KIP-479: Add StreamJoined config object to Join

Status

Current state: *Accepted*

Discussion thread: *here*

***Vote thread**: here*

JIRA: *here*

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

Motivation

In the 2.2 (KIP-372 https://cwiki.apache.org/confluence/display/KAFKA/KIP-372%3A+Naming+Repartition+Topics+for+Joins+and+Grouping) release of Kafka streams, we added the Grouped class, which gave users the ability to name a repartition topic for aggregation operations. In that KIP, we piggybacked a change also to allow users to provide a name for the repartition topics of join operations.

In KIP-307 (https://cwiki.apache.org/confluence/display/KAFKA/KIP-307%3A+Allow+to+define+custom+processor+names+with+KStreams+DSL), we decided to use the same base name users provided for the repartition topic from KIP-372 to extend to naming the join operator as well as the state stores, hence changing the name of the changelog topics. Naming the state store for a join will not expose them for IQ. But it is a recommended practice to keep the changelog topic names stable in the face of upstream changes in the topology.

One point we overlooked when discussing KIP-307 is the `KStream#join` method does not take a `Materialized` object; thus, users could never name the state store hence the changelog topic. Due to the scope of changes in KIP-307, it was broken up over several PRs, and the first two were included in the 2.3 release. Having the partially implemented KIP does not in and of itself represent a problem as only internal classes were added needed for the ability to name operators. But since KSTream#join does not take a Materialized object if users elect to name the Join processor (hence naming the repartition topics as well). When 2.4 is released, the provided name will automatically extend to the state stores for the Join and change the changelog topics as well. This sudden state store name change represents a breaking change that might catch users by surprise.

Additionally, users currently can't configure the state stores involved in a stream - stream join. In some cases, users may want to use in-memory stores or custom stores for the join. But with the API in its current state, this isn't possible. But simply adding a single Materialized object won't solve the issue fully for the users wanting to customize the stores used in a join. When performing a stream-stream join, Kafka Streams requires two state stores, one for the calling side and another for "other" stream participating in the join.

While a StoreSupplier will return a unique StateStore instance with each invocation of StoreSupplier.get(), the names need to be unique as well. Hence we need to provide users a way to 1) Name the state stores to prevent shifting of names with upstream topology changes or 2) fully customize the join by delivering their StoreSupplier instances. While we could add another Materialized parameter, doing so is starting to go against the grain of using configuration objects in the first place; reducing the number of overloads. Also, for users wishing to name the stores, we don't need 2 Materialized instances, just the base name will suffice.

Public Interfaces

This KIP will add the following methods to the KStream interface.

Methods Added to KStream

```
<VO, VR> KStream<K, VR> join(final KStream<K, VO> otherStream,
                             final ValueJoiner<? super V, ? super VO, ? extends VR> joiner,
                             final JoinWindows windows,
                             final StreamJoined<K, V1, V2> streamJoined);

<VO, VR> KStream<K, VR> leftJoin(final KStream<K, VO> otherStream,
                                 final ValueJoiner<? super V, ? super VO, ? extends VR> joiner,
                                 final JoinWindows windows,
                                 final StreamJoined<K, V1, V2> streamJoined);

<VO, VR> KStream<K, VR> outerJoin(final KStream<K, VO> otherStream,
                                  final ValueJoiner<? super V, ? super VO, ? extends VR> joiner,
                                  final JoinWindows windows,
                                  final StreamJoined<K, V1, V2> streamJoined);
```

This KIP will also add a new configuration object StreamJoined.

StreamJoined

```
public class StreamJoined<K, V1, V2> implements NamedOperation<StreamJoined<K, V1, V2>> {

public static <K, V1, V2> StreamJoined<K, V1, V2> with(final WindowBytesStoreSupplier storeSupplier,
                                                       final WindowBytesStoreSupplier otherSupplier){}

public static <K, V1, V2> StreamJoined<K, V1, V2> as(final String storeName) {}

public static <K, V1, V2> StreamJoined<K, V1, V2> with(final Serde<K> keySerde,
                                                       final Serde<V1> valueSerde,
                                                       final Serde<V2> otherValueSerde) {}
 // The withName method will name the process and provide the base name
 // for any repartition topics if required

 public StreamJoined<K, V1, V2> withName(final String name) {}

 // The withStoreName is used as the base name for stores provided by Kafka Streams
 // If users provide state store suppliers, then the name in the store supplier is used
 public StreamJoined<K, V1, V2> withStoreName(final String storeName) {}

 public StreamJoined<K, V1, V2> withKeySerde(final Serde<K> keySerde) {}

 public StreamJoined<K, V1, V2> withValueSerde(final Serde<V1> valueSerde) {}

 public StreamJoined<K, V1, V2> withOtherValueSerde(final Serde<V2> otherValueSerde) {}

 public StreamJoined<K, V1, V2> withThisStoreSupplier(final WindowBytesStoreSupplier storeSupplier) {}

 public StreamJoined<K, V1, V2> withOtherStoreSupplier(final WindowBytesStoreSupplier otherStoreSupplier) {}
```

Proposed Changes

With this in mind, this KIP aims to add a new configuration object `StreamJoined`. The `StreamJoined` configuration allows users to specify Serdes for the join, naming of join processor, the base name of the default state stores, and provide store suppliers. Essentially a merging of Joined and Materialized configuration objects. We'll add an overloaded `KStream#join` method accepting a `StreamJoined` parameter without a `Joined` parameter. The overloads will apply to all flavors of `KStream#join` (join, left, and outer). Due to the significant overlap of the new `StreamJoined` configuration, we'll also deprecate the `KStream` join methods taking the `Joined` parameter.

This will allow for users who wish to upgrade to 2.4 without having to make a breaking change. Now the `Joined#as` method will only name the Join processor and repartition topics and no longer name the associated state stores and changelog topics. So if users wish to upgrade to 2.4 and don't provide a name for the states stores, it will continue to use the auto-generated name in 2.3. Of course, if users elect to name the state stores, they can do so via `StreamJoined#as("store name"). Naming the store` is recommended to pin the changelog topic names in case of any upstream topology changes.

We should note that providing store suppliers to the join ***will not enable*** interactive queries over the join state stores. We'll update the documentation stating as much.

Semantics of StreamJoined

With the ability to provide so many configuration items, we should probably discuss the semantics of the new configuration.

Serdes

- If users do not provide serdes then Kafka Streams uses the default key and value serdes specified in the configs .
- KafkaStreams will use the serdes to configure both stores regardless if the stores are the default or come from user provided `StoreSupplier` instances.

Naming

- The `StreamJoined#withName` method will name ***only*** the join processor and the base name of any required repartition topics. Not using the `StreamJoined#withName` will result in Kafka Streams auto-generating names for the join processor and any repartition topics.
- The `StreamJoined#withStoreName` will provide ***only*** the base name for the state stores of the join. If users pass in `StoreSupplier`(s), then the name of the StoreSupplier is used. The StoreSupplier names always take priority over names coming from `StreamJoined#withStoreName`.

Store Suppliers

- Users can provide 0, 1 or 2 `StoreSupplier` instances.
    - The provided `StoreSupplier` instances must implement `WindowBytesStoreSupplier.`
    - In the case of providing **zero** StoreSupplier(s), KafkaStreams will create two persistent state stores as it does now. The names for the stores will be auto-generated unless the user gives a name with the `StreamJoined#withName` method.
    - In the case of providing **one** StoreSupplier, KafkaStreams will create one persistent state store, and the `StoreSupplier` will create one state store. The name for the default store will be auto-generated unless the user gives a name with the `StreamJoined#withName` method.
    - When providing **two** StoreSuppliers, the state stores for the join will come from the suppliers, and the names will come from the names given to the suppliers.
- There will be a runtime check to ensure that the values of JoinWindows match those of the provided StoreSuppliers.
- The name of the StoreSupplier will take priority over the name given via `StreamJoined#withStoreName.`

Compatibility, Deprecation, and Migration Plan

Since the changes are additions in a strict sense, there are no compatibility issues with existing code.

- For users that have not named the Join repartition node, then there are no migration concerns.
- For users that previously have named the Join repartition node and want to upgrade to 2.4, a rolling restart should be possible since the store name will not change.
- Users electing to add a name to the state store will need to stop all instances and redeploy the new code. Then reset the application before starting to ensure no data is unprocessed.
- We'll deprecate the Stream-Stream join methods using Joined object. These items will be removed in a later major release.

Rejected Alternatives

- Adding a configuration to indicate if the Joined.named should be used for the state store name. Adding a configuration was ruled out as it's not a best practice to add configs for this type of change.
- Adding a method to the Joined config item as it seemed to clutter the API.