KIP-120: Cleanup Kafka Streams builder API

Status

Current state: *Accepted* [VOTE] KIP-120: Cleanup Kafka Streams builder API

Discussion thread: [DISCUSS] KIP-120: Cleanup Kafka Streams builder API

JIRA: *KAFKA-3856*

Released: 1.0.0

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

Motivation

Currently, Kafka Streams public API leaks a bunch of internal methods that should not be public. Furthermore, DSL and PAPI abstraction are not completely separated at the moment and `TopologyBuilder` offers methods that belong to DSL only.

Public Interfaces

In order to get a clean refactoring, we will ***deprecate classes***

- `org.apache.kafka.streams.processor.TopologyBuilder`
- `org.apache.kafka.streams.kstream.KStreamBuilder`

and ***add*** them again with new name and ***different*** package

- `org.apache.kafka.streams.Topology`
- `org.apache.kafka.streams.StreamsBuilder`

Furthermore, we add a new interface to get a full description the a topology (to compensate for some internal methods that get removed)

- `org.apache.kafka.streams.TopologyDescription`
    - `TopologyDescription` will have public interfaces:
        - `Subtopology`
        - `GlobalStores`
        - `Node`
        - `Source`
        - `Sink`
        - `Processor`

The following methods will not be available in the newly added classes:

- `TopologyBuilder -> Topology`: setApplicationId, ***connectSourceStoreAndTopic***, ***connectProcessors***, ***addInternalTopic***, ***copartitionSources***, nodeGroups, build, buildGlobalStateTopology, globalStateStores, topicGroups, earliestResetTopicPattern, latestResetTopicPattern, ***stateStoreNamesToSourceTopics***, copartitioneGroups, sourceTopicPattern, updateSubscriptions
- `KStreamBuilder -> StreamsBuilder`: all methods from TopologyBuilder (as KStreamBuilder does not inherit from TopologyBuilder anymore) except addStateStore and addGlobalStore (which are both added explicitly to `StreamsBuilder`) plus newName
- methods highlighted in `TopologyBuilder` list, are methods that actually belong to DSL abstraction

New `org.apache.kafka.streams.Topology` class:

```
public class Topology {

    // existing public API from current TopologyBuilder class that will not be changed

    public enum AutoOffsetReset {
        EARLIEST , LATEST
    }

    public Topology();


    public synchronized Topology addSource(String name, String... topics);
    public synchronized Topology addSource(String name, Pattern topicPattern);

    public synchronized Topology addSource(AutoOffsetReset offsetReset, String name, String... topics);
    public synchronized Topology addSource(AutoOffsetReset offsetReset, String name, Pattern topicPattern);

    public synchronized Topology addSource(String name, Deserializer keyDeserializer, Deserializer
valueDeserializer, String... topics);
    public synchronized Topology addSource(String name, Deserializer keyDeserializer, Deserializer
valueDeserializer, Pattern topicPattern);

    public synchronized Topology addSource(AutoOffsetReset offsetReset, String name, Deserializer
keyDeserializer, Deserializer valueDeserializer, String... topics);
    public synchronized Topology addSource(AutoOffsetReset offsetReset, String name, Deserializer
keyDeserializer, Deserializer valueDeserializer, Pattern topicPattern);


    public synchronized Topology addSink(String name, String topic, String... parentNames);
    public synchronized Topology addSink(String name, String topic, Serializer keySerializer, Serializer
valueSerializer, String... parentNames);

    public synchronized Topology addSink(String name, String topic, StreamPartitioner partitioner, String...
parentNames);
    public synchronized <K, V> Topology addSink(String name, String topic, Serializer<K> keySerializer,
Serializer<V> valueSerializer, StreamPartitioner<? super K, ? super V> partitioner, String... parentNames);


    public synchronized Topology addProcessor(String name, ProcessorSupplier supplier, String... parentNames);

    public synchronized Topology addStateStore(StateStoreSupplier supplier, String... processorNames);

    public synchronized Topology addGlobalStore(StateStore store,
                                                String sourceName,
                                                Deserializer keyDeserializer,
                                                Deserializer valueDeserializer,
                                                String topic,
                                                String processorName,
                                                ProcessorSupplier stateUpdateSupplier);

    public synchronized Topology connectProcessorAndStateStores(String processorName, String...
stateStoreNames);



    // newly added method to reveal topology structure

    // describes the current Topology, ie, TopologyDescription will not be updated if Topology is modified but
#describe() must be called again
    public synchronized TopologyDescription describe();
}
```

New `org.apache.kafka.streams.StreamsBuilder` class:

```
public class StreamsBuilder {

    // existing public API from current KStreamBuilder class that will not be changed
    // small change: adding `synchronized` to all methods to align with TopologyBuilder

    public StreamsBuilder();


    public synchronized <K, V> KStream<K, V> stream(String... topics);
    public synchronized <K, V> KStream<K, V> stream(Pattern topicPattern);

    public synchronized <K, V> KStream<K, V> stream(AutoOffsetReset offsetReset, String... topics);
    public synchronized <K, V> KStream<K, V> stream(AutoOffsetReset offsetReset, Pattern topicPattern);

    public synchronized <K, V> KStream<K, V> stream(Serde<K> keySerde, Serde<V> valueSerde, String... topics);
    public synchronized <K, V> KStream<K, V> stream(Serde<K> keySerde, Serde<V> valueSerde, Pattern
topicPattern);

    public synchronized <K, V> KStream<K, V> stream(AutoOffsetReset offsetReset, Serde<K> keySerde, Serde<V>
valueSerde, String... topics);
    public synchronized <K, V> KStream<K, V> stream(AutoOffsetReset offsetReset, Serde<K> keySerde, Serde<V>
valueSerde, Pattern topicPattern);


    public synchronized <K, V> KTable<K, V> table(String topic, String storeName);
    public synchronized <K, V> KTable<K, V> table(AutoOffsetReset offsetReset, String topic, String storeName);
    public synchronized <K, V> KTable<K, V> table(Serde<K> keySerde, Serde<V> valueSerde, String topic, String
storeName);
    public synchronized <K, V> KTable<K, V> table(AutoOffsetReset offsetReset, Serde<K> keySerde, Serde<V>
valueSerde, String topic, String storeName);


    public synchronized <K, V> GlobalKTable<K, V> globalTable String topic, String storeName);
    public synchronized <K, V> GlobalKTable<K, V> globalTable Serde<K> keySerde, Serde<V> valueSerde, String
topic, String storeName);

    public synchronized <K, V> KStream<K, V> merge KStream<K, V>... streams);



    // newly added method

    public synchronized KStreamBuilder addStateStore(StateStoreSupplier supplier, String... processorNames);
    public synchronized KStreamBuilder addGlobalStore(StateStore store,
                                                      String sourceName,
                                                      Deserializer keyDeserializer,
                                                      Deserializer valueDeserializer,
                                                      String topic,
                                                      String processorName,
                                                      ProcessorSupplier stateUpdateSupplier);

    public synchronized Topology build();
}
```

New `org.apache.kafka.streams.TopologyDescription` class:

```
public interface TopologyDescription {
    Set<Subtopology> subtopologies();
    Set<GlobalStore> globalStores();

    inferface Subtopology {
        int id();
        Set<Node> nodes();
    }

    interface GlobalStore {
        Source source();
        Processor processor();
    }

    interface Node {
        String name()
        Set<Node> predecessors();
        Set<Node> successors();
    }

    interface Source extends Node {
        String topics(); // can be comma separated list of topic names or pattern (as String)
    }

    interface Processor extends Node {
        Set<String> stores();
    }

    interface Sink extends Node {
        String topic();
    }

}
```

Proposed Changes

We will add two new internal classes

- `org.apache.kafka.streams.processor.internals.InternalTopologyBuilder`
- `org.apache.kafka.streams.kstream.internals.InternalStreamsBuilder`

that offer the methods remove from current API. Thus, both classes are the actual implementation. Old `TopologyBuilder` and `KStreamBuilder` are only proxy classes to both classes respectively, for backward compatibility.

The newly added `Topology` uses `InternalTopologyBuilder` as member.

The newly added `StreamsBuilder` uses the new `Topology` as a member (no class hierarchy anymore – using it as member gives a clear separation between PAPI and DSL).

Because the new `StreamsBuilder` does not inherit from new `Topology` we need to add `StreamsBuilder#build()` that returns the actual `Topology` to be passed into `KafkaStreams` client.

Note: because of backward compatibility, removed DSL specific classes offered by old `TopologyBuilder` must be offered by `InternalTopologyBuilder` for now. However, after both deprecated classes got removed, this cleanup can be done (and does not require a KIP anymore, because it's internal refactoring -- we just need to create a JIRA for this). Thus, this KIP falls short of separating PAPI and DSL completely. But it's a necessary first step to do the separation in a backward compatible way (backward compatibility requires a two step approach).

Compatibility, Deprecation, and Migration Plan

- Because no classes/method will be removed but only deprecated, this change will be fully backward compatible
- We intend to remove all deprecated classes/methods in 0.11.1, but we can keep them longer on user request

Test Plan

Tests need to be rewritten but no new tests are required. All tests for public API need to be updated to use new `Topology` and `StreamsBuilder`. All tests using internal API, need to be rewritten to use `InternalTopologyBuilder` and `InternalStreamsBuilder`.

Rejected Alternatives

None.