KIP-123: Allow per stream/table timestamp extractor

Status

Current state: *"accepted"*

Discussion thread: HERE

JIRA: *KAFKA-4144*

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

Motivation

At the moment the timestamp extractor is configured via a `StreamConfig` value to `KafkaStreams`. That means you can only have a single timestamp extractor per app, even though you may be joining multiple `streams/tables` that require different timestamp extraction methods.

Ideally the user should be able to specify a timestamp extractor via `KStreamBuilder.stream/table`, just like you can specify key and value serdes that override the `StreamConfig` defaults.

Public Changes

We propose to add the following overloaded methods to KStreamBuilder.java and TopologyBuilder.java:

```
KStreamBuilder.stream(final TimestampExtractor timestampExtractor, final Serde<K> keySerde, final Serde<V>
valSerde, final String... topics)

KStreamBuilder.stream(final AutoOffsetReset offsetReset, final TimestampExtractor timestampExtractor, final
Serde<K> keySerde, final Serde<V> valSerde, final String... topics)

KStreamBuilder.stream(final TimestampExtractor timestampExtractor, final Serde<K> keySerde, final Serde<V>
valSerde, final Pattern topicPattern)

KStreamBuilder.stream(final AutoOffsetReset offsetReset, final TimestampExtractor timestampExtractor, final
Serde<K> keySerde, final Serde<V> valSerde, final Pattern topicPattern)

KStreamBuilder.table(final TimestampExtractor timestampExtractor, final String topic, final String
storeName)

KStreamBuilder.table(final AutoOffsetReset offsetReset, final TimestampExtractor timestampExtractor, final
String topic, final String storeName)

KStreamBuilder.table(final TimestampExtractor timestampExtractor, final Serde<K> keySerde, final Serde<V>
valSerde, final String topic, final String storeName)

KStreamBuilder.table(final AutoOffsetReset offsetReset, final TimestampExtractor timestampExtractor, final
Serde<K> keySerde, final Serde<V> valSerde, final String topic, final String storeName)

KStreamBuilder.globalTable(final TimestampExtractor timestampExtractor, final Serde<K> keySerde, final
Serde<V> valSerde, final String topic, final String storeName)

KStreamBuilder.globalTable(final Serde<K> keySerde, final Serde<V> valSerde, final String topic, final
String storeName)


TopologyBuilder.addSource(final TimestampExtractor timestampExtractor, final String name, final String...
topics)
```

```
TopologyBuilder.addSource(final AutoOffsetReset offsetReset, final TimestampExtractor timestampExtractor,
final String name, final String... topics)

TopologyBuilder.addSource(final TimestampExtractor timestampExtractor, final String name, final Pattern
topicPattern)

TopologyBuilder.addSource(final AutoOffsetReset offsetReset, final TimestampExtractor timestampExtractor,
final String name, final Pattern topicPattern)

TopologyBuilder.addSource(final AutoOffsetReset offsetReset, final String name, final TimestampExtractor
timestampExtractor, final Deserializer keyDeserializer, final Deserializer valDeserializer, final String...
topics)

TopologyBuilder.addGlobalStore(final StateStore store, final String sourceName, final TimestampExtractor
timestampExtractor, final Deserializer keyDeserializer, final Deserializer valueDeserializer, final String
topic, final String processorName, final ProcessorSupplier stateUpdateSupplier)

TopologyBuilder.addSource(final AutoOffsetReset offsetReset, final String name, final TimestampExtractor
timestampExtractor, final Deserializer keyDeserializer, final Deserializer valDeserializer, final Pattern
topicPattern)
```

Moreover, we propose to make the following changes in StreamConfig class:

Deprecate:

```
StreamConfig.TIMESTAMP_EXTRACTOR_CLASS_CONFIG

StreamConfig.TIMESTAMP_EXTRACTOR_CLASS_DOC

StreamConfig.KEY_SERDE_CLASS_CONFIG

StreamConfig.KEY_SERDE_CLASS_DOC

StreamConfig.VALUE_SERDE_CLASS_CONFIG

StreamConfig.VALUE_SERDE_CLASS_DOC

StreamConfig.keySerde()

StreamConfig.valueSerde()
```

Add:

```
StreamConfig.DEFAULT_TIMESTAMP_EXTRACTOR_CLASS_CONFIG

StreamConfig.DEFAULT_TIMESTAMP_EXTRACTOR_CLASS_DOC

StreamConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG

StreamConfig.DEFAULT_KEY_SERDE_CLASS_DOC

StreamConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG

StreamConfig.DEFAULT_VALUE_SERDE_CLASS_DOC

StreamConfig.defaultKeySerde()

StreamConfig.defaultValueSerde()
```

`All the changes to StreamConfig class are backward compatible.`

Proposed Changes

- First, we add `TimestampExtractor` property to `SourceNode` class and `TopologyBuilder.SourceNodeFactory` inner class.
- Second, in `StreamTask` class rather than passing `TimestampExtractor` defined in `StreamsConfig`, we pass each each `SourceNode`'s timestamp. Here there is a small issue we should consider. If the source is defined with `Pattern`, then getting source with exact topic name returns null in `StreamTask` class. For example, we define the source by `Pattern` "t.*" and topic name is "topic". In this case, `topology.source(partition.topic())` will return null. Because in the topology we have the source name as "Pattern [t.*]" for particular `SourceNode`. To overcome this issue firstly, we change the SourceNode name to be exactly the `Pattern.string` when defined with Pattern. For example, if the source is defined with "t.*", its name will be exactly "t.*" (up to now it was "Pattern [t.*]"). Secondly, in `StreamTask` we change how we

discover the `SourceNodes` for particular `partition.topic()`. If there exist some `SourceNode` in topology with `topology.source(partition.topic())`, then we immediately return the result. Otherwise, we search all `SourceNodes` in topology to check if their Pattern matches with `partition.topic()`.

The PR can be found here.

Test Plan

The unit tests are added to `TopologyBuilderTest` and `KStreamBuilderTest` classes.

Rejected Alternatives

None yet.