KIP-209 - Connection String Support

Status

Current state: *Under Discussions*

Discussion thread: KIP-209 Connection String Support

JIRA: a JIRA will be created once there is consensus about this KIP

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

Motivation

Users currently need to type in a HashMap for connection properties on Producers, Consumers and AdminClient.

The API could be simplified here by allowing a single line with string properties, what would be:

- *simpler to use, easier for newer users*
- *Single string object for configuration*
    - *This would allow users to store their configurations in single places, in a better fashion. Example property files with a single object string*

Example of current code:

```
Map<String, Object> config = new HashMap<>();
config.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9999");
config.put(ConsumerConfig.RECEIVE_BUFFER_CONFIG, -2);
KafkaConsumer consumer = new KafkaConsumer<>(config, new ByteArrayDeserializer(), new ByteArrayDeserializer());
```

This following code would be the equivalent piece for the previous example.

KafkaConsumer consumer = new KafkaConsumer("localhost:9999;receiver.buffer.bytes=-2", new ByteArrayDeserializer(), new ByteArrayDeserializer());

KafkaConsumer consumer = new KafkaConsumer("(localhost:9999;localhost2:99999);parameter1=value2;parameter2=value3", new ByteArrayDeserializer(), new ByteArrayDeserializer());

This is similar to what users are used when connecting to Databases (e.g. JDBC), other message systems or many other server solutions.

Public Interfaces

No public interfaces would be added, however a new contructor would be added to KafkaConsumer, KafkaProducer and KafkaAdminClient

```
public KafkaConsumer(String connectionString,
 Deserializer<K> keyDeserializer,
 Deserializer<V> valueDeserializer) {
}
```

Proposed Changes

- As mentioned above, there would be an addition to KafkaConsumer, KafkaProducer and KafkaAdminClient
- An utility class will be added to map the string parameters to current configurations. That constructor would then delegate to the already existing code.

Format

My proposed format for the connection string would be:

(host1:port1;host2:port2);…host:portn;param-name1=param-val1;..param-nameN=param-valN

- the first parameter should be a list of connections and ports in parenthesis
- The other parameters will be separated by semi-colon (;)

Values and Escaping

- Strings are bound by regular single quotes (') or double quotes (")
  - This is because this would make the syntax easier for when writing constants.
- Parameters are separated by ;
- the first parameter should be send between parenthesis, as there's usually a list of connections
- A back slash (\) will make the next character a regular constants. You can use these valid combinations
  - \\ meaning a regular \
  - \" meaning a literal "
  - \$ meaning a literal $
  - \; is reserved to be used as a literal colon. Its usage won't be necessary if inside strings.
- $$ will also mean literal $

It should be possible to use \ to introduce either ; or " as part of the parameters

$Arguments

- $ Arguments will be translated as the System property defined.
- Example: user=$userID will be translated as the system property "userID"

Error Handling

Typos will be informed with a log.warn. This is to avoid moving back and forth to versions where the parameter is not supported. We had a discussion on the dev list and it was suggested we should just ignore invalid parameters to avoid version compatibility issues.

For invalid encoding (things like string open), the implementation should throw an InvalidArgumentException informing the syntax error.

Compatibility, Deprecation, and Migration Plan

- *There wouldn't be any compatibility issues. The new constructor would be selected by simple method override*
- *Previous client implementations will continue to work without any issues.*

Rejected Alternatives

Create a Factory Method for KafkaConsumer or KafkaProducer providing the Connection String:

KafkaFactory:

buildConsumer(String connectionString, Serializer, Deserializer);

buildProducer(String connectionString, Serializer, Deserializer);