KIP-886: Add Client Producer and Consumer Builders

Status

Current state: *Under Discussion*

Discussion thread: *here [Change the link from the KIP proposal email archive to your own email thread]*

JIRA: Unable to render Jira issues macro, execution error.

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

Motivation

Creating a producer/consumer involves looking up the consts for kafka configuration, and usually doing a lot of googling for what the possible values are for each configuration. Providing a builder where there are named methods to set various configuration, with meaningful parameters (ie: enums for configurations that can only have a few values), and good javadoc would make it much easier from developers working from ides (such as intellij, eclipse, netbeans, etc) to discover the various configurations, and navigate to the documentation for the various options when creating producers/consumers. This is potentially a duplicate/overlapping in scope with KIP-839: Provide builders for KafkaProducer/KafkaConsumer and KafkaStreams (apologies, hadn't seen it before making this one).

Public Interfaces

A builder will be added to the kafka producer and kafka consumer (separate classes than the existing ones). Enums will be added where relevant. Affected components

- org/apache/kafka/clients/producer
- org/apache/kafka/clients/consumer (eventually, once stable)

Proposed Changes

ConsumerBuilder and ProducerBuilder will be added. They will basically take the bootstrap url, have a bunch of .with<Config>(<config params>) methods, and finally a .build() method. This provides a more user friendly alternative to looking up the consts, adding them to a properties object, and then creating the producer consumer, the de-facto standard today. In certain cases, the build() method may do additional things (ie: for a transactional producer, initTransactions() would be called), in order to ensure that the producer/consumer is ready to begin work.

Examples:

Pseudocode:

Producer Example:

```
ProducerBuilder(String bootstrapUrl)
{ ... }
ProducerBuilder withTransactionsEnabled(String transactionalId){ ... // internally sets a flag so that
transactions are initialized when producer is built }
Producer build(){ ... }
```

Consumer Example:

```
/**
*
* @param strategy : if the strategy is set to NONE, then an error is thrown if a valid offset is passed to seek
(etc.), if it is set to EARLIEST, then the consumer will seek to beginning (etc).
*/
ConsumerBuilder withAutoOffsetResetStrategy(OffsetResetStrategy strategy){ ... }
```

---

where OffsetResetStrategy is an enum with 3 values

```
public enum OffsetResetStrategy {EARLIEST,LATEST, NONE}
```

Compatibility, Deprecation, and Migration Plan

Fully backward compatible, wraps existing mechanism to build producer/consumer, just a lot more user friendly

Test Plan

As this is a new feature, no breakages are expected. Unit tests and a few integration tests should be sufficient to ensure it works. Additionally, existing tests that initialize a producer/consumer can be migrated to the builder in order to exercise the functionality more.

Rejected Alternatives

The major alternative is to add nothing, and leave things the way they are. This means there are fewer changes, but it will continue being really annoying to have to look at all the producer and consumer consts, and then have to google what they mean, and what the possible values are for the config.