
push all items to the other stream. this should produce six items. - // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } - // w2 = { 0:Y0, 1:Y1 }
- // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } + // push all items to the
other stream. this should produce six items. + // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } + // w2 = { 0:Y0, 1:Y1 } + // --> w1 = {
0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } - for (int expectedKey : expectedKeys) { -
driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey)); - } + for (int expectedKey : expectedKeys) { +
```

driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey)); + } - processor.checkAndClearProcessResult("0:X0+YY0", "0:X0+YY0", "1:X1+YY1", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); + processor.checkAndClearProcessResult("0:X0+YY0", "0:X0+YY0", "1:X1+YY1", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - // push all four items to the primary stream. this should produce six items. - // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 - // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } + // push all four items to the primary stream. this should produce six items. + // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 + // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey)); - } + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey)); + } - processor.checkAndClearProcessResult("0:XX0+Y0", "0:XX0+YY0", "1:XX1+Y1", "1:XX1+YY1", "2:XX2+YY2", "3:XX3+YY3"); + processor.checkAndClearProcessResult("0:XX0+Y0", "0:XX0+YY0", "1:XX1+Y1", "1:XX1+YY1", "2:XX2+YY2", "3:XX3+YY3"); - // push two items to the other stream. this should produce six item. - // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 - // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3, 0:YYY0, 1:YYY1 } + // push two items to the other stream. this should produce six item. + // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 + // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3, 0:YYY0, 1:YYY1 } - for (int i = 0; i < 2; i++) { - driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "YYY" + expectedKeys[i])); - } + for (int i = 0; i < 2; i++) { + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "YYY" + expectedKeys[i])); + } - processor.checkAndClearProcessResult("0:X0+YYY0", "0:X0+YYY0", "0:XX0+YYY0", "1:X1+YYY1", "1:X1+YYY1", "1:XX1+YYY1"); + processor.checkAndClearProcessResult("0:X0+YYY0", "0:X0+YYY0", "0:XX0+YYY0", "1:X1+YYY1", "1:X1+YYY1", "1:XX1+YYY1"); + } } @Test @@ -229,88 +205,89 @@ public void testOuterJoin() { stream2, MockValueJoiner.TOSTRING_JOINER, JoinWindows.of(100), - Joined.with(intSerde, stringSerde, stringSerde)); + Joined.with(Serdes.Integer(), Serdes.String(), Serdes.String())); joined.process(supplier); final Collection<Set<String>> copartitionGroups = StreamsBuilderTest.getCopartitionedGroups(builder); assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver = new TopologyTestDriver(builder.build(), props, 0L); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { - final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); + final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); - // push two items to the primary stream. the other window is empty.this should produce two items - // w1 = {} - // w2 = {} - // --> w1 = { 0:X0, 1:X1 } - // w2 = {} + // push two items to the primary stream. the other window is empty.this should produce two items + // w1 = {} + // w2 = {} + // --> w1 = { 0:X0, 1:X1 } + // w2 = {} - for (int i = 0; i < 2; i++) { - driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); - } + for (int i = 0; i < 2; i++) { + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); + } - processor.checkAndClearProcessResult("0:X0+null", "1:X1+null"); + processor.checkAndClearProcessResult("0:X0+null", "1:X1+null"); - // push two items to the other stream. this should produce two items. - // w1 = { 0:X0, 1:X1 } - // w2 = {} - // --> w1 = { 0:X0, 1:X1 } - // w2 = { 0:Y0, 1:Y1 } + // push two items to the other stream. this should produce two items. + // w1 = { 0:X0, 1:X1 } + // w2 = {} + // --> w1 = { 0:X0, 1:X1 } + // w2 = { 0:Y0, 1:Y1 } - for (int i = 0; i < 2; i++) { - driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i])); - } + for (int i = 0; i < 2; i++) { + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i])); + } - processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); + processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); - // push all four items to the primary stream. this should produce four items. - // w1 = { 0:X0, 1:X1 } - // w2 = { 0:Y0, 1:Y1 } - // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } - // w2 = { 0:Y0, 1:Y1 } + // push all four items to the primary stream. this should produce four items. + // w1 = { 0:X0, 1:X1 } + // w2 = { 0:Y0, 1:Y1 } + // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } + // w2 = { 0:Y0, 1:Y1 } - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "X" + expectedKey)); - } + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "X" + expectedKey)); + } - processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1", "2:X2+null", "3:X3+null"); + processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1", "2:X2+null", "3:X3+null"); - // push all items to the other stream. this should produce six items. - // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } - // w2 = { 0:Y0, 1:Y1 } - // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } + // push all items to the other stream. this should produce six items. + // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } + // w2 = { 0:Y0, 1:Y1 } + // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey)); - } + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey)); + } - processor.checkAndClearProcessResult("0:X0+YY0", "0:X0+YY0", "1:X1+YY1", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); + processor.checkAndClearProcessResult("0:X0+YY0", "0:X0+YY0", "1:X1+YY1", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - // push all four items to the primary stream. this should produce six items. - // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 - // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } + // push all four items to the primary stream. this should produce six items. + // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 + // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 } - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey)); - } + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey)); + } - processor.checkAndClearProcessResult("0:XX0+Y0", "0:XX0+YY0", "1:XX1+Y1", "1:XX1+YY1", "2:XX2+YY2", "3:XX3+YY3"); + processor.checkAndClearProcessResult("0:XX0+Y0", "0:XX0+YY0", "1:XX1+Y1", "1:XX1+YY1", "2:XX2+YY2", "3:XX3+YY3"); - // push two items to the other stream. this should produce six item. - // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 - // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3, 0:YYY0, 1:YYY1 } + // push two items to the other stream. this should produce six item. + // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3 + // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 3:X3, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 0:YY0, 1:YY1, 2:YY2, 3:YY3, 0:YYY0, 1:YYY1 } - for (int i = 0; i < 2; i++) { - driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "YYY" + expectedKeys[i])); - } + for (int i = 0; i < 2; i++) { + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "YYY" + expectedKeys[i])); + } - processor.checkAndClearProcessResult("0:X0+YYY0", "0:X0+YYY0", "0:XX0+YYY0", "1:X1+YYY1", "1:X1+YYY1", "1:XX1+YYY1"); + processor.checkAndClearProcessResult("0:X0+YYY0", "0:X0+YYY0", "0:XX0+YYY0", "1:X1+YYY1", "1:X1+YYY1", "1:XX1+YYY1"); + } } @Test @@ -332,7 +309,7 @@ public void testWindowing() { stream2, MockValueJoiner.TOSTRING_JOINER, JoinWindows.of(100), - Joined.with(intSerde, stringSerde, stringSerde)); +

```
Joined.with(Serdes.Integer(), Serdes.String(), Serdes.String())); joined.process(supplier); final Collection<Set<String>> copartitionGroups
= StreamsBuilderTest.getCopartitionedGroups(builder); @@ -340,197 +317,198 @@ public void testWindowing() { assertEquals(1,
copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver = new
TopologyTestDriver(builder.build(), props, time); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props,
time)) { - // push two items to the primary stream. the other window is empty. this should produce no items. - // w1 = {} - // w2 = {} - // --
> w1 = { 0:X0, 1:X1 } - // w2 = {} - for (int i = 0; i < 2; i++) { - driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" +
expectedKeys[i], time)); - } + // push two items to the primary stream. the other window is empty. this should produce no items. + // w1 =
{} + // w2 = {} + // --> w1 = { 0:X0, 1:X1 } + // w2 = {} + for (int i = 0; i < 2; i++) { + driver.pipeInput(recordFactory.create(topic1,
expectedKeys[i], "X" + expectedKeys[i], time)); + } - final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor();
+ final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); - processor.checkAndClearProcessResult(); +
processor.checkAndClearProcessResult(); - // push two items to the other stream. this should produce two items. - // w1 = { 0:X0, 1:X1 } -
// w2 = {} - // --> w1 = { 0:X0, 1:X1 } - // w2 = { 0:Y0, 1:Y1 } + // push two items to the other stream. this should produce two items. + //
w1 = { 0:X0, 1:X1 } + // w2 = {} + // --> w1 = { 0:X0, 1:X1 } + // w2 = { 0:Y0, 1:Y1 } - for (int i = 0; i < 2; i++) { -
driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i], time)); - } + for (int i = 0; i < 2; i++) { +
driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i], time)); + } -
processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); + processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); - //
clear logically - time = 1000L; - for (int i = 0; i < expectedKeys.length; i++) { - driver.pipeInput(recordFactory.create(topic1,
expectedKeys[i], "X" + expectedKeys[i], time + i)); - } - processor.checkAndClearProcessResult(); + // clear logically + time = 1000L; +
for (int i = 0; i < expectedKeys.length; i++) { + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i],
time + i)); + } + processor.checkAndClearProcessResult(); - // gradually expires items in w1 - // w1 = { 0:X0, 1:X1, 2:X2, 3:X3 } + //
gradually expires items in w1 + // w1 = { 0:X0, 1:X1, 2:X2, 3:X3 } - time += 100L; - for (int expectedKey : expectedKeys) { -
driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + time += 100L; + for (int expectedKey :
expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } -
processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); +
processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - time += 1L; - for (int expectedKey :
expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + time += 1L; + for (int
expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } -
processor.checkAndClearProcessResult("1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); +
processor.checkAndClearProcessResult("1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - time += 1L; - for (int expectedKey : expectedKeys)
{ - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + time += 1L; + for (int expectedKey :
expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } -
processor.checkAndClearProcessResult("2:X2+YY2", "3:X3+YY3"); + processor.checkAndClearProcessResult("2:X2+YY2",
"3:X3+YY3"); - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY"
+ expectedKey, time)); - } + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2,
expectedKey, "YY" + expectedKey, time)); + } - processor.checkAndClearProcessResult("3:X3+YY3"); +
processor.checkAndClearProcessResult("3:X3+YY3"); - time += 1L; - for (int expectedKey : expectedKeys) { -
driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + time += 1L; + for (int expectedKey :
expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } -
processor.checkAndClearProcessResult(); + processor.checkAndClearProcessResult(); - // go back to the time before expiration + // go
back to the time before expiration - time = 1000L - 100L - 1L; - for (int expectedKey : expectedKeys) { -
driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + time = 1000L - 100L - 1L; + for (int
expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } -
processor.checkAndClearProcessResult(); + processor.checkAndClearProcessResult(); - time += 1L; - for (int expectedKey :
expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + time += 1L; + for (int
expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } -
processor.checkAndClearProcessResult("0:X0+YY0"); + processor.checkAndClearProcessResult("0:X0+YY0"); - time += 1L; - for (int
expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + time +=
1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time));
+ } - processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1"); + processor.checkAndClearProcessResult("0:X0+YY0",
"1:X1+YY1"); - time += 1; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" +
expectedKey, time)); - } + time += 1; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2,
expectedKey, "YY" + expectedKey, time)); + } - processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2"); +
processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2"); - time += 1; - for (int expectedKey : expectedKeys) {
- driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + time += 1; + for (int expectedKey :
expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } -
processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); +
processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - // clear (logically) - time = 2000L; -
for (int i = 0; i < expectedKeys.length; i++) { - driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i],
time + i)); - } + // clear (logically) + time = 2000L; + for (int i = 0; i < expectedKeys.length; i++) { +
driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i], time + i)); + } -
processor.checkAndClearProcessResult(); + processor.checkAndClearProcessResult(); - // gradually expires items in w2 - // w2 = { 0:Y0,
1:Y1, 2:Y2, 3:Y3 } + // gradually expires items in w2 + // w2 = { 0:Y0, 1:Y1, 2:Y2, 3:Y3 } - time = 2000L + 100L; - for (int expectedKey
: expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } + time = 2000L + 100L;
+ for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } -
processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); +
processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); - time += 1L; - for (int expectedKey :
expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } + time += 1L; + for (int
expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } -
processor.checkAndClearProcessResult("1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); +
processor.checkAndClearProcessResult("1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); - time += 1L; - for (int expectedKey : expectedKeys)
{ - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } + time += 1L; + for (int expectedKey :
expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } -
processor.checkAndClearProcessResult("2:XX2+Y2", "3:XX3+Y3"); + processor.checkAndClearProcessResult("2:XX2+Y2",
"3:XX3+Y3"); - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX"
+ expectedKey, time)); - } + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1,
expectedKey, "XX" + expectedKey, time)); + } - processor.checkAndClearProcessResult("3:XX3+Y3"); +
```

processor.checkAndClearProcessResult("3:XX3+Y3"); - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } - processor.checkAndClearProcessResult(); + processor.checkAndClearProcessResult(); - // go back to the time before expiration + // go back to the time before expiration - time = 2000L - 100L - 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } + time = 2000L - 100L - 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } - processor.checkAndClearProcessResult(); + processor.checkAndClearProcessResult(); - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } - processor.checkAndClearProcessResult("0:XX0+Y0"); + processor.checkAndClearProcessResult("0:XX0+Y0"); - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } - processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1"); + processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1"); - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } - processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2"); + processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2"); - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } - processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); + processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); + } } @Test @@ -552,9 +530,9 @@ public void testAsymmetricWindowingAfter() { stream2, MockValueJoiner.TOSTRING_JOINER, JoinWindows.of(0).after(100), - Joined.with(intSerde, - stringSerde, - stringSerde)); + Joined.with(Serdes.Integer(), + Serdes.String(), + Serdes.String())); joined.process(supplier); final Collection<Set<String>> copartitionGroups = StreamsBuilderTest.getCopartitionedGroups(builder); @@ -562,85 +540,85 @@ public void testAsymmetricWindowingAfter() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver = new TopologyTestDriver(builder.build(), props, time); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, time)) { - final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); + final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); - for (int i = 0; i < expectedKeys.length; i++) { - driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i], time + i)); - } - processor.checkAndClearProcessResult(); + for (int i = 0; i < expectedKeys.length; i++) { + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i], time + i)); + } + processor.checkAndClearProcessResult(); + time = 1000L - 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time = 1000L - 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult(); - processor.checkAndClearProcessResult(); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult("0:X0+YY0"); - processor.checkAndClearProcessResult("0:X0+YY0"); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1"); - processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1"); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2"); - processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2"); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); + time = 1000 + 100L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time = 1000 + 100L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult("1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); - processor.checkAndClearProcessResult("1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult("2:X2+YY2", "3:X3+YY3"); - processor.checkAndClearProcessResult("2:X2+YY2", "3:X3+YY3"); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult("3:X3+YY3"); - processor.checkAndClearProcessResult("3:X3+YY3"); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + processor.checkAndClearProcessResult(); } - - processor.checkAndClearProcessResult(); } @Test @@ -663,7 +641,7 @@ public void testAsymmetricWindowingBefore() { stream2, MockValueJoiner.TOSTRING_JOINER, JoinWindows.of(0).before(100), - Joined.with(intSerde, stringSerde, stringSerde)); + Joined.with(Serdes.Integer(), Serdes.String(), Serdes.String())); joined.process(supplier); final Collection<Set<String>> copartitionGroups = StreamsBuilderTest.getCopartitionedGroups(builder); @@ -671,84 +649,84 @@ public void testAsymmetricWindowingBefore() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver = new TopologyTestDriver(builder.build(), props, time); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, time)) { - final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); + final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); - for (int i = 0; i < expectedKeys.length; i++) { - driver.pipeInput(recordFactory.create(topic1,

```
expectedKeys[i], "X" + expectedKeys[i], time + i)); - } - processor.checkAndClearProcessResult(); + for (int i = 0; i <
expectedKeys.length; i++) { + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i], time + i)); + } +
processor.checkAndClearProcessResult(); + time = 1000L - 100L - 1L; + for (int expectedKey : expectedKeys) { +
driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time = 1000L - 100L - 1L; - for (int
expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } +
processor.checkAndClearProcessResult(); - processor.checkAndClearProcessResult(); + time += 1L; + for (int expectedKey :
expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int
expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } +
processor.checkAndClearProcessResult("0:X0+YY0"); - processor.checkAndClearProcessResult("0:X0+YY0"); + time += 1L; + for (int
expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time +=
1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); -
} + processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1"); - processor.checkAndClearProcessResult("0:X0+YY0",
"1:X1+YY1"); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey,
"YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2,
expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2"); -
processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2"); + time += 1L; + for (int expectedKey :
expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int
expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } +
processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); -
processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); + time = 1000L; + for (int
expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time =
1000L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey,
time)); - } + processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); -
processor.checkAndClearProcessResult("0:X0+YY0", "1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); + time += 1L; + for (int expectedKey :
expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int
expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } +
processor.checkAndClearProcessResult("1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); -
processor.checkAndClearProcessResult("1:X1+YY1", "2:X2+YY2", "3:X3+YY3"); + time += 1L; + for (int expectedKey :
expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int
expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); - } +
processor.checkAndClearProcessResult("2:X2+YY2", "3:X3+YY3"); - processor.checkAndClearProcessResult("2:X2+YY2",
"3:X3+YY3"); + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey,
"YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2,
expectedKey, "YY" + expectedKey, time)); - } + processor.checkAndClearProcessResult("3:X3+YY3"); -
processor.checkAndClearProcessResult("3:X3+YY3"); + time += 1L; + for (int expectedKey : expectedKeys) { +
driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); + } - time += 1L; - for (int expectedKey :
expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey, time)); +
processor.checkAndClearProcessResult(); } - - processor.checkAndClearProcessResult(); } } diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamLeftJoinTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamLeftJoinTest.java index c67e13df223..11c5c5b9852
100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamLeftJoinTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKStreamLeftJoinTest.java @@ -17,24 +17,20 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; -import
org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.JoinWindows; import org.apache.kafka.streams.kstream.Joined; import
org.apache.kafka.streams.kstream.KStream; -import org.apache.kafka.test.MockProcessor; import
org.apache.kafka.streams.test.ConsumerRecordFactory; +import org.apache.kafka.test.MockProcessor; import
org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; -import org.apache.kafka.test.TestUtils; -
import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import
java.util.Arrays; @@ -50,30 +46,9 @@ final private String topic1 = "topic1"; final private String topic2 = "topic2"; - final private
Serde<Integer> intSerde = Serdes.Integer(); - final private Serde<String> stringSerde = Serdes.String(); - private final Consumed<Integer,
String> consumed = Consumed.with(intSerde, stringSerde); + private final Consumed<Integer, String> consumed =
Consumed.with(Serdes.Integer(), Serdes.String()); private final ConsumerRecordFactory<Integer, String> recordFactory = new
ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); - private TopologyTestDriver driver; - private Properties
props = new Properties(); - - @Before - public void setUp() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG,
"kstream-kstream-left-join-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); -
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After -
public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props =
StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); @Test public void testLeftJoin() { @@ -91,7 +66,7 @@ public
void testLeftJoin() { joined = stream1.leftJoin(stream2, MockValueJoiner.TOSTRING_JOINER, JoinWindows.of(100), -
Joined.with(intSerde, stringSerde, stringSerde)); + Joined.with(Serdes.Integer(), Serdes.String(), Serdes.String()));
joined.process(supplier); final Collection<Set<String>> copartitionGroups = StreamsBuilderTest.getCopartitionedGroups(builder); @@
-99,65 +74,66 @@ public void testLeftJoin() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>
(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver = new TopologyTestDriver(builder.build(), props, 0L); - - final
MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); - - // push two items to the primary stream. the other
window is empty - // w1 {} - // w2 {} - // --> w1 = { 0:X0, 1:X1 } - // --> w2 = {} - - for (int i = 0; i < 2; i++) { -
driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); - } -
processor.checkAndClearProcessResult("0:X0+null", "1:X1+null"); - - // push two items to the other stream. this should produce two
items. - // w1 = { 0:X0, 1:X1 } - // w2 {} - // --> w1 = { 0:X0, 1:X1 } - // --> w2 = { 0:Y0, 1:Y1 } - - for (int i = 0; i < 2; i++) { -
driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i])); - } - -
processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); - - // push three items to the primary stream. this should produce four
```

items. - // w1 = { 0:X0, 1:X1 } - // w2 = { 0:Y0, 1:Y1 } - // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2 } - // --> w2 = { 0:Y0, 1:Y1 } - - for (int i = 0; i < 3; i++) { - driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + + final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); + + // push two items to the primary stream. the other window is empty + // w1 {} + // w2 {} + // --> w1 = { 0:X0, 1:X1 } + // --> w2 = {} + + for (int i = 0; i < 2; i++) { + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); + } + processor.checkAndClearProcessResult("0:X0+null", "1:X1+null"); + + // push two items to the other stream. this should produce two items. + // w1 = { 0:X0, 1:X1 } + // w2 {} + // --> w1 = { 0:X0, 1:X1 } + // --> w2 = { 0:Y0, 1:Y1 } + + for (int i = 0; i < 2; i++) { + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i])); + } + + processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); + + // push three items to the primary stream. this should produce four items. + // w1 = { 0:X0, 1:X1 } + // w2 = { 0:Y0, 1:Y1 } + // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2 } + // --> w2 = { 0:Y0, 1:Y1 } + + for (int i = 0; i < 3; i++) { + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i])); + } + processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1", "2:X2+null"); + + // push all items to the other stream. this should produce 5 items + // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2 } + // w2 = { 0:Y0, 1:Y1 } + // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2 } + // --> w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3} + + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey)); + } + processor.checkAndClearProcessResult("0:X0+YY0", "0:X0+YY0", "1:X1+YY1", "1:X1+YY1", "2:X2+YY2"); + + // push all four items to the primary stream. this should produce six items. + // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2 } + // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3} + // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } + // --> w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3} + + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey)); + } + processor.checkAndClearProcessResult("0:XX0+Y0", "0:XX0+YY0", "1:XX1+Y1", "1:XX1+YY1", "2:XX2+YY2", "3:XX3+YY3"); } - processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1", "2:X2+null"); - - // push all items to the other stream. this should produce 5 items - // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2 } - // w2 = { 0:Y0, 1:Y1 } - // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2 } - // --> w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3} - - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic2, expectedKey, "YY" + expectedKey)); - } - processor.checkAndClearProcessResult("0:X0+YY0", "0:X0+YY0", "1:X1+YY1", "1:X1+YY1", "2:X2+YY2"); - - // push all four items to the primary stream. this should produce six items. - // w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2 } - // w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3} - // --> w1 = { 0:X0, 1:X1, 0:X0, 1:X1, 2:X2, 0:XX0, 1:XX1, 2:XX2, 3:XX3 } - // --> w2 = { 0:Y0, 1:Y1, 0:YY0, 1:YY1, 2:YY2, 3:YY3} - - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey)); - } - processor.checkAndClearProcessResult("0:XX0+Y0", "0:XX0+YY0", "1:XX1+Y1", "1:XX1+YY1", "2:XX2+YY2", "3:XX3+YY3"); } @Test @@ -176,7 +152,7 @@ public void testWindowing() { joined = stream1.leftJoin(stream2, MockValueJoiner.TOSTRING_JOINER, JoinWindows.of(100), - Joined.with(intSerde, stringSerde, stringSerde)); + Joined.with(Serdes.Integer(), Serdes.String(), Serdes.String())); joined.process(supplier); final Collection<Set<String>> copartitionGroups = StreamsBuilderTest.getCopartitionedGroups(builder); @@ -184,111 +160,112 @@ public void testWindowing() { assertEquals(1, copartitionGroups.size()); assertEquals(new HashSet<>(Arrays.asList(topic1, topic2)), copartitionGroups.iterator().next()); - driver = new TopologyTestDriver(builder.build(), props, time); - - final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); - - // push two items to the primary stream. the other window is empty. this should produce two items - // w1 = {} - // w2 = {} - // --> w1 = { 0:X0, 1:X1 } - // --> w2 = {} - - for (int i = 0; i < 2; i++) { - driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i], time)); - } - processor.checkAndClearProcessResult("0:X0+null", "1:X1+null"); - - // push two items to the other stream. this should produce no items. - // w1 = { 0:X0, 1:X1 } - // w2 = {} - // --> w1 = { 0:X0, 1:X1 } - // --> w2 = { 0:Y0, 1:Y1 } - - for (int i = 0; i < 2; i++) { - driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i], time)); - } - processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); - - // clear logically - time = 1000L; - - // push all items to the other stream. this should produce no items. - // w1 = {} - // w2 = {} - // --> w1 = {} - // --> w2 = { 0:Y0, 1:Y1, 2:Y2, 3:Y3 } - for (int i = 0; i < expectedKeys.length; i++) { - driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i], time + i)); - } - processor.checkAndClearProcessResult(); - - // gradually expire items in window 2. - // w1 = {} - // w2 = {} - // --> w1 = {} - // --> w2 = { 0:Y0, 1:Y1, 2:Y2, 3:Y3 } - - time = 1000L + 100L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } - processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); - - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } - processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); - - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } - processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+Y2", "3:XX3+Y3"); - - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } - processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+null", "3:XX3+Y3"); - - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } - processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+null", "3:XX3+null"); - - // go back to the time before expiration - - time = 1000L - 100L - 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } - processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+null", "3:XX3+null"); - - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } - processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+null", "2:XX2+null", "3:XX3+null"); - - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } - processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+null", "3:XX3+null"); - - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); - } - processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+null"); - - time += 1L; - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, time)) { + + final MockProcessor<Integer, String> processor = supplier.theCapturedProcessor(); + + // push two items to the primary stream. the other window is empty. this should produce two items + // w1 = {} + // w2 = {} + // --> w1 = { 0:X0, 1:X1 } + // --> w2 = {} + + for (int i = 0; i < 2; i++) { + driver.pipeInput(recordFactory.create(topic1, expectedKeys[i], "X" + expectedKeys[i], time)); + } + processor.checkAndClearProcessResult("0:X0+null", "1:X1+null"); + + // push two items to the other stream. this should produce no items. + // w1 = { 0:X0, 1:X1 } + // w2 = {} + // --> w1 = { 0:X0, 1:X1 } + // --> w2 = { 0:Y0, 1:Y1 } + + for (int i = 0; i < 2; i++) { + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i], time)); + } + processor.checkAndClearProcessResult("0:X0+Y0", "1:X1+Y1"); + + // clear logically + time = 1000L; + + // push all items to the other stream. this should produce no items. + // w1 = {} + // w2 = {} + // --> w1 = {} + // --> w2 = { 0:Y0, 1:Y1, 2:Y2, 3:Y3 } + for (int i = 0; i < expectedKeys.length; i++) { + driver.pipeInput(recordFactory.create(topic2, expectedKeys[i], "Y" + expectedKeys[i], time + i)); + } + processor.checkAndClearProcessResult(); + + // gradually expire items in window 2. + // w1 = {} + // w2 = {} + // --> w1 = {} + // --> w2 = { 0:Y0, 1:Y1, 2:Y2, 3:Y3 } + + time = 1000L + 100L; + for (int expectedKey : expectedKeys) { +

```
driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } +
processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); + + time += 1L; + for (int expectedKey
: expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } +
processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); + + time += 1L; + for (int
expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } +
processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+Y2", "3:XX3+Y3"); + + time += 1L; + for (int
expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } +
processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+null", "3:XX3+Y3"); + + time += 1L; + for (int
expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } +
processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+null", "3:XX3+null"); + + // go back to the time before
expiration + + time = 1000L - 100L - 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1,
expectedKey, "XX" + expectedKey, time)); + } + processor.checkAndClearProcessResult("0:XX0+null", "1:XX1+null", "2:XX2+null",
"3:XX3+null"); + + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey,
"XX" + expectedKey, time)); + } + processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+null", "2:XX2+null", "3:XX3+null");
+ + time += 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" +
expectedKey, time)); + } + processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+null", "3:XX3+null"); + + time
+= 1L; + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey,
time)); + } + processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+null"); + + time += 1L; + for
(int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topic1, expectedKey, "XX" + expectedKey, time)); + } +
processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); } -
processor.checkAndClearProcessResult("0:XX0+Y0", "1:XX1+Y1", "2:XX2+Y2", "3:XX3+Y3"); } } diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableJoinTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableJoinTest.java index ec31b5a1d60..0ce27ab51cb 100644
--- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableJoinTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableJoinTest.java @@ -17,22 +17,21 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; -import
org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KTable; -import
org.apache.kafka.test.MockProcessor; import org.apache.kafka.streams.processor.internals.testutil.LogCaptureAppender; import
org.apache.kafka.streams.test.ConsumerRecordFactory; +import org.apache.kafka.test.MockProcessor; import
org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockValueJoiner; -import org.apache.kafka.test.TestUtils;
+import org.apache.kafka.test.StreamsTestUtils; +import org.junit.After; import org.junit.Before; import org.junit.Test; @@ -52,8 +51,6
@@ private final String streamTopic = "streamTopic"; private final String tableTopic = "tableTopic"; - private final Serde<Integer>
intSerde = Serdes.Integer(); - private final Serde<String> stringSerde = Serdes.String(); private final ConsumerRecordFactory<Integer,
String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); private final int[] expectedKeys
= {0, 1, 2, 3}; @@ -70,23 +67,22 @@ public void setUp() { final KTable<Integer, String> table; final MockProcessorSupplier<Integer,
String> supplier = new MockProcessorSupplier<>(); - final Consumed<Integer, String> consumed = Consumed.with(intSerde,
stringSerde); + final Consumed<Integer, String> consumed = Consumed.with(Serdes.Integer(), Serdes.String()); stream =
builder.stream(streamTopic, consumed); table = builder.table(tableTopic, consumed); stream.join(table,
MockValueJoiner.TOSTRING_JOINER).process(supplier); - final Properties props = new Properties(); -
props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-ktable-join-test"); -
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); -
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - + final
Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); driver = new
TopologyTestDriver(builder.build(), props, 0L); processor = supplier.theCapturedProcessor(); } + @After + public void cleanup() { +
driver.close(); + } + private void pushToStream(final int messageCount, final String valuePrefix) { for (int i = 0; i < messageCount; i++) {
driver.pipeInput(recordFactory.create(streamTopic, expectedKeys[i], valuePrefix + expectedKeys[i])); diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableLeftJoinTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableLeftJoinTest.java index 735f71c2ef0..eedda074a41
100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableLeftJoinTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamKTableLeftJoinTest.java @@ -17,13 +17,11 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.StreamsBuilderTest; -import
org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KTable; @@ -31,7 +29,8 @@ import
org.apache.kafka.test.MockProcessor; import org.apache.kafka.test.MockProcessorSupplier; import
org.apache.kafka.test.MockValueJoiner; -import org.apache.kafka.test.TestUtils; +import org.apache.kafka.test.StreamsTestUtils; +import
org.junit.After; import org.junit.Before; import org.junit.Test; @@ -48,8 +47,6 @@ final private String streamTopic = "streamTopic";
final private String tableTopic = "tableTopic"; - final private Serde<Integer> intSerde = Serdes.Integer(); - final private Serde<String>
stringSerde = Serdes.String(); private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>
(new IntegerSerializer(), new StringSerializer()); private TopologyTestDriver driver; private MockProcessor<Integer, String> processor;
@@ -66,23 +63,22 @@ public void setUp() { final KTable<Integer, String> table; final MockProcessorSupplier<Integer, String> supplier
= new MockProcessorSupplier<>(); - final Consumed<Integer, String> consumed = Consumed.with(intSerde, stringSerde); + final
Consumed<Integer, String> consumed = Consumed.with(Serdes.Integer(), Serdes.String()); stream = builder.stream(streamTopic,
consumed); table = builder.table(tableTopic, consumed); stream.leftJoin(table, MockValueJoiner.TOSTRING_JOINER).process(supplier);
- final Properties props = new Properties(); - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-ktable-left-join-
test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); -
```

props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - + final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); driver = new TopologyTestDriver(builder.build(), props, 0L); processor = supplier.theCapturedProcessor(); } + @After + public void cleanup() { + driver.close(); + } + private void pushToStream(final int messageCount, final String valuePrefix) { for (int i = 0; i < messageCount; i++) { driver.pipeInput(recordFactory.create(streamTopic, expectedKeys[i], valuePrefix + expectedKeys[i])); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapTest.java index b0a383ba23a..b55d8e1b8e4 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapTest.java @@ -17,21 +17,17 @@ package org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KeyValueMapper; import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import java.util.Properties; @@ -41,30 +37,8 @@ public class KStreamMapTest { private String topicName = "topic"; - - final private Serde<Integer> intSerde = Serdes.Integer(); - final private Serde<String> stringSerde = Serdes.String(); private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - - @Before - public void setup() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-map-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After - public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); @Test public void testMap() { @@ -80,15 +54,14 @@ public void testMap() { final int[] expectedKeys = new int[]{0, 1, 2, 3}; - KStream<Integer, String> stream = builder.stream(topicName, Consumed.with(intSerde, stringSerde)); - MockProcessorSupplier<String, Integer> supplier; - - supplier = new MockProcessorSupplier<>(); + MockProcessorSupplier<String, Integer> supplier = new MockProcessorSupplier<>(); + KStream<Integer, String> stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.String())); stream.map(mapper).process(supplier); - driver = new TopologyTestDriver(builder.build(), props); - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topicName, expectedKey, "V" + expectedKey)); + } } assertEquals(4, supplier.theCapturedProcessor().processed.size()); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapValuesTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapValuesTest.java index ed110383001..95593aad467 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapValuesTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamMapValuesTest.java @@ -17,21 +17,17 @@ package org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.ValueMapper; import org.apache.kafka.streams.kstream.ValueMapperWithKey; import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import java.util.Properties; @@ -41,33 +37,9 @@ public class KStreamMapValuesTest { private String topicName = "topic"; - - final private Serde<Integer> intSerde = Serdes.Integer(); - final private Serde<String> stringSerde = Serdes.String(); - final private MockProcessorSupplier<Integer, Integer> supplier = new MockProcessorSupplier<>(); - - + private final MockProcessorSupplier<Integer, Integer> supplier = new MockProcessorSupplier<>(); private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - - @Before - public void setup() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-map-values-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After - public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); @Test public void testFlatMapValues() { @@ -83,13 +55,13 @@ public Integer apply(CharSequence value) { final int[] expectedKeys = {1, 10, 100, 1000}; - KStream<Integer, String> stream; - stream = builder.stream(topicName, Consumed.with(intSerde, stringSerde)); + KStream<Integer, String> stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.String())); stream.mapValues(mapper).process(supplier); - driver = new TopologyTestDriver(builder.build(), props); - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topicName, expectedKey, Integer.toString(expectedKey))); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topicName, expectedKey, Integer.toString(expectedKey))); + } } String[] expected = {"1:1", "10:2", "100:3", "1000:4"}; @@ -110,13 +82,13 @@ public Integer apply(final Integer readOnlyKey, final CharSequence value) { final int[] expectedKeys = {1, 10, 100, 1000}; - KStream<Integer, String> stream; - stream = builder.stream(topicName, Consumed.with(intSerde, stringSerde)); + KStream<Integer, String> stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.String())); stream.mapValues(mapper).process(supplier); - driver = new TopologyTestDriver(builder.build(), props); - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topicName, expectedKey, Integer.toString(expectedKey))); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topicName, expectedKey, Integer.toString(expectedKey))); + } } String[] expected = {"1:2", "10:12", "100:103", "1000:1004"}; diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamPeekTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamPeekTest.java index 2c6ff81b8f6..137aa6f473f 100644 ---

```
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamPeekTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamPeekTest.java @@ -17,20 +17,16 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig;
import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.ForeachAction; import
org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.test.ConsumerRecordFactory; -import
org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import
org.junit.Test; import java.util.ArrayList; @@ -43,53 +39,33 @@ public class KStreamPeekTest { private final String topicName =
"topic"; - private final Serde<Integer> intSerd = Serdes.Integer(); - private final Serde<String> stringSerd = Serdes.String(); private final
ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new
StringSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - - @Before - public void
setup() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-peek-test"); -
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); -
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After -
public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props =
StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); @Test public void shouldObserveStreamElements() { final
StreamsBuilder builder = new StreamsBuilder(); - final KStream<Integer, String> stream = builder.stream(topicName,
Consumed.with(intSerd, stringSerd)); + final KStream<Integer, String> stream = builder.stream(topicName,
Consumed.with(Serdes.Integer(), Serdes.String())); final List<KeyValue<Integer, String>> peekObserved = new ArrayList<>(),
streamObserved = new ArrayList<>(); stream.peek(collect(peekObserved)).foreach(collect(streamObserved)); - driver = new
TopologyTestDriver(builder.build(), props); - final List<KeyValue<Integer, String>> expected = new ArrayList<>(); - for (int key = 0; key
< 32; key++) { - final String value = "V" + key; - driver.pipeInput(recordFactory.create(topicName, key, value)); - expected.add(new
KeyValue<>(key, value)); - } + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + final
List<KeyValue<Integer, String>> expected = new ArrayList<>(); + for (int key = 0; key < 32; key++) { + final String value = "V" + key;
+ driver.pipeInput(recordFactory.create(topicName, key, value)); + expected.add(new KeyValue<>(key, value)); + } -
assertEquals(expected, peekObserved); - assertEquals(expected, streamObserved); + assertEquals(expected, peekObserved); +
assertEquals(expected, streamObserved); + } } @Test public void shouldNotAllowNullAction() { final StreamsBuilder builder = new
StreamsBuilder(); - final KStream<Integer, String> stream = builder.stream(topicName, Consumed.with(intSerd, stringSerd)); + final
KStream<Integer, String> stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.String())); try { stream.peek(null);
fail("expected null action to throw NPE"); diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamSelectKeyTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamSelectKeyTest.java index 1abc0b99be6..b030233fa82 100644
--- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamSelectKeyTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamSelectKeyTest.java @@ -17,21 +17,17 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.Consumed; import
org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig; import
org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.ForeachAction; import
org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KeyValueMapper; import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import
org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import
org.junit.Test; import java.util.HashMap; @@ -44,29 +40,8 @@ private String topicName = "topic_key_select"; - final private
Serde<Integer> integerSerde = Serdes.Integer(); - final private Serde<String> stringSerde = Serdes.String(); private final
ConsumerRecordFactory<String, Integer> recordFactory = new ConsumerRecordFactory<>(topicName, new StringSerializer(), new
IntegerSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - - @Before - public void
setup() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-select-key-test"); -
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); -
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - } - - @After
- public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props =
StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.Integer()); @Test public void testSelectKey() { @@ -88,16 +63,16 @@
public String apply(Object key, Number value) { final String[] expected = new String[]{"ONE:1", "TWO:2", "THREE:3"}; final int[]
expectedValues = new int[]{1, 2, 3}; - KStream<String, Integer> stream = builder.stream(topicName, Consumed.with(stringSerde,
integerSerde)); + KStream<String, Integer> stream = builder.stream(topicName, Consumed.with(Serdes.String(), Serdes.Integer()));
MockProcessorSupplier<String, Integer> supplier = new MockProcessorSupplier<>(); stream.selectKey(selector).process(supplier); -
driver = new TopologyTestDriver(builder.build(), props); - - for (int expectedValue : expectedValues) { -
driver.pipeInput(recordFactory.create(expectedValue)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(),
props)) { + for (int expectedValue : expectedValues) { + driver.pipeInput(recordFactory.create(expectedValue)); + } } assertEquals(3,
supplier.theCapturedProcessor().processed.size()); diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformTest.java index 1567fe1657c..8a05aac7fa9 100644 -
-- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformTest.java @@ -17,12 +17,10 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; -
import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.Transformer; @@ -33,9 +31,7 @@ import
org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.KStreamTestDriver; import
org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before;
```

+import org.apache.kafka.test.StreamsTestUtils; import org.junit.Rule; import org.junit.Test; @@ -47,33 +43,12 @@ private String topicName = "topic"; - final private Serde<Integer> intSerde = Serdes.Integer(); - private final ConsumerRecordFactory<Integer, Integer> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new IntegerSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.Integer()); @Rule public final KStreamTestDriver kstreamDriver = new KStreamTestDriver(); - @Before - public void setup() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-transform-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - } - - @After - public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } - @Test public void testTransform() { StreamsBuilder builder = new StreamsBuilder(); @@ -102,7 +77,7 @@ public void close() {} final int[] expectedKeys = {1, 10, 100, 1000}; MockProcessorSupplier<Integer, Integer> processor = new MockProcessorSupplier<>(); - KStream<Integer, Integer> stream = builder.stream(topicName, Consumed.with(intSerde, intSerde)); + KStream<Integer, Integer> stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.Integer())); stream.transform(transformerSupplier).process(processor); kstreamDriver.setUp(builder); @@ -161,18 +136,19 @@ public void close() {} final int[] expectedKeys = {1, 10, 100, 1000}; MockProcessorSupplier<Integer, Integer> processor = new MockProcessorSupplier<>(); - KStream<Integer, Integer> stream = builder.stream(topicName, Consumed.with(intSerde, intSerde)); + KStream<Integer, Integer> stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.Integer())); stream.transform(transformerSupplier).process(processor); - driver = new TopologyTestDriver(builder.build(), props, 0L); - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey * 10, 0L)); - } + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey * 10, 0L)); + } - // This tick will yield yields the "-1:2" result - driver.advanceWallClockTime(2); - // This tick further advances the clock to 3, which leads to the "-1:3" result - driver.advanceWallClockTime(1); + // This tick will yield yields the "-1:2" result + driver.advanceWallClockTime(2); + // This tick further advances the clock to 3, which leads to the "-1:3" result + driver.advanceWallClockTime(1); + } assertEquals(6, processor.theCapturedProcessor().processed.size()); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformValuesTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformValuesTest.java index 6bfc813077e..419e6f1333f 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformValuesTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamTransformValuesTest.java @@ -17,11 +17,9 @@ package org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.errors.StreamsException; import org.apache.kafka.streams.kstream.KStream; @@ -34,9 +32,7 @@ import org.apache.kafka.streams.processor.To; import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import java.util.Properties; @@ -47,31 +43,9 @@ public class KStreamTransformValuesTest { private String topicName = "topic"; - - final private Serde<Integer> intSerde = Serdes.Integer(); - final private MockProcessorSupplier<Integer, Integer> supplier = new MockProcessorSupplier<>(); - + private final MockProcessorSupplier<Integer, Integer> supplier = new MockProcessorSupplier<>(); private final ConsumerRecordFactory<Integer, Integer> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new IntegerSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - - @Before - public void setup() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-transform-values-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); - props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); - props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - } - - @After - public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.Integer()); @Test public void testTransform() { @@ -109,13 +83,13 @@ public void close() { final int[] expectedKeys = {1, 10, 100, 1000}; KStream<Integer, Integer> stream; - stream = builder.stream(topicName, Consumed.with(intSerde, intSerde)); + stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.Integer())); stream.transformValues(valueTransformerSupplier).process(supplier); - driver = new TopologyTestDriver(builder.build(), props); - - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey * 10, 0L)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey * 10, 0L)); + } } String[] expected = {"1:10", "10:110", "100:1110", "1000:11110"}; @@ -152,13 +126,13 @@ public void close() { final int[] expectedKeys = {1, 10, 100, 1000}; KStream<Integer, Integer> stream; - stream = builder.stream(topicName, Consumed.with(intSerde, intSerde)); + stream = builder.stream(topicName, Consumed.with(Serdes.Integer(), Serdes.Integer())); stream.transformValues(valueTransformerSupplier).process(supplier); - driver = new TopologyTestDriver(builder.build(), props); - - for (int expectedKey : expectedKeys) { - driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey * 10, 0L)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + for (int expectedKey : expectedKeys) { + driver.pipeInput(recordFactory.create(topicName, expectedKey, expectedKey * 10, 0L)); + } } String[] expected = {"1:11", "10:121", "100:1221", "1000:12221"}; diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowAggregateTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowAggregateTest.java index 9050edb1941..7a2a8e0958b 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowAggregateTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowAggregateTest.java @@ -16,14 +16,12 @@ */ package org.apache.kafka.streams.kstream.internals; -import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; import org.apache.kafka.common.utils.Utils; import org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.StreamsBuilder; -import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KTable; @@ -39,9 +37,7 @@ import org.apache.kafka.test.MockInitializer; import org.apache.kafka.test.MockProcessor; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; -import org.junit.After; -import org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import

```
java.util.List; @@ -54,28 +50,8 @@ public class KStreamWindowAggregateTest { - final private Serde<String> strSerde =
Serdes.String(); private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new
StringSerializer(), new StringSerializer()); - private TopologyTestDriver driver; - private final Properties props = new Properties(); - -
@Before - public void setup() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "kstream-window-aggregate-test"); -
props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); -
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); - } - - @After
- public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props =
StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); @Test public void testAggBasic() { @@ -83,31 +59,31 @@
public void testAggBasic() { final String topic1 = "topic1"; final KTable<Windowed<String>, String> table2 = builder - .stream(topic1,
Consumed.with(strSerde, strSerde)) - .groupByKey(Serialized.with(strSerde, strSerde)) - .aggregate(MockInitializer.STRING_INIT,
MockAggregator.TOSTRING_ADDER, TimeWindows.of(10).advanceBy(5), strSerde, "topic1-Canonized"); + .stream(topic1,
Consumed.with(Serdes.String(), Serdes.String())) + .groupByKey(Serialized.with(Serdes.String(), Serdes.String())) +
.aggregate(MockInitializer.STRING_INIT, MockAggregator.TOSTRING_ADDER, TimeWindows.of(10).advanceBy(5), Serdes.String(),
"topic1-Canonized"); final MockProcessorSupplier<Windowed<String>, String> supplier = new MockProcessorSupplier<>();
table2.toStream().process(supplier); - driver = new TopologyTestDriver(builder.build(), props, 0L); - -
driver.pipeInput(recordFactory.create(topic1, "A", "1", 0L)); - driver.pipeInput(recordFactory.create(topic1, "B", "2", 1L)); -
driver.pipeInput(recordFactory.create(topic1, "C", "3", 2L)); - driver.pipeInput(recordFactory.create(topic1, "D", "4", 3L)); -
driver.pipeInput(recordFactory.create(topic1, "A", "1", 4L)); - - driver.pipeInput(recordFactory.create(topic1, "A", "1", 5L)); -
driver.pipeInput(recordFactory.create(topic1, "B", "2", 6L)); - driver.pipeInput(recordFactory.create(topic1, "D", "4", 7L)); -
driver.pipeInput(recordFactory.create(topic1, "B", "2", 8L)); - driver.pipeInput(recordFactory.create(topic1, "C", "3", 9L)); -
driver.pipeInput(recordFactory.create(topic1, "A", "1", 10L)); - driver.pipeInput(recordFactory.create(topic1, "B", "2", 11L)); -
driver.pipeInput(recordFactory.create(topic1, "D", "4", 12L)); - driver.pipeInput(recordFactory.create(topic1, "B", "2", 13L)); -
driver.pipeInput(recordFactory.create(topic1, "C", "3", 14L)); + try (final TopologyTestDriver driver = new
TopologyTestDriver(builder.build(), props, 0L)) { + driver.pipeInput(recordFactory.create(topic1, "A", "1", 0L)); +
driver.pipeInput(recordFactory.create(topic1, "B", "2", 1L)); + driver.pipeInput(recordFactory.create(topic1, "C", "3", 2L)); +
driver.pipeInput(recordFactory.create(topic1, "D", "4", 3L)); + driver.pipeInput(recordFactory.create(topic1, "A", "1", 4L)); + +
driver.pipeInput(recordFactory.create(topic1, "A", "1", 5L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2", 6L)); +
driver.pipeInput(recordFactory.create(topic1, "D", "4", 7L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2", 8L)); +
driver.pipeInput(recordFactory.create(topic1, "C", "3", 9L)); + driver.pipeInput(recordFactory.create(topic1, "A", "1", 10L)); +
driver.pipeInput(recordFactory.create(topic1, "B", "2", 11L)); + driver.pipeInput(recordFactory.create(topic1, "D", "4", 12L)); +
driver.pipeInput(recordFactory.create(topic1, "B", "2", 13L)); + driver.pipeInput(recordFactory.create(topic1, "C", "3", 14L)); + }
assertEquals( @@ -141,16 +117,16 @@ public void testJoin() { final String topic2 = "topic2"; final KTable<Windowed<String>, String>
table1 = builder - .stream(topic1, Consumed.with(strSerde, strSerde)) - .groupByKey(Serialized.with(strSerde, strSerde)) -
.aggregate(MockInitializer.STRING_INIT, MockAggregator.TOSTRING_ADDER, TimeWindows.of(10).advanceBy(5), strSerde,
"topic1-Canonized"); + .stream(topic1, Consumed.with(Serdes.String(), Serdes.String())) + .groupByKey(Serialized.with(Serdes.String(),
Serdes.String())) + .aggregate(MockInitializer.STRING_INIT, MockAggregator.TOSTRING_ADDER,
TimeWindows.of(10).advanceBy(5), Serdes.String(), "topic1-Canonized"); final MockProcessorSupplier<Windowed<String>, String>
supplier = new MockProcessorSupplier<>(); table1.toStream().process(supplier); final KTable<Windowed<String>, String> table2 =
builder - .stream(topic2, Consumed.with(strSerde, strSerde)).groupByKey(Serialized.with(strSerde, strSerde)) -
.aggregate(MockInitializer.STRING_INIT, MockAggregator.TOSTRING_ADDER, TimeWindows.of(10).advanceBy(5), strSerde,
"topic2-Canonized"); + .stream(topic2, Consumed.with(Serdes.String(), Serdes.String())).groupByKey(Serialized.with(Serdes.String(),
Serdes.String())) + .aggregate(MockInitializer.STRING_INIT, MockAggregator.TOSTRING_ADDER,
TimeWindows.of(10).advanceBy(5), Serdes.String(), "topic2-Canonized"); table2.toStream().process(supplier); @@ -162,84 +138,84
@@ public String apply(final String p1, final String p2) { } }).toStream().process(supplier); - driver = new
TopologyTestDriver(builder.build(), props, 0L); - - driver.pipeInput(recordFactory.create(topic1, "A", "1", 0L)); -
driver.pipeInput(recordFactory.create(topic1, "B", "2", 1L)); - driver.pipeInput(recordFactory.create(topic1, "C", "3", 2L)); -
driver.pipeInput(recordFactory.create(topic1, "D", "4", 3L)); - driver.pipeInput(recordFactory.create(topic1, "A", "1", 4L)); - - final
List<MockProcessor<Windowed<String>, String>> processors = supplier.capturedProcessors(3); - -
processors.get(0).checkAndClearProcessResult( - "[A@0/10]:0+1", - "[B@0/10]:0+2", - "[C@0/10]:0+3", - "[D@0/10]:0+4", - "
[A@0/10]:0+1+1" - ); - processors.get(1).checkAndClearProcessResult(); - processors.get(2).checkAndClearProcessResult(); - -
driver.pipeInput(recordFactory.create(topic1, "A", "1", 5L)); - driver.pipeInput(recordFactory.create(topic1, "B", "2", 6L)); -
driver.pipeInput(recordFactory.create(topic1, "D", "4", 7L)); - driver.pipeInput(recordFactory.create(topic1, "B", "2", 8L)); -
driver.pipeInput(recordFactory.create(topic1, "C", "3", 9L)); - - processors.get(0).checkAndClearProcessResult( - "[A@0/10]:0+1+1+1", "
[A@5/15]:0+1", - "[B@0/10]:0+2+2", "[B@5/15]:0+2", - "[D@0/10]:0+4+4", "[D@5/15]:0+4", - "[B@0/10]:0+2+2+2", "
[B@5/15]:0+2+2", - "[C@0/10]:0+3+3", "[C@5/15]:0+3" - ); - processors.get(1).checkAndClearProcessResult(); -
processors.get(2).checkAndClearProcessResult(); - - driver.pipeInput(recordFactory.create(topic2, "A", "a", 0L)); -
driver.pipeInput(recordFactory.create(topic2, "B", "b", 1L)); - driver.pipeInput(recordFactory.create(topic2, "C", "c", 2L)); -
driver.pipeInput(recordFactory.create(topic2, "D", "d", 3L)); - driver.pipeInput(recordFactory.create(topic2, "A", "a", 4L)); - -
processors.get(0).checkAndClearProcessResult(); - processors.get(1).checkAndClearProcessResult( - "[A@0/10]:0+a", - "[B@0/10]:0+b",
- "[C@0/10]:0+c", - "[D@0/10]:0+d", - "[A@0/10]:0+a+a" - ); - processors.get(2).checkAndClearProcessResult( - "
[A@0/10]:0+1+1+1%0+a", - "[B@0/10]:0+2+2+2%0+b", - "[C@0/10]:0+3+3%0+c", - "[D@0/10]:0+4+4%0+d", - "
[A@0/10]:0+1+1+1%0+a+a"); - - driver.pipeInput(recordFactory.create(topic2, "A", "a", 5L)); -
driver.pipeInput(recordFactory.create(topic2, "B", "b", 6L)); - driver.pipeInput(recordFactory.create(topic2, "D", "d", 7L)); -
driver.pipeInput(recordFactory.create(topic2, "B", "b", 8L)); - driver.pipeInput(recordFactory.create(topic2, "C", "c", 9L)); - -
processors.get(0).checkAndClearProcessResult(); - processors.get(1).checkAndClearProcessResult( - "[A@0/10]:0+a+a+a", "
[A@5/15]:0+a", - "[B@0/10]:0+b+b", "[B@5/15]:0+b", - "[D@0/10]:0+d+d", "[D@5/15]:0+d", - "[B@0/10]:0+b+b+b", "
[B@5/15]:0+b+b", - "[C@0/10]:0+c+c", "[C@5/15]:0+c" - ); - processors.get(2).checkAndClearProcessResult( - "
[A@0/10]:0+1+1+1%0+a+a+a", "[A@5/15]:0+1%0+a", - "[B@0/10]:0+2+2+2%0+b+b", "[B@5/15]:0+2+2%0+b", - "
[D@0/10]:0+4+4%0+d+d", "[D@5/15]:0+4%0+d", - "[B@0/10]:0+2+2+2%0+b+b+b", "[B@5/15]:0+2+2%0+b+b", - "
[C@0/10]:0+3+3%0+c+c", "[C@5/15]:0+3%0+c" - ); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(),
props, 0L)) { + driver.pipeInput(recordFactory.create(topic1, "A", "1", 0L)); + driver.pipeInput(recordFactory.create(topic1, "B", "2",
1L)); + driver.pipeInput(recordFactory.create(topic1, "C", "3", 2L)); + driver.pipeInput(recordFactory.create(topic1, "D", "4", 3L)); +
driver.pipeInput(recordFactory.create(topic1, "A", "1", 4L)); + + final List<MockProcessor<Windowed<String>, String>> processors =
supplier.capturedProcessors(3); + + processors.get(0).checkAndClearProcessResult( + "[A@0/10]:0+1", + "[B@0/10]:0+2", + "
[C@0/10]:0+3", + "[D@0/10]:0+4", + "[A@0/10]:0+1+1" + ); + processors.get(1).checkAndClearProcessResult(); +
processors.get(2).checkAndClearProcessResult(); + + driver.pipeInput(recordFactory.create(topic1, "A", "1", 5L)); +
driver.pipeInput(recordFactory.create(topic1, "B", "2", 6L)); + driver.pipeInput(recordFactory.create(topic1, "D", "4", 7L)); +
driver.pipeInput(recordFactory.create(topic1, "B", "2", 8L)); + driver.pipeInput(recordFactory.create(topic1, "C", "3", 9L)); + +
processors.get(0).checkAndClearProcessResult( + "[A@0/10]:0+1+1+1", "[A@5/15]:0+1", + "[B@0/10]:0+2+2", "[B@5/15]:0+2", + "
[D@0/10]:0+4+4", "[D@5/15]:0+4", + "[B@0/10]:0+2+2+2", "[B@5/15]:0+2+2", + "[C@0/10]:0+3+3", "[C@5/15]:0+3" + ); +
processors.get(1).checkAndClearProcessResult(); + processors.get(2).checkAndClearProcessResult(); + +
driver.pipeInput(recordFactory.create(topic2, "A", "a", 0L)); + driver.pipeInput(recordFactory.create(topic2, "B", "b", 1L)); +
driver.pipeInput(recordFactory.create(topic2, "C", "c", 2L)); + driver.pipeInput(recordFactory.create(topic2, "D", "d", 3L)); +
driver.pipeInput(recordFactory.create(topic2, "A", "a", 4L)); + + processors.get(0).checkAndClearProcessResult(); +
processors.get(1).checkAndClearProcessResult( + "[A@0/10]:0+a", + "[B@0/10]:0+b", + "[C@0/10]:0+c", + "[D@0/10]:0+d", + "
[A@0/10]:0+a+a" + ); + processors.get(2).checkAndClearProcessResult( + "[A@0/10]:0+1+1+1%0+a", + "[B@0/10]:0+2+2+2%0+b", +
"[C@0/10]:0+3+3%0+c", + "[D@0/10]:0+4+4%0+d", + "[A@0/10]:0+1+1+1%0+a+a"); + +
driver.pipeInput(recordFactory.create(topic2, "A", "a", 5L)); + driver.pipeInput(recordFactory.create(topic2, "B", "b", 6L)); +
driver.pipeInput(recordFactory.create(topic2, "D", "d", 7L)); + driver.pipeInput(recordFactory.create(topic2, "B", "b", 8L)); +
driver.pipeInput(recordFactory.create(topic2, "C", "c", 9L)); + + processors.get(0).checkAndClearProcessResult(); +
processors.get(1).checkAndClearProcessResult( + "[A@0/10]:0+a+a+a", "[A@5/15]:0+a", + "[B@0/10]:0+b+b", "[B@5/15]:0+b", + "
[D@0/10]:0+d+d", "[D@5/15]:0+d", + "[B@0/10]:0+b+b+b", "[B@5/15]:0+b+b", + "[C@0/10]:0+c+c", "[C@5/15]:0+c" + ); +
processors.get(2).checkAndClearProcessResult( + "[A@0/10]:0+1+1+1%0+a+a+a", "[A@5/15]:0+1%0+a", + "
[B@0/10]:0+2+2+2%0+b+b", "[B@5/15]:0+2+2%0+b", + "[D@0/10]:0+4+4%0+d+d", "[D@5/15]:0+4%0+d", + "
[B@0/10]:0+2+2+2%0+b+b+b", "[B@5/15]:0+2+2%0+b+b", + "[C@0/10]:0+3+3%0+c+c", "[C@5/15]:0+3%0+c" + ); + } } @Test @@
-247,22 +223,22 @@ public void shouldLogAndMeterWhenSkippingNullKey() { final StreamsBuilder builder = new StreamsBuilder();
final String topic = "topic"; - final KStream<String, String> stream1 = builder.stream(topic, Consumed.with(strSerde, strSerde)); -
stream1.groupByKey(Serialized.with(strSerde, strSerde)) + final KStream<String, String> stream1 = builder.stream(topic,
Consumed.with(Serdes.String(), Serdes.String())); + stream1.groupByKey(Serialized.with(Serdes.String(), Serdes.String()))
.windowedBy(TimeWindows.of(10).advanceBy(5)) .aggregate( MockInitializer.STRING_INIT, MockAggregator.<String,
String>toStringInstance("+"), - Materialized.<String, String, WindowStore<Bytes, byte[]>>as("topic1-
Canonicalized").withValueSerde(strSerde) + Materialized.<String, String, WindowStore<Bytes, byte[]>>as("topic1-
Canonicalized").withValueSerde(Serdes.String()) ); - driver = new TopologyTestDriver(builder.build(), props, 0L); - final
LogCaptureAppender appender = LogCaptureAppender.createAndRegister(); - driver.pipeInput(recordFactory.create(topic, null, "1")); -
LogCaptureAppender.unregister(appender); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L))
{ + driver.pipeInput(recordFactory.create(topic, null, "1")); + LogCaptureAppender.unregister(appender); - assertEquals(1.0,
getMetricByName(driver.metrics(), "skipped-records-total", "stream-metrics").metricValue()); - assertThat(appender.getMessages(),
hasItem("Skipping record due to null key. value=[1] topic=[topic] partition=[0] offset=[0]")); + assertEquals(1.0,
getMetricByName(driver.metrics(), "skipped-records-total", "stream-metrics").metricValue()); + assertThat(appender.getMessages(),
hasItem("Skipping record due to null key. value=[1] topic=[topic] partition=[0] offset=[0]")); + } } } diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableForeachTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableForeachTest.java index 30c0a7a51c3..a6b6c649299 100644 ---
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableForeachTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableForeachTest.java @@ -17,23 +17,19 @@ package
org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.IntegerSerializer; -import
org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; import
org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; import
org.apache.kafka.streams.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; -
import org.apache.kafka.streams.StreamsConfig; import org.apache.kafka.streams.TopologyTestDriver; import
org.apache.kafka.streams.kstream.ForeachAction; import org.apache.kafka.streams.kstream.KTable; import
org.apache.kafka.streams.kstream.Materialized; import org.apache.kafka.streams.state.KeyValueStore; import
org.apache.kafka.streams.test.ConsumerRecordFactory; -import org.apache.kafka.test.TestUtils; -import org.junit.After; -import
org.junit.Before; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; import java.util.ArrayList; @@ -48,29 +44,8 @@
public class KTableForeachTest { final private String topicName = "topic"; - final private Serde<Integer> intSerde = Serdes.Integer(); -
final private Serde<String> stringSerde = Serdes.String(); private final ConsumerRecordFactory<Integer, String> recordFactory = new
ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); - private TopologyTestDriver driver; - private final Properties
props = new Properties(); - - @Before - public void setup() { - props.setProperty(StreamsConfig.APPLICATION_ID_CONFIG, "ktable-
foreach-test"); - props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9091"); -
props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); -
props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.Integer().getClass().getName()); -
props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName()); - } - - @After -
public void cleanup() { - props.clear(); - if (driver != null) { - driver.close(); - } - driver = null; - } + private final Properties props =
StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); @Test public void testForeach() { @@ -101,17 +76,17 @@
public void apply(Integer key, String value) { // When StreamsBuilder builder = new StreamsBuilder(); KTable<Integer, String> table =
builder.table(topicName, - Consumed.with(intSerde, stringSerde), + Consumed.with(Serdes.Integer(), Serdes.String()), Materialized.
<Integer, String, KeyValueStore<Bytes, byte[]>>as(topicName) - .withKeySerde(intSerde) - .withValueSerde(stringSerde)); +
.withKeySerde(Serdes.Integer()) + .withValueSerde(Serdes.String())); table.foreach(action); // Then - driver = new
TopologyTestDriver(builder.build(), props); - - for (KeyValue<Integer, String> record: inputRecords) { -
driver.pipeInput(recordFactory.create(topicName, record.key, record.value)); + try (final TopologyTestDriver driver = new
TopologyTestDriver(builder.build(), props)) { + for (KeyValue<Integer, String> record : inputRecords) { +
driver.pipeInput(recordFactory.create(topicName, record.key, record.value)); + } } assertEquals(expectedRecords.size(),
actualRecords.size()); diff --git a/streams/src/test/java/org/apache/kafka/test/KStreamTestDriver.java
b/streams/src/test/java/org/apache/kafka/test/KStreamTestDriver.java index cf4460dfdd0..bcb98566a6d 100644 ---
a/streams/src/test/java/org/apache/kafka/test/KStreamTestDriver.java +++
b/streams/src/test/java/org/apache/kafka/test/KStreamTestDriver.java @@ -44,6 +44,11 @@ import java.util.Set; import
java.util.regex.Pattern; +/** + * KStreamTestDriver + * + * @deprecated please use {@link
org.apache.kafka.streams.TopologyTestDriver} instead + */ @Deprecated public class KStreamTestDriver extends ExternalResource {
diff --git a/streams/src/test/java/org/apache/kafka/test/StreamsTestUtils.java
b/streams/src/test/java/org/apache/kafka/test/StreamsTestUtils.java index a19b55ce451..940651915f2 100644 ---
a/streams/src/test/java/org/apache/kafka/test/StreamsTestUtils.java +++ b/streams/src/test/java/org/apache/kafka/test/StreamsTestUtils.java
```

@@ -19,6 +19,7 @@ import org.apache.kafka.clients.consumer.ConsumerConfig; import org.apache.kafka.common.Metric; import org.apache.kafka.common.MetricName; +import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.utils.Bytes; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsConfig; @@ -58,6 +59,28 @@ public static Properties getStreamsConfig(final String applicationId, } + public static Properties topologyTestConfig(final String applicationId, + final String bootstrapServers, + final String keyDeserializer, + final String valueDeserializer) { + final Properties props = new Properties(); + props.put(StreamsConfig.APPLICATION_ID_CONFIG, applicationId); + props.setProperty(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServers); + props.setProperty(StreamsConfig.STATE_DIR_CONFIG, TestUtils.tempDirectory().getAbsolutePath()); + props.setProperty(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, keyDeserializer); + props.setProperty(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, valueDeserializer); + return props; + } + + public static Properties topologyTestConfig(final Serde keyDeserializer, + final Serde valueDeserializer) { + return topologyTestConfig(+ UUID.randomUUID().toString(), + "localhost:9091", + keyDeserializer.getClass().getName(), + valueDeserializer.getClass().getName()); + } + public static Properties minimalStreamsConfig() { final Properties properties = new Properties(); properties.put(StreamsConfig.APPLICATION_ID_CONFIG, UUID.randomUUID().toString()); diff --git a/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java b/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java index f6bbc4b5c27..7b8bdd5664d 100644 --- a/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java +++ b/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java @@ -226,7 +226,7 @@ public TopologyTestDriver(final Topology topology, * @param builder builder for the topology to be tested * @param config the configuration for the topology */ - protected TopologyTestDriver(final InternalTopologyBuilder builder, + TopologyTestDriver(final InternalTopologyBuilder builder, final Properties config) { this(builder, config, System.currentTimeMillis()); -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

24. h314to opened a new pull request #4986: KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [part 2] URL: https://github.com/apache/kafka/pull/4986 This PR is a further step towards the complete replacement of `KStreamTestDriver` with `TopologyTestDriver`. Currently there's only a few tests in `KTableAggregateTest` which don't work with `TopologyTestDriver` . Everything else has now been migrated to the new driver. * Refactor: - TimeWindowedKStreamImplTest - SessionWindowedKStreamImplTest - KTableMapKeysTest - KTableFilterTest - KTableKTableInnerJoinTest - KTableSourceTest - KTableMapValuesTest - KTableKTableOuterJoinTest - KTableKTableLeftJoinTest - KStreamWindowReduceTest - KTableImplTest. - KTableAggregateTest (partial) * Rename TopologyTestDriverWrapper to TopologyTestDriverAccessor. * Add task, processorTopology, and globalTopology access to TopologyTestDriverAccessor * Add condition to prevent NPE in ProcessorContextImpl -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
25. h314to opened a new pull request #5052: KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [part 3] URL: https://github.com/apache/kafka/pull/5052 This PR is a further step towards the complete replacement of `KStreamTestDriver` with `TopologyTestDriver`. These straightforward changes were split from another [PR](https://github.com/apache/kafka/pull/4986) to simplify the review process. * Refactor: - KStreamWindowReduceTest - KTableMapKeysTest - SessionWindowedKStreamImplTest - TimeWindowedKStreamImplTest -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
26. guozhangwang closed pull request #5052: KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [part 3] URL: https://github.com/apache/kafka/pull/5052 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowReduceTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowReduceTest.java index aa2397170f6..4ae2f76698b 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowReduceTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KStreamWindowReduceTest.java @@ -17,25 +17,32 @@ package org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.Serdes; -import org.apache.kafka.streams.kstream.Consumed; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.TopologyTestDriver; +import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.kstream.Reducer; import org.apache.kafka.streams.kstream.Serialized; import org.apache.kafka.streams.kstream.TimeWindows; import org.apache.kafka.streams.processor.internals.testutil.LogCaptureAppender; -import org.apache.kafka.test.KStreamTestDriver; -import org.apache.kafka.test.TestUtils; +import org.apache.kafka.streams.test.ConsumerRecordFactory; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; +import java.util.Properties; + import static org.apache.kafka.test.StreamsTestUtils.getMetricByName; import static org.hamcrest.CoreMatchers.hasItem; import static org.hamcrest.MatcherAssert.assertThat; import static org.junit.Assert.assertEquals; public class KStreamWindowReduceTest { + + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); + private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + @Test public void shouldLogAndMeterOnNullKey() { - final KStreamTestDriver driver = new KStreamTestDriver(); final StreamsBuilder builder = new StreamsBuilder(); builder @@ -49,14 +56,14 @@ public String apply(final String value1, final String value2) { } }); - driver.setUp(builder, TestUtils.tempDirectory(), 0); - final LogCaptureAppender appender = LogCaptureAppender.createAndRegister(); - driver.process("TOPIC", null, "asdf"); - driver.flushState(); - LogCaptureAppender.unregister(appender); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + final LogCaptureAppender appender = LogCaptureAppender.createAndRegister(); + driver.pipeInput(recordFactory.create("TOPIC", null, "asdf")); + LogCaptureAppender.unregister(appender); - assertEquals(1.0, getMetricByName(driver.context().metrics().metrics(), "skipped-records-total", "stream-metrics").metricValue()); - assertThat(appender.getMessages(), hasItem("Skipping record due to null key. value=[asdf] topic=[TOPIC] partition=[-1] offset=[-1]")); + assertEquals(1.0, getMetricByName(driver.metrics(), "skipped-records-total", "stream-metrics").metricValue()); + assertThat(appender.getMessages(), hasItem("Skipping record due to null key. value=[asdf] topic=[TOPIC] partition=[0] offset=[0]")); + } } } diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableMapKeysTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableMapKeysTest.java index 14552d6b325..081c6a069aa 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableMapKeysTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableMapKeysTest.java @@ -17,40 +17,30 @@ package

org.apache.kafka.streams.kstream.internals; -import org.apache.kafka.common.serialization.Serde; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serdes; -import org.apache.kafka.streams.kstream.Consumed; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.TopologyTestDriver; +import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.KTable; import org.apache.kafka.streams.kstream.KeyValueMapper; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; - -import org.junit.Before; -import org.junit.Rule; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; -import java.io.File; import java.util.HashMap; import java.util.Map; +import java.util.Properties; import static org.junit.Assert.assertEquals; public class KTableMapKeysTest { - final private Serde<String> stringSerde = new Serdes.StringSerde(); - final private Serde<Integer> integerSerde = new Serdes.IntegerSerde(); - private File stateDir = null; - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); - - - @Before - public void setUp() { - stateDir = TestUtils.tempDirectory("kafka-test"); - } + private final ConsumerRecordFactory<Integer, String> recordFactory = new ConsumerRecordFactory<>(new IntegerSerializer(), new StringSerializer()); + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.Integer(), Serdes.String()); @Test public void testMapKeysConvertingToStream() { @@ -58,7 +48,7 @@ public void testMapKeysConvertingToStream() { String topic1 = "topic_map_keys"; - KTable<Integer, String> table1 = builder.table(topic1, Consumed.with(integerSerde, stringSerde)); + KTable<Integer, String> table1 = builder.table(topic1, Consumed.with(Serdes.Integer(), Serdes.String())); final Map<Integer, String> keyMap = new HashMap<>(); keyMap.put(1, "ONE"); @@ -82,11 +72,11 @@ public String apply(Integer key, String value) { convertedStream.process(supplier); - driver.setUp(builder, stateDir); - for (int i = 0; i < originalKeys.length; i++) { - driver.process(topic1, originalKeys[i], values[i]); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + for (int i = 0; i < originalKeys.length; i++) { + driver.pipeInput(recordFactory.create(topic1, originalKeys[i], values[i])); + } } - driver.flushState(); assertEquals(3, supplier.theCapturedProcessor().processed.size()); diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/SessionWindowedKStreamImplTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/SessionWindowedKStreamImplTest.java index 08fa65c2ad9..825edb3eb37 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/SessionWindowedKStreamImplTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/SessionWindowedKStreamImplTest.java @@ -18,10 +18,12 @@ package org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; -import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.TopologyTestDriver; +import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.kstream.ForeachAction; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.Materialized; @@ -31,20 +33,19 @@ import org.apache.kafka.streams.kstream.SessionWindows; import org.apache.kafka.streams.kstream.Windowed; import org.apache.kafka.streams.state.SessionStore; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockAggregator; import org.apache.kafka.test.MockInitializer; import org.apache.kafka.test.MockReducer; import org.apache.kafka.test.StreamsTestUtils; -import org.apache.kafka.test.TestUtils; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; import java.util.Arrays; import java.util.HashMap; import java.util.List; import java.util.Map; +import java.util.Properties; import static org.hamcrest.CoreMatchers.equalTo; import static org.hamcrest.MatcherAssert.assertThat; @@ -53,9 +54,9 @@ private static final String TOPIC = "input"; private final StreamsBuilder builder = new StreamsBuilder(); + private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); private final Merger<String, String> sessionMerger = new Merger<String, String>() { @Override public String apply(final String aggKey, final String aggOne, final String aggTwo) { @@ -83,7 +84,9 @@ public void apply(final Windowed<String> key, final Long value) { } }); - processData(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + } assertThat(results.get(new Windowed<>("1", new SessionWindow(10, 15))), equalTo(2L)); assertThat(results.get(new Windowed<>("2", new SessionWindow(600, 600))), equalTo(1L)); assertThat(results.get(new Windowed<>("1", new SessionWindow(600, 600))), equalTo(1L)); @@ -101,7 +104,9 @@ public void apply(final Windowed<String> key, final String value) { } }); - processData(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + } assertThat(results.get(new Windowed<>("1", new SessionWindow(10, 15))), equalTo("1+2")); assertThat(results.get(new Windowed<>("2", new SessionWindow(600, 600))), equalTo("1")); assertThat(results.get(new Windowed<>("1", new SessionWindow(600, 600))), equalTo("3")); @@ -121,42 +126,45 @@ public void apply(final Windowed<String> key, final String value) { results.put(key, value); } }); - processData(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + } assertThat(results.get(new Windowed<>("1", new SessionWindow(10, 15))), equalTo("0+0+1+2")); assertThat(results.get(new Windowed<>("2", new SessionWindow(600, 600))), equalTo("0+1")); assertThat(results.get(new Windowed<>("1", new SessionWindow(600, 600))), equalTo("0+3")); } - @SuppressWarnings("unchecked") @Test public void shouldMaterializeCount() { stream.count(Materialized.<String, Long, SessionStore<Bytes, byte[]>>as("count-store")); - processData(); - final SessionStore<String, Long> store = (SessionStore<String, Long>) driver.allStateStores().get("count-store"); - final List<KeyValue<Windowed<String>, Long>> data = StreamsTestUtils.toList(store.fetch("1", "2")); - assertThat(data, equalTo(Arrays.asList(- KeyValue.pair(new Windowed<>("1", new SessionWindow(10, 15)), 2L), - KeyValue.pair(new Windowed<>("1", new SessionWindow(600, 600)), 1L), - KeyValue.pair(new Windowed<>("2", new SessionWindow(600, 600)), 1L)))); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + final SessionStore<String, Long> store = driver.getSessionStore("count-store"); + final List<KeyValue<Windowed<String>, Long>> data = StreamsTestUtils.toList(store.fetch("1", "2")); + assertThat(data, equalTo(Arrays.asList(+ KeyValue.pair(new Windowed<>("1", new SessionWindow(10, 15)), 2L), + KeyValue.pair(new Windowed<>("1", new SessionWindow(600, 600)), 1L), + KeyValue.pair(new Windowed<>("2", new SessionWindow(600, 600)), 1L)))); + } } - @SuppressWarnings("unchecked") @Test public void shouldMaterializeReduced() { stream.reduce(MockReducer.STRING_ADDER, Materialized.<String, String, SessionStore<Bytes, byte[]>>as("reduced")); - processData(); - final SessionStore<String, String> sessionStore = (SessionStore<String, String>) driver.allStateStores().get("reduced"); - final List<KeyValue<Windowed<String>, String>> data = StreamsTestUtils.toList(sessionStore.fetch("1", "2")); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + final SessionStore<String, String> sessionStore = driver.getSessionStore("reduced"); + final List<KeyValue<Windowed<String>, String>> data = StreamsTestUtils.toList(sessionStore.fetch("1", "2")); - assertThat(data, equalTo(Arrays.asList(- KeyValue.pair(new Windowed<>("1", new SessionWindow(10, 15)), "1+2"), - KeyValue.pair(new Windowed<>("1", new SessionWindow(600, 600)), "3"), -

KeyValue.pair(new Windowed<>("2", new SessionWindow(600, 600)), "1")))); + assertThat(data, equalTo(Arrays.asList(+ KeyValue.pair(new Windowed<>("1", new SessionWindow(10, 15)), "1+2"), + KeyValue.pair(new Windowed<>("1", new SessionWindow(600, 600)), "3"), + KeyValue.pair(new Windowed<>("2", new SessionWindow(600, 600)), "1")))); + } } - @SuppressWarnings("unchecked") @Test public void shouldMaterializeAggregated() { stream.aggregate(MockInitializer.STRING_INIT, @@ -164,13 +172,15 @@ public void shouldMaterializeAggregated() { sessionMerger, Materialized.<String, String, SessionStore<Bytes, byte[]>>as("aggregated").withValueSerde(Serdes.String())); - processData(); - final SessionStore<String, String> sessionStore = (SessionStore<String, String>) driver.allStateStores().get("aggregated"); - final List<KeyValue<Windowed<String>, String>> data = StreamsTestUtils.toList(sessionStore.fetch("1", "2")); - assertThat(data, equalTo(Arrays.asList(- KeyValue.pair(new Windowed<>("1", new SessionWindow(10, 15)), "0+0+1+2"), - KeyValue.pair(new Windowed<>("1", new SessionWindow(600, 600)), "0+3"), - KeyValue.pair(new Windowed<>("2", new SessionWindow(600, 600)), "0+1")))); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + final SessionStore<String, String> sessionStore = driver.getSessionStore("aggregated"); + final List<KeyValue<Windowed<String>, String>> data = StreamsTestUtils.toList(sessionStore.fetch("1", "2")); + assertThat(data, equalTo(Arrays.asList(+ KeyValue.pair(new Windowed<>("1", new SessionWindow(10, 15)), "0+0+1+2"), + KeyValue.pair(new Windowed<>("1", new SessionWindow(600, 600)), "0+3"), + KeyValue.pair(new Windowed<>("2", new SessionWindow(600, 600)), "0+1")))); + } } @Test(expected = NullPointerException.class) @@ -243,16 +253,11 @@ public void shouldThrowNullPointerOnCountIfMaterializedIsNull() { stream.count(null); } - private void processData() { - driver.setUp(builder, TestUtils.tempDirectory(), 0); - driver.setTime(10); - driver.process(TOPIC, "1", "1"); - driver.setTime(15); - driver.process(TOPIC, "1", "2"); - driver.setTime(600); - driver.process(TOPIC, "1", "3"); - driver.process(TOPIC, "2", "1"); - driver.flushState(); + private void processData(final TopologyTestDriver driver) { + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 10)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "2", 15)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "3", 600)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "1", 600)); } } \ No newline at end of file diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/TimeWindowedKStreamImplTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/TimeWindowedKStreamImplTest.java index 610e52f2ed6..7b885b23bf2 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/TimeWindowedKStreamImplTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/TimeWindowedKStreamImplTest.java @@ -18,32 +18,33 @@ package org.apache.kafka.streams.kstream.internals; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; -import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.KeyValue; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.TopologyTestDriver; +import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.kstream.ForeachAction; import org.apache.kafka.streams.kstream.KStream; import org.apache.kafka.streams.kstream.Materialized; import org.apache.kafka.streams.kstream.Serialized; +import org.apache.kafka.streams.kstream.TimeWindowedKStream; import org.apache.kafka.streams.kstream.TimeWindows; import org.apache.kafka.streams.kstream.Windowed; -import org.apache.kafka.streams.kstream.TimeWindowedKStream; import org.apache.kafka.streams.state.WindowStore; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockAggregator; import org.apache.kafka.test.MockInitializer; import org.apache.kafka.test.MockReducer; import org.apache.kafka.test.StreamsTestUtils; -import org.apache.kafka.test.TestUtils; import org.junit.Before; -import org.junit.Rule; import org.junit.Test; import java.util.Arrays; import java.util.HashMap; import java.util.List; import java.util.Map; +import java.util.Properties; import static org.hamcrest.CoreMatchers.equalTo; import static org.hamcrest.MatcherAssert.assertThat; @@ -52,9 +53,8 @@ private static final String TOPIC = "input"; private final StreamsBuilder builder = new StreamsBuilder(); - - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); + private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); private TimeWindowedKStream<String, String> windowedStream; @Before @@ -76,7 +76,9 @@ public void apply(final Windowed<String> key, final Long value) { } }); - processData(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + } assertThat(results.get(new Windowed<>("1", new TimeWindow(0, 500))), equalTo(2L)); assertThat(results.get(new Windowed<>("2", new TimeWindow(500, 1000))), equalTo(1L)); assertThat(results.get(new Windowed<>("1", new TimeWindow(500, 1000))), equalTo(1L)); @@ -95,7 +97,9 @@ public void apply(final Windowed<String> key, final String value) { } }); - processData(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + } assertThat(results.get(new Windowed<>("1", new TimeWindow(0, 500))), equalTo("1+2")); assertThat(results.get(new Windowed<>("2", new TimeWindow(500, 1000))), equalTo("1")); assertThat(results.get(new Windowed<>("1", new TimeWindow(500, 1000))), equalTo("3")); @@ -115,29 +119,32 @@ public void apply(final Windowed<String> key, final String value) { results.put(key, value); } }); - processData(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + } assertThat(results.get(new Windowed<>("1", new TimeWindow(0, 500))), equalTo("0+1+2")); assertThat(results.get(new Windowed<>("2", new TimeWindow(500, 1000))), equalTo("0+1")); assertThat(results.get(new Windowed<>("1", new TimeWindow(500, 1000))), equalTo("0+3")); } - @SuppressWarnings("unchecked") @Test public void shouldMaterializeCount() { windowedStream.count(Materialized.<String, Long, WindowStore<Bytes, byte[]>>as("count-store") .withKeySerde(Serdes.String()) .withValueSerde(Serdes.Long())); - processData(); - final WindowStore<String, Long> windowStore = (WindowStore<String, Long>) driver.allStateStores().get("count-store"); - final List<KeyValue<Windowed<String>, Long>> data = StreamsTestUtils.toList(windowStore.fetch("1", "2", 0, 1000)); - assertThat(data, equalTo(Arrays.asList(- KeyValue.pair(new Windowed<>("1", new TimeWindow(0, 500)), 2L), - KeyValue.pair(new Windowed<>("1", new TimeWindow(500, 1000)), 1L), - KeyValue.pair(new Windowed<>("2", new TimeWindow(500, 1000)), 1L)))); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + final WindowStore<String, Long> windowStore = driver.getWindowStore("count-store"); + final List<KeyValue<Windowed<String>, Long>> data = StreamsTestUtils.toList(windowStore.fetch("1", "2", 0, 1000)); + + assertThat(data, equalTo(Arrays.asList(+ KeyValue.pair(new Windowed<>("1", new TimeWindow(0, 500)), 2L), + KeyValue.pair(new Windowed<>("1", new TimeWindow(500, 1000)), 1L), + KeyValue.pair(new Windowed<>("2", new TimeWindow(500, 1000)), 1L)))); + } } - @SuppressWarnings("unchecked") @Test public void shouldMaterializeReduced() { windowedStream.reduce(MockReducer.STRING_ADDER, @@ -145,17 +152,18 @@ public void shouldMaterializeReduced() { .withKeySerde(Serdes.String()) .withValueSerde(Serdes.String())); - processData(); - final WindowStore<String, String> windowStore = (WindowStore<String, String>) driver.allStateStores().get("reduced"); - final List<KeyValue<Windowed<String>, String>> data = StreamsTestUtils.toList(windowStore.fetch("1", "2", 0, 1000)); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + final WindowStore<String, String> windowStore = driver.getWindowStore("reduced"); + final List<KeyValue<Windowed<String>, String>> data = StreamsTestUtils.toList(windowStore.fetch("1", "2", 0, 1000)); - assertThat(data, equalTo(Arrays.asList(- KeyValue.pair(new Windowed<>("1", new TimeWindow(0, 500)), "1+2"), - KeyValue.pair(new Windowed<>("1", new TimeWindow(500, 1000)), "3"), -

KeyValue.pair(new Windowed<>("2", new TimeWindow(500, 1000)), "1")))); + assertThat(data, equalTo(Arrays.asList(+ KeyValue.pair(new Windowed<>("1", new TimeWindow(0, 500)), "1+2"), + KeyValue.pair(new Windowed<>("1", new TimeWindow(500, 1000)), "3"), + KeyValue.pair(new Windowed<>("2", new TimeWindow(500, 1000)), "1")))); + } } - @SuppressWarnings("unchecked") @Test public void shouldMaterializeAggregated() { windowedStream.aggregate(MockInitializer.STRING_INIT, @@ -164,13 +172,15 @@ public void shouldMaterializeAggregated() { .withKeySerde(Serdes.String()) .withValueSerde(Serdes.String())); - processData(); - final WindowStore<String, String> windowStore = (WindowStore<String, String>) driver.allStateStores().get("aggregated"); - final List<KeyValue<Windowed<String>, String>> data = StreamsTestUtils.toList(windowStore.fetch("1", "2", 0, 1000)); - assertThat(data, equalTo(Arrays.asList(- KeyValue.pair(new Windowed<>("1", new TimeWindow(0, 500)), "0+1+2"), - KeyValue.pair(new Windowed<>("1", new TimeWindow(500, 1000)), "0+3"), - KeyValue.pair(new Windowed<>("2", new TimeWindow(500, 1000)), "0+1")))); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props, 0L)) { + processData(driver); + final WindowStore<String, String> windowStore = driver.getWindowStore("aggregated"); + final List<KeyValue<Windowed<String>, String>> data = StreamsTestUtils.toList(windowStore.fetch("1", "2", 0, 1000)); + assertThat(data, equalTo(Arrays.asList(+ KeyValue.pair(new Windowed<>("1", new TimeWindow(0, 500)), "0+1+2"), + KeyValue.pair(new Windowed<>("1", new TimeWindow(500, 1000)), "0+3"), + KeyValue.pair(new Windowed<>("2", new TimeWindow(500, 1000)), "0+1")))); + } } @Test(expected = NullPointerException.class) @@ -227,16 +237,11 @@ public void shouldThrowNullPointerOnCountIfMaterializedIsNull() { windowedStream.count(null); } - private void processData() { - driver.setUp(builder, TestUtils.tempDirectory(), 0); - driver.setTime(10); - driver.process(TOPIC, "1", "1"); - driver.setTime(15); - driver.process(TOPIC, "1", "2"); - driver.setTime(500); - driver.process(TOPIC, "1", "3"); - driver.process(TOPIC, "2", "1"); - driver.flushState(); + private void processData(final TopologyTestDriver driver) { + driver.pipeInput(recordFactory.create(TOPIC, "1", "1", 10L)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "2", 15L)); + driver.pipeInput(recordFactory.create(TOPIC, "1", "3", 500L)); + driver.pipeInput(recordFactory.create(TOPIC, "2", "1", 500L)); } } \ No newline at end of file -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

27. guozhangwang closed pull request #4986: KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [part 2] URL: https://github.com/apache/kafka/pull/4986 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/streams/src/test/java/org/apache/kafka/streams/TopologyTestDriverWrapper.java b/streams/src/test/java/org/apache/kafka/streams/TopologyTestDriverWrapper.java new file mode 100644 index 00000000000..bec4b5f79ff --- /dev/null +++ b/streams/src/test/java/org/apache/kafka/streams/TopologyTestDriverWrapper.java @@ -0,0 +1,70 @@ +/* + * Licensed to the Apache Software Foundation (ASF) under one or more + * contributor license agreements. See the NOTICE file distributed with + * this work for additional information regarding copyright ownership. + * The ASF licenses this file to You under the Apache License, Version 2.0 + * (the "License"); you may not use this file except in compliance with + * the License. You may obtain a copy of the License at + * + * http://www.apache.org/licenses/LICENSE-2.0 + * + * Unless required by applicable law or agreed to in writing, software + * distributed under the License is distributed on an "AS IS" BASIS, + * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. + * See the License for the specific language governing permissions and + * limitations under the License. + */ +package org.apache.kafka.streams; + +import org.apache.kafka.streams.errors.StreamsException; +import org.apache.kafka.streams.processor.ProcessorContext; +import org.apache.kafka.streams.processor.internals.ProcessorContextImpl; +import org.apache.kafka.streams.processor.internals.ProcessorNode; + +import java.util.Properties; + +/** + * This class provides access to {@link TopologyTestDriver} protected methods. + * It should only be used for internal testing, in the rare occasions where the + * necessary functionality is not supported by {@link TopologyTestDriver}. + */ +public class TopologyTestDriverWrapper extends TopologyTestDriver { + + + public TopologyTestDriverWrapper(final Topology topology, + final Properties config) { + super(topology, config); + } + + /** + * Get the processor context, setting the processor whose name is given as current node + * + * @param processorName processor name to set as current node + * @return the processor context + */ + public ProcessorContext setCurrentNodeForProcessorContext(final String processorName) { + final ProcessorContext context = task.context(); + ((ProcessorContextImpl) context).setCurrentNode(getProcessor(processorName)); + return context; + } + + /** + * Get a processor by name + * + * @param name the name to search for + * @return the processor matching the search name + */ + public ProcessorNode getProcessor(final String name) { + for (final ProcessorNode node : processorTopology.processors()) { + if (node.name().equals(name)) { + return node; + } + } + for (final ProcessorNode node : globalTopology.processors()) { + if (node.name().equals(name)) { + return node; + } + } + throw new StreamsException("Could not find a processor named '" + name + "'"); + } +} diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableFilterTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableFilterTest.java index c37078df99c..2cf192b9f45 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableFilterTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableFilterTest.java @@ -16,45 +16,40 @@ */ package org.apache.kafka.streams.kstream.internals; -import org.apache.kafka.common.serialization.Serde; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; -import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.StreamsBuilder; import org.apache.kafka.streams.Topology; +import org.apache.kafka.streams.TopologyTestDriver; +import org.apache.kafka.streams.TopologyTestDriverWrapper; +import org.apache.kafka.streams.TopologyWrapper; +import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.kstream.KTable; import org.apache.kafka.streams.kstream.Materialized; import org.apache.kafka.streams.kstream.Predicate; +import org.apache.kafka.streams.processor.internals.InternalTopologyBuilder; import org.apache.kafka.streams.state.KeyValueStore; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; +import org.apache.kafka.test.MockMapper; import org.apache.kafka.test.MockProcessor; import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockReducer; -import org.apache.kafka.test.MockMapper; -import org.apache.kafka.test.TestUtils; -import org.junit.Before; -import org.junit.Rule; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; -import java.io.File; import java.util.List; +import java.util.Properties; import static org.junit.Assert.assertEquals; import static org.junit.Assert.assertNull; public class KTableFilterTest { - final private Serde<Integer> intSerde = Serdes.Integer(); - final private Serde<String> stringSerde = Serdes.String(); - private final Consumed<String, Integer> consumed = Consumed.with(stringSerde, intSerde); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); - private File stateDir = null; - - @Before - public void setUp() { - stateDir = TestUtils.tempDirectory("kafka-test"); - } + private final Consumed<String, Integer> consumed = Consumed.with(Serdes.String(), Serdes.Integer()); + private final ConsumerRecordFactory<String, Integer> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new IntegerSerializer()); + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.Integer()); private void doTestKTable(final StreamsBuilder builder, final KTable<String, Integer> table2, @@ -64,16 +59,14 @@ private void doTestKTable(final StreamsBuilder builder, table2.toStream().process(supplier); table3.toStream().process(supplier); -

```
driver.setUp(builder, stateDir, Serdes.String(), Serdes.Integer()); - - driver.process(topic, "A", 1); - driver.process(topic, "B", 2); -
driver.process(topic, "C", 3); - driver.process(topic, "D", 4); - driver.flushState(); - driver.process(topic, "A", null); - driver.process(topic,
"B", null); - driver.flushState(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { +
driver.pipeInput(recordFactory.create(topic, "A", 1)); + driver.pipeInput(recordFactory.create(topic, "B", 2)); +
driver.pipeInput(recordFactory.create(topic, "C", 3)); + driver.pipeInput(recordFactory.create(topic, "D", 4)); +
driver.pipeInput(recordFactory.create(topic, "A", null)); + driver.pipeInput(recordFactory.create(topic, "B", null)); + } final
List<MockProcessor<String, Integer>> processors = supplier.capturedProcessors(2); @@ -136,63 +129,68 @@ private void
doTestValueGetter(final StreamsBuilder builder, final KTableImpl<String, Integer, Integer> table2, final KTableImpl<String, Integer,
Integer> table3, final String topic1) { + + final Topology topology = builder.build(); + KTableValueGetterSupplier<String, Integer>
getterSupplier2 = table2.valueGetterSupplier(); KTableValueGetterSupplier<String, Integer> getterSupplier3 =
table3.valueGetterSupplier(); - driver.setUp(builder, stateDir, Serdes.String(), Serdes.Integer()); + final InternalTopologyBuilder
topologyBuilder = TopologyWrapper.getInternalTopologyBuilder(topology); +
topologyBuilder.connectProcessorAndStateStores(table2.name, getterSupplier2.storeNames()); +
topologyBuilder.connectProcessorAndStateStores(table3.name, getterSupplier3.storeNames()); - KTableValueGetter<String, Integer>
getter2 = getterSupplier2.get(); - KTableValueGetter<String, Integer> getter3 = getterSupplier3.get(); + try (final
TopologyTestDriverWrapper driver = new TopologyTestDriverWrapper(topology, props)) { - getter2.init(driver.context()); -
getter3.init(driver.context()); + KTableValueGetter<String, Integer> getter2 = getterSupplier2.get(); + KTableValueGetter<String, Integer>
getter3 = getterSupplier3.get(); - driver.process(topic1, "A", 1); - driver.process(topic1, "B", 1); - driver.process(topic1, "C", 1); +
getter2.init(driver.setCurrentNodeForProcessorContext(table2.name)); +
getter3.init(driver.setCurrentNodeForProcessorContext(table3.name)); - assertNull(getter2.get("A")); - assertNull(getter2.get("B")); -
assertNull(getter2.get("C")); + driver.pipeInput(recordFactory.create(topic1, "A", 1)); + driver.pipeInput(recordFactory.create(topic1, "B",
1)); + driver.pipeInput(recordFactory.create(topic1, "C", 1)); - assertEquals(1, (int) getter3.get("A")); - assertEquals(1, (int)
getter3.get("B")); - assertEquals(1, (int) getter3.get("C")); + assertNull(getter2.get("A")); + assertNull(getter2.get("B")); +
assertNull(getter2.get("C")); - driver.process(topic1, "A", 2); - driver.process(topic1, "B", 2); - driver.flushState(); + assertEquals(1, (int)
getter3.get("A")); + assertEquals(1, (int) getter3.get("B")); + assertEquals(1, (int) getter3.get("C")); - assertEquals(2, (int)
getter2.get("A")); - assertEquals(2, (int) getter2.get("B")); - assertNull(getter2.get("C")); + driver.pipeInput(recordFactory.create(topic1,
"A", 2)); + driver.pipeInput(recordFactory.create(topic1, "B", 2)); - assertNull(getter3.get("A")); - assertNull(getter3.get("B")); -
assertEquals(1, (int) getter3.get("C")); + assertEquals(2, (int) getter2.get("A")); + assertEquals(2, (int) getter2.get("B")); +
assertNull(getter2.get("C")); - driver.process(topic1, "A", 3); - driver.flushState(); + assertNull(getter3.get("A")); +
assertNull(getter3.get("B")); + assertEquals(1, (int) getter3.get("C")); - assertNull(getter2.get("A")); - assertEquals(2, (int)
getter2.get("B")); - assertNull(getter2.get("C")); + driver.pipeInput(recordFactory.create(topic1, "A", 3)); - assertEquals(3, (int)
getter3.get("A")); - assertNull(getter3.get("B")); - assertEquals(1, (int) getter3.get("C")); + assertNull(getter2.get("A")); + assertEquals(2,
(int) getter2.get("B")); + assertNull(getter2.get("C")); - driver.process(topic1, "A", null); - driver.process(topic1, "B", null); -
driver.flushState(); + assertEquals(3, (int) getter3.get("A")); + assertNull(getter3.get("B")); + assertEquals(1, (int) getter3.get("C")); -
assertNull(getter2.get("A")); - assertNull(getter2.get("B")); - assertNull(getter2.get("C")); + driver.pipeInput(recordFactory.create(topic1,
"A", null)); + driver.pipeInput(recordFactory.create(topic1, "B", null)); - assertNull(getter3.get("A")); - assertNull(getter3.get("B")); -
assertEquals(1, (int) getter3.get("C")); + assertNull(getter2.get("A")); + assertNull(getter2.get("B")); + assertNull(getter2.get("C")); + +
assertNull(getter3.get("A")); + assertNull(getter3.get("B")); + assertEquals(1, (int) getter3.get("C")); + } } @Test @@ -259,34 +257,34
@@ private void doTestNotSendingOldValue(final StreamsBuilder builder, builder.build().addProcessor("proc1", supplier, table1.name);
builder.build().addProcessor("proc2", supplier, table2.name); - driver.setUp(builder, stateDir, Serdes.String(), Serdes.Integer()); + try (final
TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { - driver.process(topic1, "A", 1); - driver.process(topic1,
"B", 1); - driver.process(topic1, "C", 1); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", 1)); +
driver.pipeInput(recordFactory.create(topic1, "B", 1)); + driver.pipeInput(recordFactory.create(topic1, "C", 1)); - final
List<MockProcessor<String, Integer>> processors = supplier.capturedProcessors(2); + final List<MockProcessor<String, Integer>>
processors = supplier.capturedProcessors(2); + + processors.get(0).checkAndClearProcessResult("A:(1<-null)", "B:(1<-null)", "C:(1<-
null)"); + processors.get(1).checkAndClearProcessResult("A:(null<-null)", "B:(null<-null)", "C:(null<-null)"); -
processors.get(0).checkAndClearProcessResult("A:(1<-null)", "B:(1<-null)", "C:(1<-null)"); -
processors.get(1).checkAndClearProcessResult("A:(null<-null)", "B:(null<-null)", "C:(null<-null)"); - - driver.process(topic1, "A", 2); -
driver.process(topic1, "B", 2); - driver.flushState(); - processors.get(0).checkAndClearProcessResult("A:(2<-null)", "B:(2<-null)"); -
processors.get(1).checkAndClearProcessResult("A:(2<-null)", "B:(2<-null)"); - - driver.process(topic1, "A", 3); - driver.flushState(); -
processors.get(0).checkAndClearProcessResult("A:(3<-null)"); - processors.get(1).checkAndClearProcessResult("A:(null<-null)"); - -
driver.process(topic1, "A", null); - driver.process(topic1, "B", null); - driver.flushState(); -
processors.get(0).checkAndClearProcessResult("A:(null<-null)", "B:(null<-null)"); - processors.get(1).checkAndClearProcessResult("A:
(null<-null)", "B:(null<-null)"); + driver.pipeInput(recordFactory.create(topic1, "A", 2)); + driver.pipeInput(recordFactory.create(topic1,
"B", 2)); + + processors.get(0).checkAndClearProcessResult("A:(2<-null)", "B:(2<-null)"); +
processors.get(1).checkAndClearProcessResult("A:(2<-null)", "B:(2<-null)"); + + driver.pipeInput(recordFactory.create(topic1, "A", 3));
+ + processors.get(0).checkAndClearProcessResult("A:(3<-null)"); + processors.get(1).checkAndClearProcessResult("A:(null<-null)"); +
+ driver.pipeInput(recordFactory.create(topic1, "A", null)); + driver.pipeInput(recordFactory.create(topic1, "B", null)); + +
processors.get(0).checkAndClearProcessResult("A:(null<-null)", "B:(null<-null)"); + processors.get(1).checkAndClearProcessResult("A:
(null<-null)", "B:(null<-null)"); + } } @@ -340,34 +338,34 @@ private void doTestSendingOldValue(final StreamsBuilder builder,
topology.addProcessor("proc1", supplier, table1.name); topology.addProcessor("proc2", supplier, table2.name); - driver.setUp(builder,
stateDir, Serdes.String(), Serdes.Integer()); + try (final TopologyTestDriver driver = new TopologyTestDriver(topology, props)) { -
driver.process(topic1, "A", 1); - driver.process(topic1, "B", 1); - driver.process(topic1, "C", 1); - driver.flushState(); +
driver.pipeInput(recordFactory.create(topic1, "A", 1)); + driver.pipeInput(recordFactory.create(topic1, "B", 1)); +
driver.pipeInput(recordFactory.create(topic1, "C", 1)); - final List<MockProcessor<String, Integer>> processors =
supplier.capturedProcessors(2); + final List<MockProcessor<String, Integer>> processors = supplier.capturedProcessors(2); -
processors.get(0).checkAndClearProcessResult("A:(1<-null)", "B:(1<-null)", "C:(1<-null)"); -
processors.get(1).checkEmptyAndClearProcessResult(); + processors.get(0).checkAndClearProcessResult("A:(1<-null)", "B:(1<-null)",
"C:(1<-null)"); + processors.get(1).checkEmptyAndClearProcessResult(); + + driver.pipeInput(recordFactory.create(topic1, "A", 2)); +
driver.pipeInput(recordFactory.create(topic1, "B", 2)); + + processors.get(0).checkAndClearProcessResult("A:(2<-1)", "B:(2<-1)"); +
processors.get(1).checkAndClearProcessResult("A:(2<-null)", "B:(2<-null)"); + + driver.pipeInput(recordFactory.create(topic1, "A", 3));
+ + processors.get(0).checkAndClearProcessResult("A:(3<-2)"); + processors.get(1).checkAndClearProcessResult("A:(null<-2)"); + +
driver.pipeInput(recordFactory.create(topic1, "A", null)); + driver.pipeInput(recordFactory.create(topic1, "B", null)); -
driver.process(topic1, "A", 2); - driver.process(topic1, "B", 2); - driver.flushState(); - processors.get(0).checkAndClearProcessResult("A:
(2<-1)", "B:(2<-1)"); - processors.get(1).checkAndClearProcessResult("A:(2<-null)", "B:(2<-null)"); - - driver.process(topic1, "A", 3); -
driver.flushState(); - processors.get(0).checkAndClearProcessResult("A:(3<-2)"); - processors.get(1).checkAndClearProcessResult("A:
```

```
(null<-2)"); - - driver.process(topic1, "A", null); - driver.process(topic1, "B", null); - driver.flushState(); -
processors.get(0).checkAndClearProcessResult("A:(null<-3)", "B:(null<-2)"); - processors.get(1).checkAndClearProcessResult("B:
(null<-2)"); + processors.get(0).checkAndClearProcessResult("A:(null<-3)", "B:(null<-2)"); +
processors.get(1).checkAndClearProcessResult("B:(null<-2)"); + } } @Test @@ -418,12 +416,13 @@ private void
doTestSkipNullOnMaterialization(final StreamsBuilder builder, topology.addProcessor("proc1", supplier, table1.name);
topology.addProcessor("proc2", supplier, table2.name); - driver.setUp(builder, stateDir, stringSerde, stringSerde); + final
ConsumerRecordFactory<String, String> stringRecordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new
StringSerializer()); + try (final TopologyTestDriver driver = new TopologyTestDriver(topology, props)) { - driver.process(topic1, "A",
"reject"); - driver.process(topic1, "B", "reject"); - driver.process(topic1, "C", "reject"); - driver.flushState(); +
driver.pipeInput(stringRecordFactory.create(topic1, "A", "reject")); + driver.pipeInput(stringRecordFactory.create(topic1, "B", "reject")); +
driver.pipeInput(stringRecordFactory.create(topic1, "C", "reject")); + } final List<MockProcessor<String, String>> processors =
supplier.capturedProcessors(2); processors.get(0).checkAndClearProcessResult("A:(reject<-null)", "B:(reject<-null)", "C:(reject<-null)");
@@ -437,7 +436,7 @@ public void testSkipNullOnMaterialization() { String topic1 = "topic1"; - final Consumed<String, String>
consumed = Consumed.with(stringSerde, stringSerde); + final Consumed<String, String> consumed = Consumed.with(Serdes.String(),
Serdes.String()); KTableImpl<String, String, String> table1 = (KTableImpl<String, String, String>) builder.table(topic1, consumed);
KTableImpl<String, String, String> table2 = (KTableImpl<String, String, String>) table1.filter( @@ -459,7 +458,7 @@ public void
testQueryableSkipNullOnMaterialization() { String topic1 = "topic1"; - final Consumed<String, String> consumed =
Consumed.with(stringSerde, stringSerde); + final Consumed<String, String> consumed = Consumed.with(Serdes.String(),
Serdes.String()); KTableImpl<String, String, String> table1 = (KTableImpl<String, String, String>) builder.table(topic1, consumed);
KTableImpl<String, String, String> table2 = (KTableImpl<String, String, String>) table1.filter( diff --git
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableImplTest.java
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableImplTest.java index 399e519bb4c..0d194e44816 100644 ---
a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableImplTest.java +++
b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableImplTest.java @@ -16,12 +16,17 @@ */ package
org.apache.kafka.streams.kstream.internals; -import org.apache.kafka.common.serialization.Serde; import
org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import
org.apache.kafka.common.utils.Bytes; import org.apache.kafka.common.utils.Utils; -import org.apache.kafka.streams.kstream.Consumed;
import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.Topology; +import
org.apache.kafka.streams.TopologyDescription; +import org.apache.kafka.streams.TopologyTestDriver; +import
org.apache.kafka.streams.TopologyTestDriverWrapper; +import org.apache.kafka.streams.TopologyWrapper; +import
org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.kstream.KTable; import
org.apache.kafka.streams.kstream.Materialized; import org.apache.kafka.streams.kstream.Predicate; @@ -30,10 +35,11 @@ import
org.apache.kafka.streams.kstream.ValueMapper; import org.apache.kafka.streams.kstream.ValueMapperWithKey; import
org.apache.kafka.streams.kstream.ValueTransformerWithKeySupplier; +import
org.apache.kafka.streams.processor.internals.InternalTopologyBuilder; import org.apache.kafka.streams.processor.internals.SinkNode;
import org.apache.kafka.streams.processor.internals.SourceNode; import org.apache.kafka.streams.state.KeyValueStore; -import
org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import
org.apache.kafka.test.MockAggregator; import org.apache.kafka.test.MockInitializer; import org.apache.kafka.test.MockMapper; @@
-41,36 +47,31 @@ import org.apache.kafka.test.MockProcessorSupplier; import org.apache.kafka.test.MockReducer; import
org.apache.kafka.test.MockValueJoiner; -import org.apache.kafka.test.TestUtils; +import org.apache.kafka.test.StreamsTestUtils; import
org.junit.Before; -import org.junit.Rule; import org.junit.Test; -import java.io.File; import java.lang.reflect.Field; import java.util.List;
+import java.util.Properties; import static org.easymock.EasyMock.mock; import static org.junit.Assert.assertEquals; import static
org.junit.Assert.assertNotNull; import static org.junit.Assert.assertNull; -import static org.junit.Assert.assertTrue; public class
KTableImplTest { - private final Serde<String> stringSerde = Serdes.String(); - private final Consumed<String, String> consumed =
Consumed.with(stringSerde, stringSerde); - private final Produced<String, String> produced = Produced.with(stringSerde, stringSerde); +
private final Consumed<String, String> consumed = Consumed.with(Serdes.String(), Serdes.String()); + private final Produced<String,
String> produced = Produced.with(Serdes.String(), Serdes.String()); + private final Properties props =
StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); + private final ConsumerRecordFactory<String, String>
recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); - @Rule - public final
KStreamTestDriver driver = new KStreamTestDriver(); - private File stateDir = null; private StreamsBuilder builder; private
KTable<String, String> table; @Before public void setUp() { - stateDir = TestUtils.tempDirectory("kafka-test"); builder = new
StreamsBuilder(); table = builder.table("test"); } @@ -110,17 +111,12 @@ public boolean test(String key, Integer value) {
table4.toStream().process(supplier); - driver.setUp(builder, stateDir); - - driver.process(topic1, "A", "01"); - driver.flushState(); -
driver.process(topic1, "B", "02"); - driver.flushState(); - driver.process(topic1, "C", "03"); - driver.flushState(); - driver.process(topic1,
"D", "04"); - driver.flushState(); - driver.flushState(); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(),
props)) { + driver.pipeInput(recordFactory.create(topic1, "A", "01")); + driver.pipeInput(recordFactory.create(topic1, "B", "02")); +
driver.pipeInput(recordFactory.create(topic1, "C", "03")); + driver.pipeInput(recordFactory.create(topic1, "D", "04")); + } final
List<MockProcessor<String, Object>> processors = supplier.capturedProcessors(4); assertEquals(Utils.mkList("A:01", "B:02", "C:03",
"D:04"), processors.get(0).processed); @@ -156,104 +152,109 @@ public boolean test(String key, Integer value) {
table1.toStream().to(topic2, produced); final KTableImpl<String, String, String> table4 = (KTableImpl<String, String, String>)
builder.table(topic2, consumed); + final Topology topology = builder.build(); + final KTableValueGetterSupplier<String, String>
getterSupplier1 = table1.valueGetterSupplier(); final KTableValueGetterSupplier<String, Integer> getterSupplier2 =
table2.valueGetterSupplier(); final KTableValueGetterSupplier<String, Integer> getterSupplier3 = table3.valueGetterSupplier(); final
KTableValueGetterSupplier<String, String> getterSupplier4 = table4.valueGetterSupplier(); - driver.setUp(builder, stateDir, null, null); +
final InternalTopologyBuilder topologyBuilder = TopologyWrapper.getInternalTopologyBuilder(topology); +
topologyBuilder.connectProcessorAndStateStores(table1.name, getterSupplier1.storeNames()); +
topologyBuilder.connectProcessorAndStateStores(table2.name, getterSupplier2.storeNames()); +
topologyBuilder.connectProcessorAndStateStores(table3.name, getterSupplier3.storeNames()); +
topologyBuilder.connectProcessorAndStateStores(table4.name, getterSupplier4.storeNames()); + + try (final TopologyTestDriverWrapper
driver = new TopologyTestDriverWrapper(topology, props)) { - // two state store should be created - assertEquals(2,
driver.allStateStores().size()); + assertEquals(2, driver.getAllStateStores().size()); - final KTableValueGetter<String, String> getter1 =
getterSupplier1.get(); - getter1.init(driver.context()); - final KTableValueGetter<String, Integer> getter2 = getterSupplier2.get(); -
getter2.init(driver.context()); - final KTableValueGetter<String, Integer> getter3 = getterSupplier3.get(); - getter3.init(driver.context()); -
final KTableValueGetter<String, String> getter4 = getterSupplier4.get(); - getter4.init(driver.context()); + final KTableValueGetter<String,
String> getter1 = getterSupplier1.get(); + final KTableValueGetter<String, Integer> getter2 = getterSupplier2.get(); + final
KTableValueGetter<String, Integer> getter3 = getterSupplier3.get(); + final KTableValueGetter<String, String> getter4 =
```

getterSupplier4.get(); - driver.process(topic1, "A", "01"); - driver.process(topic1, "B", "01"); - driver.process(topic1, "C", "01"); - driver.flushState(); + getter1.init(driver.setCurrentNodeForProcessorContext(table1.name)); + getter2.init(driver.setCurrentNodeForProcessorContext(table2.name)); + getter3.init(driver.setCurrentNodeForProcessorContext(table3.name)); + getter4.init(driver.setCurrentNodeForProcessorContext(table4.name)); - assertEquals("01", getter1.get("A")); - assertEquals("01", getter1.get("B")); - assertEquals("01", getter1.get("C")); + driver.pipeInput(recordFactory.create(topic1, "A", "01")); + driver.pipeInput(recordFactory.create(topic1, "B", "01")); + driver.pipeInput(recordFactory.create(topic1, "C", "01")); - assertEquals(new Integer(1), getter2.get("A")); - assertEquals(new Integer(1), getter2.get("B")); - assertEquals(new Integer(1), getter2.get("C")); + assertEquals("01", getter1.get("A")); + assertEquals("01", getter1.get("B")); + assertEquals("01", getter1.get("C")); - assertNull(getter3.get("A")); - assertNull(getter3.get("B")); - assertNull(getter3.get("C")); + assertEquals(new Integer(1), getter2.get("A")); + assertEquals(new Integer(1), getter2.get("B")); + assertEquals(new Integer(1), getter2.get("C")); - assertEquals("01", getter4.get("A")); - assertEquals("01", getter4.get("B")); - assertEquals("01", getter4.get("C")); + assertNull(getter3.get("A")); + assertNull(getter3.get("B")); + assertNull(getter3.get("C")); - driver.process(topic1, "A", "02"); - driver.process(topic1, "B", "02"); - driver.flushState(); + assertEquals("01", getter4.get("A")); + assertEquals("01", getter4.get("B")); + assertEquals("01", getter4.get("C")); - assertEquals("02", getter1.get("A")); - assertEquals("02", getter1.get("B")); - assertEquals("01", getter1.get("C")); + driver.pipeInput(recordFactory.create(topic1, "A", "02")); + driver.pipeInput(recordFactory.create(topic1, "B", "02")); - assertEquals(new Integer(2), getter2.get("A")); - assertEquals(new Integer(2), getter2.get("B")); - assertEquals(new Integer(1), getter2.get("C")); + assertEquals("02", getter1.get("A")); + assertEquals("02", getter1.get("B")); + assertEquals("01", getter1.get("C")); - assertEquals(new Integer(2), getter3.get("A")); - assertEquals(new Integer(2), getter3.get("B")); - assertNull(getter3.get("C")); + assertEquals(new Integer(2), getter2.get("A")); + assertEquals(new Integer(2), getter2.get("B")); + assertEquals(new Integer(1), getter2.get("C")); - assertEquals("02", getter4.get("A")); - assertEquals("02", getter4.get("B")); - assertEquals("01", getter4.get("C")); + assertEquals(new Integer(2), getter3.get("A")); + assertEquals(new Integer(2), getter3.get("B")); + assertNull(getter3.get("C")); - driver.process(topic1, "A", "03"); - driver.flushState(); + assertEquals("02", getter4.get("A")); + assertEquals("02", getter4.get("B")); + assertEquals("01", getter4.get("C")); - assertEquals("03", getter1.get("A")); - assertEquals("02", getter1.get("B")); - assertEquals("01", getter1.get("C")); + driver.pipeInput(recordFactory.create(topic1, "A", "03")); - assertEquals(new Integer(3), getter2.get("A")); - assertEquals(new Integer(2), getter2.get("B")); - assertEquals(new Integer(1), getter2.get("C")); + assertEquals("03", getter1.get("A")); + assertEquals("02", getter1.get("B")); + assertEquals("01", getter1.get("C")); - assertNull(getter3.get("A")); - assertEquals(new Integer(2), getter3.get("B")); - assertNull(getter3.get("C")); + assertEquals(new Integer(3), getter2.get("A")); + assertEquals(new Integer(2), getter2.get("B")); + assertEquals(new Integer(1), getter2.get("C")); - assertEquals("03", getter4.get("A")); - assertEquals("02", getter4.get("B")); - assertEquals("01", getter4.get("C")); + assertNull(getter3.get("A")); + assertEquals(new Integer(2), getter3.get("B")); + assertNull(getter3.get("C")); - driver.process(topic1, "A", null); - driver.flushState(); + assertEquals("03", getter4.get("A")); + assertEquals("02", getter4.get("B")); + assertEquals("01", getter4.get("C")); - assertNull(getter1.get("A")); - assertEquals("02", getter1.get("B")); - assertEquals("01", getter1.get("C")); + driver.pipeInput(recordFactory.create(topic1, "A", (String) null)); + assertNull(getter1.get("A")); + assertEquals("02", getter1.get("B")); + assertEquals("01", getter1.get("C")); - assertNull(getter2.get("A")); - assertEquals(new Integer(2), getter2.get("B")); - assertEquals(new Integer(1), getter2.get("C")); - assertNull(getter3.get("A")); - assertEquals(new Integer(2), getter3.get("B")); - assertNull(getter3.get("C")); + assertNull(getter2.get("A")); + assertEquals(new Integer(2), getter2.get("B")); + assertEquals(new Integer(1), getter2.get("C")); - assertNull(getter4.get("A")); - assertEquals("02", getter4.get("B")); - assertEquals("01", getter4.get("C")); + assertNull(getter3.get("A")); + assertEquals(new Integer(2), getter3.get("B")); + assertNull(getter3.get("C")); + + assertNull(getter4.get("A")); + assertEquals("02", getter4.get("B")); + assertEquals("01", getter4.get("C")); + } } @Test @@ -282,11 +283,9 @@ public boolean test(String key, Integer value) { } }); - driver.setUp(builder, stateDir, null, null); - driver.setTime(0L); - - // two state stores should be created - assertEquals(2, driver.allStateStores().size()); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + assertEquals(2, driver.getAllStateStores().size()); + } } @Test @@ -323,15 +322,25 @@ public String apply(String v1, Integer v2) { } }); - driver.setUp(builder, stateDir, null, null); - driver.setTime(0L); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + assertEquals(2, driver.getAllStateStores().size()); + } + } - // two state store should be created - assertEquals(2, driver.allStateStores().size()); + private void assertTopologyContainsProcessor(final Topology topology, final String processorName) { + for (final TopologyDescription.Subtopology subtopology: topology.describe().subtopologies()) { + for (final TopologyDescription.Node node: subtopology.nodes()) { + if (node.name().equals(processorName)) { + return; + } + } + } + throw new AssertionError("No processor named '" + processorName + "'" + + "found in the provided Topology:\n" + topology.describe()); } @Test - public void testRepartition() throws NoSuchFieldException, IllegalAccessException { + public void shouldCreateSourceAndSinkNodesForRepartitioningTopic() throws NoSuchFieldException, IllegalAccessException { final String topic1 = "topic1"; final String storeName1 = "storeName1"; @@ -341,8 +350,8 @@ public void testRepartition() throws NoSuchFieldException, IllegalAccessExceptio (KTableImpl<String, String, String>) builder.table(topic1, consumed, Materialized.<String, String, KeyValueStore<Bytes, byte[]>>as(storeName1) - .withKeySerde(stringSerde) - .withValueSerde(stringSerde) + .withKeySerde(Serdes.String()) + .withValueSerde(Serdes.String())); table1.groupBy(MockMapper.<String, String>noOpKeyValueMapper()) @@ -352,27 +361,26 @@ public void testRepartition() throws NoSuchFieldException, IllegalAccessExceptio table1.groupBy(MockMapper.<String, String>noOpKeyValueMapper()) .reduce(MockReducer.STRING_ADDER, MockReducer.STRING_REMOVER, Materialized.<String, String, KeyValueStore<Bytes, byte[]>>as("mock-result2")); - driver.setUp(builder, stateDir, stringSerde, stringSerde); - driver.setTime(0L); + final Topology topology = builder.build(); + try (final TopologyTestDriverWrapper driver = new TopologyTestDriverWrapper(topology, props)) { - // three state store should be created, one for source, one for aggregate and one for reduce - assertEquals(3, driver.allStateStores().size()); + assertEquals(3, driver.getAllStateStores().size()); - // contains the corresponding repartition source / sink nodes - assertTrue(driver.allProcessorNames().contains("KSTREAM-SINK-0000000003")); - assertTrue(driver.allProcessorNames().contains("KSTREAM-SOURCE-0000000004")); - assertTrue(driver.allProcessorNames().contains("KSTREAM-SINK-0000000007")); - assertTrue(driver.allProcessorNames().contains("KSTREAM-SOURCE-0000000008")); + assertTopologyContainsProcessor(topology, "KSTREAM-SINK-0000000003"); + assertTopologyContainsProcessor(topology, "KSTREAM-SOURCE-0000000004"); + assertTopologyContainsProcessor(topology, "KSTREAM-SINK-0000000007"); + assertTopologyContainsProcessor(topology, "KSTREAM-SOURCE-0000000008"); - Field valSerializerField = ((SinkNode) driver.processor("KSTREAM-SINK-0000000003")).getClass().getDeclaredField("valSerializer"); - Field valDeserializerField = ((SourceNode) driver.processor("KSTREAM-SOURCE-0000000004")).getClass().getDeclaredField("valDeserializer"); - valSerializerField.setAccessible(true); - valDeserializerField.setAccessible(true); + Field valSerializerField = ((SinkNode) driver.getProcessor("KSTREAM-SINK-0000000003")).getClass().getDeclaredField("valSerializer"); + Field valDeserializerField = ((SourceNode) driver.getProcessor("KSTREAM-SOURCE-0000000004")).getClass().getDeclaredField("valDeserializer"); + valSerializerField.setAccessible(true); + valDeserializerField.setAccessible(true); - assertNotNull(((ChangedSerializer) valSerializerField.get(driver.processor("KSTREAM-SINK-0000000003"))).inner()); - assertNotNull(((ChangedDeserializer) valDeserializerField.get(driver.processor("KSTREAM-SOURCE-0000000004"))).inner()); - assertNotNull(((ChangedSerializer) valSerializerField.get(driver.processor("KSTREAM-SINK-0000000007"))).inner()); - assertNotNull(((ChangedDeserializer) valDeserializerField.get(driver.processor("KSTREAM-SOURCE-0000000008"))).inner()); + assertNotNull(((ChangedSerializer) valSerializerField.get(driver.getProcessor("KSTREAM-SINK-0000000003"))).inner()); + assertNotNull(((ChangedDeserializer) valDeserializerField.get(driver.getProcessor("KSTREAM-SOURCE-0000000004"))).inner()); + assertNotNull(((ChangedSerializer) valSerializerField.get(driver.getProcessor("KSTREAM-SINK-0000000007"))).inner()); + assertNotNull(((ChangedDeserializer) valDeserializerField.get(driver.getProcessor("KSTREAM-SOURCE-0000000008"))).inner()); + } } @Test(expected = NullPointerException.class) diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableMapValuesTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableMapValuesTest.java index a54e43e92c3..a01d5cb215a 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableMapValuesTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableMapValuesTest.java @@ -16,27 +16,30 @@ */ package org.apache.kafka.streams.kstream.internals; -import org.apache.kafka.common.serialization.Serde; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Bytes; import org.apache.kafka.common.utils.Utils; -import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.Topology; +import org.apache.kafka.streams.TopologyTestDriver; +import org.apache.kafka.streams.TopologyTestDriverWrapper; +import org.apache.kafka.streams.TopologyWrapper; +import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.kstream.KTable; import org.apache.kafka.streams.kstream.Materialized; import org.apache.kafka.streams.kstream.Predicate; import org.apache.kafka.streams.kstream.Produced; import org.apache.kafka.streams.kstream.ValueMapper; +import org.apache.kafka.streams.processor.internals.InternalTopologyBuilder; import org.apache.kafka.streams.state.KeyValueStore; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessor; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; -import org.junit.Before; -import org.junit.Rule; +import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; -import java.io.File; +import java.util.Properties; import static org.junit.Assert.assertEquals; import static org.junit.Assert.assertFalse; @@ -45,27 +48,19 @@ public class KTableMapValuesTest { - private final Serde<String> stringSerde = Serdes.String(); - private final Consumed<String, String> consumed = Consumed.with(stringSerde, stringSerde); - private final Produced<String, String> produced = Produced.with(stringSerde, stringSerde); - @Rule - public final KStreamTestDriver driver = new KStreamTestDriver(); - private File stateDir = null; - - @Before - public void setUp() { - stateDir = TestUtils.tempDirectory("kafka-test"); - } + private final Consumed<String, String> consumed = Consumed.with(Serdes.String(), Serdes.String()); + private final Produced<String, String> produced = Produced.with(Serdes.String(), Serdes.String()); + private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new StringSerializer(), new StringSerializer()); + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.String(), Serdes.String()); private void doTestKTable(final StreamsBuilder builder, final String topic1, final MockProcessorSupplier<String, Integer> supplier) { - driver.setUp(builder, stateDir, Serdes.String(), Serdes.String()); - - driver.process(topic1, "A", "1"); - driver.process(topic1, "B", "2"); - driver.process(topic1, "C", "3"); - driver.process(topic1, "D", "4"); - driver.flushState(); - assertEquals(Utils.mkList("A:1", "B:2", "C:3", "D:4"), supplier.theCapturedProcessor().processed); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(topic1, "A", "1")); + driver.pipeInput(recordFactory.create(topic1, "B", "2")); + driver.pipeInput(recordFactory.create(topic1, "C", "3")); + driver.pipeInput(recordFactory.create(topic1, "D", "4")); + assertEquals(Utils.mkList("A:1", "B:2", "C:3", "D:4"), supplier.theCapturedProcessor().processed); + } } @Test @@ -114,99 +109,106 @@ private void doTestValueGetter(final StreamsBuilder builder, final KTableImpl<String, String, Integer> table2, final KTableImpl<String, Integer, Integer> table3, final KTableImpl<String, String, String> table4) { + + final Topology topology = builder.build(); + final KTableValueGetterSupplier<String, String> getterSupplier1 = table1.valueGetterSupplier(); final KTableValueGetterSupplier<String, Integer> getterSupplier2 = table2.valueGetterSupplier(); final KTableValueGetterSupplier<String, Integer> getterSupplier3 = table3.valueGetterSupplier(); final KTableValueGetterSupplier<String, String> getterSupplier4 = table4.valueGetterSupplier(); - driver.setUp(builder, stateDir, Serdes.String(), Serdes.String()); - final KTableValueGetter<String, String> getter1 = getterSupplier1.get(); - getter1.init(driver.context()); - final KTableValueGetter<String, Integer> getter2 = getterSupplier2.get(); - getter2.init(driver.context()); - final KTableValueGetter<String, Integer> getter3 = getterSupplier3.get(); - getter3.init(driver.context()); - final KTableValueGetter<String, String> getter4 = getterSupplier4.get(); - getter4.init(driver.context()); - - driver.process(topic1, "A", "01"); - driver.process(topic1, "B", "01"); - driver.process(topic1, "C", "01"); - driver.flushState(); - - assertEquals("01", getter1.get("A")); - assertEquals("01", getter1.get("B")); - assertEquals("01", getter1.get("C")); - - assertEquals(new Integer(1), getter2.get("A")); - assertEquals(new Integer(1), getter2.get("B")); - assertEquals(new Integer(1), getter2.get("C")); - - assertNull(getter3.get("A")); - assertNull(getter3.get("B")); - assertNull(getter3.get("C")); - - assertEquals("01", getter4.get("A")); - assertEquals("01", getter4.get("B")); - assertEquals("01", getter4.get("C")); - - driver.process(topic1, "A", "02"); - driver.process(topic1, "B", "02"); - driver.flushState(); - - assertEquals("02", getter1.get("A")); - assertEquals("02", getter1.get("B")); - assertEquals("01", getter1.get("C")); - - assertEquals(new Integer(2), getter2.get("A")); - assertEquals(new Integer(2), getter2.get("B")); - assertEquals(new Integer(1), getter2.get("C")); - - assertEquals(new Integer(2), getter3.get("A")); - assertEquals(new Integer(2), getter3.get("B")); - assertNull(getter3.get("C")); - - assertEquals("02", getter4.get("A")); - assertEquals("02", getter4.get("B")); - assertEquals("01", getter4.get("C")); - - driver.process(topic1, "A", "03"); - driver.flushState(); - - assertEquals("03", getter1.get("A")); - assertEquals("02", getter1.get("B")); - assertEquals("01", getter1.get("C")); - - assertEquals(new Integer(3), getter2.get("A")); - assertEquals(new Integer(2), getter2.get("B")); - assertEquals(new Integer(1), getter2.get("C")); - - assertNull(getter3.get("A")); - assertEquals(new Integer(2), getter3.get("B")); - assertNull(getter3.get("C")); - - assertEquals("03", getter4.get("A")); - assertEquals("02", getter4.get("B")); - assertEquals("01", getter4.get("C")); - - driver.process(topic1, "A", null); - driver.flushState(); - - assertNull(getter1.get("A")); - assertEquals("02", getter1.get("B")); - assertEquals("01", getter1.get("C")); - - assertNull(getter2.get("A")); - assertEquals(new Integer(2), getter2.get("B")); - assertEquals(new Integer(1), getter2.get("C")); - - assertNull(getter3.get("A")); - assertEquals(new Integer(2), getter3.get("B")); - assertNull(getter3.get("C")); - - assertNull(getter4.get("A")); - assertEquals("02", getter4.get("B")); - assertEquals("01", getter4.get("C")); + final InternalTopologyBuilder topologyBuilder = TopologyWrapper.getInternalTopologyBuilder(topology); + topologyBuilder.connectProcessorAndStateStores(table1.name, getterSupplier1.storeNames()); + topologyBuilder.connectProcessorAndStateStores(table2.name, getterSupplier2.storeNames()); + topologyBuilder.connectProcessorAndStateStores(table3.name, getterSupplier3.storeNames()); + topologyBuilder.connectProcessorAndStateStores(table4.name, getterSupplier4.storeNames()); + + try (final TopologyTestDriverWrapper driver = new TopologyTestDriverWrapper(builder.build(), props)) { + KTableValueGetter<String, String> getter1 = getterSupplier1.get(); + KTableValueGetter<String, Integer> getter2 = getterSupplier2.get(); + KTableValueGetter<String, Integer> getter3 = getterSupplier3.get(); + KTableValueGetter<String, String> getter4 = getterSupplier4.get(); + + getter1.init(driver.setCurrentNodeForProcessorContext(table1.name)); + getter2.init(driver.setCurrentNodeForProcessorContext(table2.name)); + getter3.init(driver.setCurrentNodeForProcessorContext(table3.name)); +

getter4.init(driver.setCurrentNodeForProcessorContext(table4.name)); + + driver.pipeInput(recordFactory.create(topic1, "A", "01")); + driver.pipeInput(recordFactory.create(topic1, "B", "01")); + driver.pipeInput(recordFactory.create(topic1, "C", "01")); + + assertEquals("01", getter1.get("A")); + assertEquals("01", getter1.get("B")); + assertEquals("01", getter1.get("C")); + + assertEquals(new Integer(1), getter2.get("A")); + assertEquals(new Integer(1), getter2.get("B")); + assertEquals(new Integer(1), getter2.get("C")); + + assertNull(getter3.get("A")); + assertNull(getter3.get("B")); + assertNull(getter3.get("C")); + + assertEquals("01", getter4.get("A")); + assertEquals("01", getter4.get("B")); + assertEquals("01", getter4.get("C")); + + driver.pipeInput(recordFactory.create(topic1, "A", "02")); + driver.pipeInput(recordFactory.create(topic1, "B", "02")); + + assertEquals("02", getter1.get("A")); + assertEquals("02", getter1.get("B")); + assertEquals("01", getter1.get("C")); + + assertEquals(new Integer(2), getter2.get("A")); + assertEquals(new Integer(2), getter2.get("B")); + assertEquals(new Integer(1), getter2.get("C")); + + assertEquals(new Integer(2), getter3.get("A")); + assertEquals(new Integer(2), getter3.get("B")); + assertNull(getter3.get("C")); + + assertEquals("02", getter4.get("A")); + assertEquals("02", getter4.get("B")); + assertEquals("01", getter4.get("C")); + + driver.pipeInput(recordFactory.create(topic1, "A", "03")); + + assertEquals("03", getter1.get("A")); + assertEquals("02", getter1.get("B")); + assertEquals("01", getter1.get("C")); + + assertEquals(new Integer(3), getter2.get("A")); + assertEquals(new Integer(2), getter2.get("B")); + assertEquals(new Integer(1), getter2.get("C")); + + assertNull(getter3.get("A")); + assertEquals(new Integer(2), getter3.get("B")); + assertNull(getter3.get("C")); + + assertEquals("03", getter4.get("A")); + assertEquals("02", getter4.get("B")); + assertEquals("01", getter4.get("C")); + + driver.pipeInput(recordFactory.create(topic1, "A", (String) null)); + + assertNull(getter1.get("A")); + assertEquals("02", getter1.get("B")); + assertEquals("01", getter1.get("C")); + + assertNull(getter2.get("A")); + assertEquals(new Integer(2), getter2.get("B")); + assertEquals(new Integer(1), getter2.get("C")); + + assertNull(getter3.get("A")); + assertEquals(new Integer(2), getter3.get("B")); + assertNull(getter3.get("C")); + + assertNull(getter4.get("A")); + assertEquals("02", getter4.get("B")); + assertEquals("01", getter4.get("C")); + } } @Test @@ -244,6 +246,8 @@ public void testQueryableValueGetter() { final String topic1 = "topic1"; final String topic2 = "topic2"; + final String storeName2 = "anyMapName"; + final String storeName3 = "anyFilterName"; final KTableImpl<String, String, String> table1 = (KTableImpl<String, String, String>) builder.table(topic1, consumed); @@ -253,14 +257,14 @@ public void testQueryableValueGetter() { public Integer apply(String value) { return new Integer(value); } - }, Materialized.<String, Integer, KeyValueStore<Bytes, byte[]>>as("anyMapName").withValueSerde(Serdes.Integer())); + }, Materialized.<String, Integer, KeyValueStore<Bytes, byte[]>>as(storeName2).withValueSerde(Serdes.Integer())); final KTableImpl<String, Integer, Integer> table3 = (KTableImpl<String, Integer, Integer>) table2.filter(new Predicate<String, Integer>() { @Override public boolean test(String key, Integer value) { return (value % 2) == 0; } - }, Materialized.<String, Integer, KeyValueStore<Bytes, byte[]>>as("anyFilterName").withValueSerde(Serdes.Integer())); + }, Materialized.<String, Integer, KeyValueStore<Bytes, byte[]>>as(storeName3).withValueSerde(Serdes.Integer())); table1.toStream().to(topic2, produced); final KTableImpl<String, String, String> table4 = (KTableImpl<String, String, String>) builder.table(topic2, consumed); @@ -285,37 +289,34 @@ public Integer apply(String value) { final MockProcessorSupplier<String, Integer> supplier = new MockProcessorSupplier<>(); - builder.build().addProcessor("proc", supplier, table2.name); + final Topology topology = builder.build().addProcessor("proc", supplier, table2.name); - driver.setUp(builder, stateDir); + try (final TopologyTestDriver driver = new TopologyTestDriver(topology, props)) { - final MockProcessor<String, Integer> proc = supplier.theCapturedProcessor(); + final MockProcessor<String, Integer> proc = supplier.theCapturedProcessor(); - assertFalse(table1.sendingOldValueEnabled()); - assertFalse(table2.sendingOldValueEnabled()); + assertFalse(table1.sendingOldValueEnabled()); + assertFalse(table2.sendingOldValueEnabled()); - driver.process(topic1, "A", "01"); - driver.process(topic1, "B", "01"); - driver.process(topic1, "C", "01"); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", "01")); + driver.pipeInput(recordFactory.create(topic1, "B", "01")); + driver.pipeInput(recordFactory.create(topic1, "C", "01")); - proc.checkAndClearProcessResult("A:(1<-null)", "B:(1<-null)", "C:(1<-null)"); + proc.checkAndClearProcessResult("A:(1<-null)", "B:(1<-null)", "C:(1<-null)"); - driver.process(topic1, "A", "02"); - driver.process(topic1, "B", "02"); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", "02")); + driver.pipeInput(recordFactory.create(topic1, "B", "02")); - proc.checkAndClearProcessResult("A:(2<-null)", "B:(2<-null)"); + proc.checkAndClearProcessResult("A:(2<-null)", "B:(2<-null)"); - driver.process(topic1, "A", "03"); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", "03")); - proc.checkAndClearProcessResult("A:(3<-null)"); + proc.checkAndClearProcessResult("A:(3<-null)"); - driver.process(topic1, "A", null); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", (String) null)); - proc.checkAndClearProcessResult("A:(null<-null)"); + proc.checkAndClearProcessResult("A:(null<-null)"); + } } @Test @@ -340,34 +341,31 @@ public Integer apply(String value) { builder.build().addProcessor("proc", supplier, table2.name); - driver.setUp(builder, stateDir); + try (final TopologyTestDriver driver = new TopologyTestDriver(builder.build(), props)) { - final MockProcessor<String, Integer> proc = supplier.theCapturedProcessor(); + final MockProcessor<String, Integer> proc = supplier.theCapturedProcessor(); - assertTrue(table1.sendingOldValueEnabled()); - assertTrue(table2.sendingOldValueEnabled()); + assertTrue(table1.sendingOldValueEnabled()); + assertTrue(table2.sendingOldValueEnabled()); - driver.process(topic1, "A", "01"); - driver.process(topic1, "B", "01"); - driver.process(topic1, "C", "01"); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", "01")); + driver.pipeInput(recordFactory.create(topic1, "B", "01")); + driver.pipeInput(recordFactory.create(topic1, "C", "01")); - proc.checkAndClearProcessResult("A:(1<-null)", "B:(1<-null)", "C:(1<-null)"); + proc.checkAndClearProcessResult("A:(1<-null)", "B:(1<-null)", "C:(1<-null)"); - driver.process(topic1, "A", "02"); - driver.process(topic1, "B", "02"); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", "02")); + driver.pipeInput(recordFactory.create(topic1, "B", "02")); - proc.checkAndClearProcessResult("A:(2<-1)", "B:(2<-1)"); + proc.checkAndClearProcessResult("A:(2<-1)", "B:(2<-1)"); - driver.process(topic1, "A", "03"); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", "03")); - proc.checkAndClearProcessResult("A:(3<-2)"); + proc.checkAndClearProcessResult("A:(3<-2)"); - driver.process(topic1, "A", null); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", (String) null)); - proc.checkAndClearProcessResult("A:(null<-3)"); + proc.checkAndClearProcessResult("A:(null<-3)"); + } } } diff --git a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableSourceTest.java b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableSourceTest.java index 504d8414aae..80a60ab2f20 100644 --- a/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableSourceTest.java +++ b/streams/src/test/java/org/apache/kafka/streams/kstream/internals/KTableSourceTest.java @@ -16,22 +16,26 @@ */ package org.apache.kafka.streams.kstream.internals; -import org.apache.kafka.common.serialization.Serde; +import org.apache.kafka.common.serialization.IntegerSerializer; import org.apache.kafka.common.serialization.Serdes; +import org.apache.kafka.common.serialization.StringSerializer; import org.apache.kafka.common.utils.Utils; -import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.StreamsBuilder; +import org.apache.kafka.streams.Topology; +import org.apache.kafka.streams.TopologyTestDriver; +import org.apache.kafka.streams.TopologyTestDriverWrapper; +import org.apache.kafka.streams.TopologyWrapper; +import org.apache.kafka.streams.kstream.Consumed; import org.apache.kafka.streams.kstream.KTable; +import org.apache.kafka.streams.processor.internals.InternalTopologyBuilder; import org.apache.kafka.streams.processor.internals.testutil.LogCaptureAppender; -import org.apache.kafka.test.KStreamTestDriver; +import org.apache.kafka.streams.test.ConsumerRecordFactory; import org.apache.kafka.test.MockProcessor; import org.apache.kafka.test.MockProcessorSupplier; -import org.apache.kafka.test.TestUtils; -import org.junit.Before; -import org.junit.Rule;

```
+import org.apache.kafka.test.StreamsTestUtils; import org.junit.Test; -import java.io.File; +import java.util.Properties; import static
org.apache.kafka.test.StreamsTestUtils.getMetricByName; import static org.hamcrest.CoreMatchers.hasItem; @@ -42,17 +46,9 @@
public class KTableSourceTest { - final private Serde<String> stringSerde = Serdes.String(); - private final Consumed<String, String>
stringConsumed = Consumed.with(stringSerde, stringSerde); - final private Serde<Integer> intSerde = Serdes.Integer(); - @Rule - public
final KStreamTestDriver driver = new KStreamTestDriver(); - private File stateDir = null; - - @Before - public void setUp() { - stateDir =
TestUtils.tempDirectory("kafka-test"); - } + private final Consumed<String, String> stringConsumed = Consumed.with(Serdes.String(),
Serdes.String()); + private final ConsumerRecordFactory<String, String> recordFactory = new ConsumerRecordFactory<>(new
StringSerializer(), new StringSerializer()); + private final Properties props = StreamsTestUtils.topologyTestConfig(Serdes.String(),
Serdes.String()); @Test public void testKTable() { @@ -60,38 +56,38 @@ public void testKTable() { final String topic1 = "topic1"; -
final KTable<String, Integer> table1 = builder.table(topic1, Consumed.with(stringSerde, intSerde)); + final KTable<String, Integer> table1
= builder.table(topic1, Consumed.with(Serdes.String(), Serdes.Integer())); final MockProcessorSupplier<String, Integer> supplier = new
MockProcessorSupplier<>(); table1.toStream().process(supplier); - driver.setUp(builder, stateDir); - driver.process(topic1, "A", 1); -
driver.process(topic1, "B", 2); - driver.process(topic1, "C", 3); - driver.process(topic1, "D", 4); - driver.flushState(); - driver.process(topic1,
"A", null); - driver.process(topic1, "B", null); - driver.flushState(); + final ConsumerRecordFactory<String, Integer> integerFactory = new
ConsumerRecordFactory<>(new StringSerializer(), new IntegerSerializer()); + try (final TopologyTestDriver driver = new
TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(integerFactory.create(topic1, "A", 1)); +
driver.pipeInput(integerFactory.create(topic1, "B", 2)); + driver.pipeInput(integerFactory.create(topic1, "C", 3)); +
driver.pipeInput(integerFactory.create(topic1, "D", 4)); + driver.pipeInput(integerFactory.create(topic1, "A", null)); +
driver.pipeInput(integerFactory.create(topic1, "B", null)); + } assertEquals(Utils.mkList("A:1", "B:2", "C:3", "D:4", "A:null", "B:null"),
supplier.theCapturedProcessor().processed); } @Test public void kTableShouldLogAndMeterOnSkippedRecords() { - final
StreamsBuilder streamsBuilder = new StreamsBuilder(); + final StreamsBuilder builder = new StreamsBuilder(); final String topic =
"topic"; - streamsBuilder.table(topic, Consumed.with(stringSerde, intSerde)); + builder.table(topic, stringConsumed); final
LogCaptureAppender appender = LogCaptureAppender.createAndRegister(); - driver.setUp(streamsBuilder, stateDir); -
driver.process(topic, null, "value"); - driver.flushState(); - LogCaptureAppender.unregister(appender); + try (final TopologyTestDriver
driver = new TopologyTestDriver(builder.build(), props)) { + driver.pipeInput(recordFactory.create(topic, null, "value")); +
LogCaptureAppender.unregister(appender); - assertEquals(1.0, getMetricByName(driver.context().metrics().metrics(), "skipped-records-
total", "stream-metrics").metricValue()); - assertThat(appender.getMessages(), hasItem("Skipping record due to null key. topic=[topic]
partition=[-1] offset=[-1]")); + assertEquals(1.0, getMetricByName(driver.metrics(), "skipped-records-total", "stream-
metrics").metricValue()); + assertThat(appender.getMessages(), hasItem("Skipping record due to null key. topic=[topic] partition=[0]
offset=[0]")); + } } @Test @@ -102,39 +98,45 @@ public void testValueGetter() { final KTableImpl<String, String, String> table1 =
(KTableImpl<String, String, String>) builder.table(topic1, stringConsumed); + final Topology topology = builder.build(); + final
KTableValueGetterSupplier<String, String> getterSupplier1 = table1.valueGetterSupplier(); - driver.setUp(builder, stateDir); - final
KTableValueGetter<String, String> getter1 = getterSupplier1.get(); - getter1.init(driver.context()); + final InternalTopologyBuilder
topologyBuilder = TopologyWrapper.getInternalTopologyBuilder(topology); +
topologyBuilder.connectProcessorAndStateStores(table1.name, getterSupplier1.storeNames()); + + try (final TopologyTestDriverWrapper
driver = new TopologyTestDriverWrapper(builder.build(), props)) { + final KTableValueGetter<String, String> getter1 =
getterSupplier1.get(); + getter1.init(driver.setCurrentNodeForProcessorContext(table1.name)); - driver.process(topic1, "A", "01"); -
driver.process(topic1, "B", "01"); - driver.process(topic1, "C", "01"); + driver.pipeInput(recordFactory.create(topic1, "A", "01")); +
driver.pipeInput(recordFactory.create(topic1, "B", "01")); + driver.pipeInput(recordFactory.create(topic1, "C", "01")); - assertEquals("01",
getter1.get("A")); - assertEquals("01", getter1.get("B")); - assertEquals("01", getter1.get("C")); + assertEquals("01", getter1.get("A")); +
assertEquals("01", getter1.get("B")); + assertEquals("01", getter1.get("C")); - driver.process(topic1, "A", "02"); - driver.process(topic1,
"B", "02"); + driver.pipeInput(recordFactory.create(topic1, "A", "02")); + driver.pipeInput(recordFactory.create(topic1, "B", "02")); -
assertEquals("02", getter1.get("A")); - assertEquals("02", getter1.get("B")); - assertEquals("01", getter1.get("C")); + assertEquals("02",
getter1.get("A")); + assertEquals("02", getter1.get("B")); + assertEquals("01", getter1.get("C")); - driver.process(topic1, "A", "03"); +
driver.pipeInput(recordFactory.create(topic1, "A", "03")); - assertEquals("03", getter1.get("A")); - assertEquals("02", getter1.get("B")); -
assertEquals("01", getter1.get("C")); + assertEquals("03", getter1.get("A")); + assertEquals("02", getter1.get("B")); + assertEquals("01",
getter1.get("C")); - driver.process(topic1, "A", null); - driver.process(topic1, "B", null); + driver.pipeInput(recordFactory.create(topic1,
"A", (String) null)); + driver.pipeInput(recordFactory.create(topic1, "B", (String) null)); - assertNull(getter1.get("A")); -
assertNull(getter1.get("B")); - assertEquals("01", getter1.get("C")); + assertNull(getter1.get("A")); + assertNull(getter1.get("B")); +
assertEquals("01", getter1.get("C")); + } } @@ -148,35 +150,32 @@ public void testNotSendingOldValue() { final
MockProcessorSupplier<String, Integer> supplier = new MockProcessorSupplier<>(); - builder.build().addProcessor("proc1", supplier,
table1.name); + final Topology topology = builder.build().addProcessor("proc1", supplier, table1.name); - driver.setUp(builder, stateDir);
+ try (final TopologyTestDriver driver = new TopologyTestDriver(topology, props)) { - final MockProcessor<String, Integer> proc1 =
supplier.theCapturedProcessor(); + final MockProcessor<String, Integer> proc1 = supplier.theCapturedProcessor(); -
driver.process(topic1, "A", "01"); - driver.process(topic1, "B", "01"); - driver.process(topic1, "C", "01"); - driver.flushState(); +
driver.pipeInput(recordFactory.create(topic1, "A", "01")); + driver.pipeInput(recordFactory.create(topic1, "B", "01")); +
driver.pipeInput(recordFactory.create(topic1, "C", "01")); - proc1.checkAndClearProcessResult("A:(01<-null)", "B:(01<-null)", "C:(01<-
null)"); + proc1.checkAndClearProcessResult("A:(01<-null)", "B:(01<-null)", "C:(01<-null)"); - driver.process(topic1, "A", "02"); -
driver.process(topic1, "B", "02"); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", "02")); +
driver.pipeInput(recordFactory.create(topic1, "B", "02")); - proc1.checkAndClearProcessResult("A:(02<-null)", "B:(02<-null)"); +
proc1.checkAndClearProcessResult("A:(02<-null)", "B:(02<-null)"); - driver.process(topic1, "A", "03"); - driver.flushState(); +
driver.pipeInput(recordFactory.create(topic1, "A", "03")); - proc1.checkAndClearProcessResult("A:(03<-null)"); +
proc1.checkAndClearProcessResult("A:(03<-null)"); - driver.process(topic1, "A", null); - driver.process(topic1, "B", null); -
driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", (String) null)); + driver.pipeInput(recordFactory.create(topic1,
"B", (String) null)); - proc1.checkAndClearProcessResult("A:(null<-null)", "B:(null<-null)"); + proc1.checkAndClearProcessResult("A:
(null<-null)", "B:(null<-null)"); + } } @Test @@ -193,34 +192,31 @@ public void testSendingOldValue() { final
MockProcessorSupplier<String, Integer> supplier = new MockProcessorSupplier<>(); - builder.build().addProcessor("proc1", supplier,
table1.name); + final Topology topology = builder.build().addProcessor("proc1", supplier, table1.name); - driver.setUp(builder, stateDir);
+ try (final TopologyTestDriver driver = new TopologyTestDriver(topology, props)) { - final MockProcessor<String, Integer> proc1 =
supplier.theCapturedProcessor(); + final MockProcessor<String, Integer> proc1 = supplier.theCapturedProcessor(); -
driver.process(topic1, "A", "01"); - driver.process(topic1, "B", "01"); - driver.process(topic1, "C", "01"); - driver.flushState(); +
driver.pipeInput(recordFactory.create(topic1, "A", "01")); + driver.pipeInput(recordFactory.create(topic1, "B", "01")); +
driver.pipeInput(recordFactory.create(topic1, "C", "01")); - proc1.checkAndClearProcessResult("A:(01<-null)", "B:(01<-null)", "C:(01<-
null)"); + proc1.checkAndClearProcessResult("A:(01<-null)", "B:(01<-null)", "C:(01<-null)"); - driver.process(topic1, "A", "02"); -
driver.process(topic1, "B", "02"); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", "02")); +
driver.pipeInput(recordFactory.create(topic1, "B", "02")); - proc1.checkAndClearProcessResult("A:(02<-01)", "B:(02<-01)"); +
```

proc1.checkAndClearProcessResult("A:(02<-01)", "B:(02<-01)"); - driver.process(topic1, "A", "03"); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", "03")); - proc1.checkAndClearProcessResult("A:(03<-02)"); + proc1.checkAndClearProcessResult("A:(03<-02)"); - driver.process(topic1, "A", null); - driver.process(topic1, "B", null); - driver.flushState(); + driver.pipeInput(recordFactory.create(topic1, "A", (String) null)); + driver.pipeInput(recordFactory.create(topic1, "B", (String) null)); - proc1.checkAndClearProcessResult("A:(null<-03)", "B:(null<-02)"); + proc1.checkAndClearProcessResult("A:(null<-03)", "B:(null<-02)"); + } } } diff --git a/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java b/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java index e46ec6a35d0..83471ce61c0 100644 --- a/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java +++ b/streams/test-utils/src/main/java/org/apache/kafka/streams/TopologyTestDriver.java @@ -175,13 +175,14 @@ private final static int PARTITION_ID = 0; private final static TaskId TASK_ID = new TaskId(0, PARTITION_ID); - private final StreamTask task; + final StreamTask task; private final GlobalStateUpdateTask globalStateTask; private final GlobalStateManager globalStateManager; private final StateDirectory stateDirectory; private final Metrics metrics; - private final ProcessorTopology processorTopology; + final ProcessorTopology processorTopology; + final ProcessorTopology globalTopology; private final MockProducer<byte[], byte[]> producer; @@ -220,18 +221,6 @@ public TopologyTestDriver(final Topology topology, this(topology.internalTopologyBuilder, config, initialWallClockTimeMs); } - /** - * Create a new test diver instance. - * - * @param builder builder for the topology to be tested - * @param config the configuration for the topology - */ - TopologyTestDriver(final InternalTopologyBuilder builder, - final Properties config) { - this(builder, config, System.currentTimeMillis()); - - } - /** * Create a new test diver instance. * @@ -249,7 +238,7 @@ private TopologyTestDriver(final InternalTopologyBuilder builder, internalTopologyBuilder.setApplicationId(streamsConfig.getString(StreamsConfig.APPLICATION_ID_CONFIG)); processorTopology = internalTopologyBuilder.build(null); - final ProcessorTopology globalTopology = internalTopologyBuilder.buildGlobalStateTopology(); + globalTopology = internalTopologyBuilder.buildGlobalStateTopology(); final Serializer<byte[]> bytesSerializer = new ByteArraySerializer(); producer = new MockProducer<byte[], byte[]>(true, bytesSerializer, bytesSerializer) { -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

28. h314to opened a new pull request #5433: KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [part 4] URL: https://github.com/apache/kafka/pull/5433 This PR continues the work towards the removal of KStreamTest driver. As suggested in a [previous PR](https://github.com/apache/kafka/pull/4986), `OutputVerifier` is used instead of `MockProcessor`. Refactor: * KTableKTableOuterJoinTest * KTableKTableLeftJoinTest * KTableKTableOuterJoinTest -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
29. [~h314to] We are thinking about resolving this ticket and track the remaining work via https://issues.apache.org/jira/browse/KAFKA-7850 â€“ thoughts (please see the comments on the other ticket)?
30. Hi [~mjsax]. Thanks for pinging me. Sorry, I haven't been able to contribute lately. I have an open PR which removes a few more instances of KStreamTest driver. I also have some work done on the remaining classes which use KStreamTestDriver (only two remain after that PR, if I'm not mistaken). To avoid duplication of efforts, if you will give me just 1 or 2 days more I can fix and submit my work. I think [https://github.com/apache/kafka/pull/5433] should be easy enough to fix. I'll just rebase it, fix the conflicts, and make the changes you suggested (I'm doing that now). If you can't spare the additional couple of days and this is blocking you in any way I'm ok with you proceeding as discussed in the other ticket.
31. Thanks [~h314to]. SGTM. \cc [~guozhang] and [~Yohan123]
32. mjsax commented on pull request #5433: KAFKA-6474: Rewrite tests to use new public TopologyTestDriver [part 4] URL: https://github.com/apache/kafka/pull/5433 -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
33. [~h314to] Just merge your latest PR. How many more do we need to resolve this completely (not urgent, but I would like to resolve this eventually). Thanks a lot for all your work!
34. Hi [~h314to] could you provide an update on the remaining of PRs you are planning on to remove the deprecated KStreamTestDriver?
35. Hey, [~h314to], I just sent https://github.com/apache/kafka/pull/6732 as well. Sorry for stepping on your toes, I was tracking down some other problem and needed to remove the class for other reasons. I sent the PR before someone reminded me about this ticket.
36. guozhangwang commented on pull request #6732: KAFKA-6474: remove KStreamTestDriver URL: https://github.com/apache/kafka/pull/6732 -- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org