KIP-819: Merge multiple KStreams in one operation

Status

Current state: Under Discussion

Discussion thread: here

JIRA: here

Motivation

The `KStream` API provides merge(`KStream`) to merge another `KStream` with this. Sometimes, it may be useful to merge more than 2 `KStreams` together. Currently, the best way to do this is using Java's `Stream.reduce`:

```
List<KStream<K, V>> streams ...;
streams.stream().reduce((left, right) -> left.merge(right));
```

This creates a merge node in the process graph for *every* `KStream` in the collection being merged.

Complex process graphs can make understanding an application and debugging more difficult.

Public Interfaces

Two new methods will be added to `KStream`:

```
KStream<K, V> merge(Collection<KStream<K, V>> streams);
KStream<K, V> merge(Collection<K, V> streams, Named named);
```

Two static methods will also be added to KStream:

```
static <K, V> KStream<K, V> merged(Collection<KStream<K, V>> streams);
static <K, V> KStream<K, V> merged(Collection<KStream<K, V>> streams, Named named);
```

These are utility methods for merging a collection of KStreams when the user doesn't already have a KStream to invoke `merge` upon.

Proposed Changes

`KStream` will be updated with the above API, and `KStreamImpl` will have its implementation updated to match.

As with the existing implementation, if any of the input `KStreams` need to be repartitioned, the entire merged `KStream` will be repartitioned. If users only wish to repartition the sub-pairs of `KStreams` that need to be, they can fall-back to the previous strategy of iteratively merging pairs of `KStream` down using `Stream#reduce`

The `KStream` that merge is called on is treated no differently in the merge than the `KStreams` in the `Collection` being merged. This is the same behaviour we have today, where the order of the `KStreams` being merged has no effect.

Compatibility, Deprecation, and Migration Plan

- Due to ambiguity in the type, passing the `null` literal as the first parameter to `merge` will no longer be possible without explicitly casting to either `KStream<K, V>` or `Collection<KStream<K, V>>` first. Since passing `null` for this argument will always produce a runtime error anyway, passing a `null` literal was never intended in the original API.
- No other modifications, deprecations or removals are being made to the existing API.