Item 256
git_comments:

1. * * The configuration class for KafkaConsumer
2. the consumer is fully typed, and deserialization can be too. But in case it is not provided we should default to byte[]
3. Override default max poll config if there is no value
4. * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License. *
5. These are values we enforce in sazma, and they cannot be overwritten.
6. if consumer bootstrap servers are not configured, get them from the producer configs
7. Always use default partition assignment strategy. Do not allow override.
8. Disable consumer auto-commit because Samza controls commits
9. group id should be unique per job
10. return default
11. accept kafka values directly
12. client id should be unique per job
13. * * Helper method to create configs for use in Kafka consumer. * The values are based on the "consumer" subset of the configs provided by the app and Samza overrides. * * @param config config provided by the app. * @param systemName system name to get the consumer configuration for. * @param clientId client id to be used in the Kafka consumer. * @return KafkaConsumerConfig
14. * * If settings for auto.reset in samza are different from settings in Kafka (auto.offset.reset), * then need to convert them (see kafka.apache.org/documentation): * "largest" -> "latest" * "smallest" -> "earliest" * * If no setting specified we return "latest" (same as Kafka). * @param autoOffsetReset value from the app provided config * @return String representing the config value for "auto.offset.reset" property
15. * By default, KafkaConsumer will fetch some big number of available messages for all the partitions. * This may cause memory issues. That's why we will limit the number of messages per partition we get on EACH poll().
16. Kafka client configuration
17. Translate samza config value to kafka config value
18. move the responses into the queue
19. Since we are not polling from ALL the subscribed topics, so we need to "change" the subscription temporarily
20. * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License. *
21. High watermark is fixed to be the offset of last available message, so the lag is now at least 0, which is the same as Samza's definition. If the lag is not 0, then isAtHead is not true, and kafkaClient keeps polling.
22. consume
23. update the metrics
24. ignore
25. find current lags for for each SSP nothing to read
26. we may get InvalidOffsetException | AuthorizationException | KafkaException exceptions, but we still just rethrow, and log it up the stack.
27. this is required by the KafkaConsumer to get the metrics
28. join() may timeout in this case we should interrupt it and wait again
29. the actual polling of the messages from kafka
30. populate the MetricNames first time
31. The only way to figure out lag for the KafkaConsumer is to look at the metrics after each poll() call. One of the metrics (records-lag) shows how far behind the HighWatermark the consumer is. This method populates the lag information for each SSP into latestLags member variable.
32. End offsets are the offset of the newest message + 1 If the message we are about to consume is < end offset, we are starting with a lag.
33. creates a separate thread for getting the messages.
34. this is already vetted offset so there is no need to validate it
35. registered SSPs
36. calls the setIsAtHead for the BlockingEnvelopeMap
37. Since we need to poll only from some subset of TopicPartitions (passed as the argument), we need to pause the rest.
38. initialize lag metrics
39. This is expensive, so only do it once at the beginning. After the first poll, we can rely on metrics for lag.
40. lag between the current offset and the highwatermark
41. we need to wait until the thread starts
42. move message to the BlockingEnvelopeMap's queue

43. * * This class contains a separate thread that reads messages from kafka and puts them into the BlockingEnvelopeMap * through KafkaSystemConsumer.KafkaConsumerMessageSink object. * This class is not thread safe. There will be only one instance of this class per KafkaSystemConsumer object. * We still need some synchronization around kafkaConsumer. See pollConsumer() method for details.
44. Synchronize, in case the consumer is used in some other thread (metadata or something else)
45. lags behind the high water mark, as reported by the Kafka consumer.
46. * * Add new partition to the list of polled partitions. * Bust only be called before {@link KafkaConsumerProxy#start} is called..
47. KafkaSystemConsumer uses the failureCause to propagate the throwable to the container
48. Parse the returned records and convert them into the IncomingMessageEnvelope.
49. list of all the SSPs we poll from, with their next(most recently read + 1) offsets correspondingly.
50. * * Stop this KafkaConsumerProxy and wait for at most {@code timeoutMs}. * @param timeoutMs maximum time to wait to stop this KafkaConsumerProxy
51. derived value for the highwatermark
52. * * Create a KafkaSystemConsumer for the provided {@code systemName} * @param systemName system name for which we create the consumer * @param config application config * @param metrics metrics for this KafkaSystemConsumer * @param clock system clock
53. * * record the ssp and the offset. Do not submit it to the consumer yet. * @param systemStreamPartition ssp to register * @param offset offset to register with
54. * * Create internal kafka consumer object, which will be used in the Proxy. * @param systemName system name for which we create the consumer * @param clientId client id to use int the kafka client * @param config config * @return kafka consumer object
55. set the offset for each TopicPartition
56. * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License. *
57. keep registration data until the start - mapping between registered SSPs and topicPartitions, and their offsets
58. * * return system name for this consumer * @return system name
59. * * Set the offsets to start from. * Register the TopicPartitions with the proxy. * Start the proxy.
60. currently this feature cannot be enabled, because we do not have the size of the messages available. messages get double buffered, hence divide by 2
61. subscribe to all the registered TopicPartitions
62. This sink is used to transfer the messages from the proxy/consumer to the BlockingEnvelopeMap.
63. this value should already be the 'upcoming' value
64. * * Compare two String offsets. * Note. There is a method in KafkaSystemAdmin that does that, but that would require instantiation of systemadmin for each consumer. * @return see {@link Long#compareTo(Long)}
65. * * convert from TopicPartition to TopicAndPartition
66. This proxy contains a separate thread, which reads kafka messages (with consumer.poll()) and populates BlockingEnvelopMap's buffers.
67. start the proxy thread
68. needs to be called after all the registrations are completed
69. all recoverable execptions are handled by the client. if we get here there is nothing left to do but bail out.
70. register the older (of the two) offset in the consumer, to guarantee we do not miss any messages.
71. * * convert to TopicPartition from SystemStreamPartition
72. we are using assign (and not subscribe), so we need to specify both topic and partition
73. Create the proxy to do the actual message reading.
74. get the thresholds, and set defaults if not defined.
75. add the partition to the proxy
76. create a sink for passing the messages between the proxy and the consumer
77. initialize the subscriptions for all the registered TopicPartitions
78. check if the proxy is running
79. stop the proxy (with 1 minute timeout)
80. extract kafka client configs
81. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
82. should be ignored
83. should NOT be ignored
84. if KAFKA_CONSUMER_PROPERTY_PREFIX is set, then PRODUCER should be ignored
85. if KAFKA_CONSUMER_PROPERTY_PREFIX is not set, then PRODUCER should be used
86. test stuff that should not be overridden
87. should be no failures

88. not full neither by size nor by messages
89. * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, * software distributed under the License is distributed on an * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY * KIND, either express or implied. See the License for the * specific language governing permissions and limitations * under the License. *
90. fake size, upto the limit fake size, below the limit event with the second message still below the size limit
91. limit by number of messages 4/2 = 2 per partition limit by number of bytes - disabled should disable
92. queue for ssp0 should be full now, because we added message of size FETCH_THRESHOLD_MSGS/partitionsNum
93. queue for ssp1 should full now, because we added message of size 20 on top
94. fake size fake size
95. Pass 0 as fetchThresholdByBytes, which disables checking for limit by size
96. queue for ssp1 should be less then full now, because we added message of size (FETCH_THRESHOLD_MSGS/partitionsNum - 1)
97. not full by size, but should be full by messages
98. should be full by size, but not full by number of messages (1 of 2)
99. mock kafkaConsumer and SystemConsumer

git_commits:

1. **summary:** NewSystemConsumer for kafka system
   message: NewSystemConsumer for kafka system Remove SimpleConsumer and BrokerProxy from Samza's KafkaSystemConsumer implementation. Instead use KafkaConsumerProxy with high-level kafka consumer. Author: Boris S <boryas@apache.org> Author: Boris Shkolnik <bshkolni@linkedin.com> Reviewers: Shanthoosh Venktataraman <spvenkat@usc.edu>, Prateek Maheshwari <pmaheshwari@linkedin.com> Closes #624 from sborya/NewConsumer2

github_issues:

github_issues_comments:

github_pulls:

1. **title:** New consumer for kafka system
   body: Remove SimpleConsumer and BrokerProxy from Samza's KafkaSystemConsumer implementation. Instead use KafkaConsumerProxy with high-level kafka consumer.

github_pulls_comments:

1. @sborya : instead of deleting the "old" Kafka consumer's source-code, it would be good to re-name them under a new package if users require the ability to rollback
2. Additionally, @prateekm and @nickpan47's feedback was to document the list of supported broker versions with this release.

github_pulls_reviews:

1. any reason to use a different constructor? would be nice to keep this PR just focussed on the KafkaConsumer and revert other changes
2. Revert this change.
3. Minor: Rephrase it as "Shuttting down SamzaContainer"
4. s/Coordinator stream manager config/configuration in coordinator stream.
5. Can we move shutting down the `coordinatorStreamManager` into finally block. Would be better have the entire thing within a try - finally block
6. Make all of these as per-instance methods on the KafkaConsumerConfig class?
7. This class is the "BrokerProxy". Maybe, rename it to reflect that accurately
8. Extract MessageSink as a separate java class instead of a static inner class
9. s/r/record
10. s/sspToMetricName/perPartitionMetrics
11. Can we complete java docs to each of these individual methods(here and everywhere).
12. This seems to me like duplication of what we have internally in LinkedIn for linkedin kafka consumer client(all internal classes). 1. What is the plan forward ? Are we going to simplify this and make the kafkaConsumer pluggable here? 2. Can we make this as a part of this patch itself.
13. Can this be made immutable too? A nice thing with doing this is that we can make the lifecycle of the "proxy" mirror the lifecycle of the "SystemConsumer" instance. ie., register() -> delegates to proxy.register(tp, offset) likewise, start() -> invokes proxy.start()
14. final here since they're set during construction anyways
15. This getAdminClientId method is not used anywhere in this patch and in samza codebase. Can we please delete it?
16. Please remove this unnecessary newline.
17. Just curious, in what scenarios does the records parameter will be null. It appears to me that this check is unnecessary and can be deleted.
18. Minor: Prefer using String.format to concatenate here.

19. Why do we use the static method in `NewKafkaSystmeConsumer` to create the object and use the factory to return it. It kind of looks odd to me. Can we inline this invocation here and delete that method?
20. May be better to rename this to `HighLevelKafkaConsumer` or simply `KafkaSystemConsumer` or choose some other better name. NewKafkaConsumer can mean multiple things here(could be a upgrade of kafka consumer version or a different samza internal implementation etc).
21. Can we remove the individual fields like clientID , groupId and other kafka consumer related properties from here and `KafkaConsumerProxy` and expose individual instance methods in `KafkaConsumerConfig` (E.x: `kafkaConsumerConfig.getClientId`). Rather than the following done everywhere(in the form of static method): ``` String clientId = KafkaConsumerConfig.getProducerClientId(config) ``` Can we do this instead: ``` String clientId = kafkaConsumerConfig.getClientId() ``` Just by passing in the kafkaConsumerConfig object alone to `NewKafkaConsumer` and `KafkaConsumerProxy` we can get rid of unnecessary fields and assignments everywhere.
22. Rather than parsing the offsets here and comparing, it will be better to use the comparator from `kafkaSystemAdmin` here(KafkaSystemAdmin.offsetComparator method).
23. All of the following static methods(`toTopicAndPartition(topicParittion)`, `toTopicAndPartition(SystemStreamPartition`) here seems to have similar names and different parameter type. Do we need a static method here just to create a object. Can we instead inline the method invocations?
24. Correct me if i'm wrong here. This appears to be used only within the `KafkaConsumerProxy`. Can we make it an internal static class of the `KafkaConsumerProxy` class?
25. Previously we were using SimpleConsumer in samza which had configurations (such as `zookeeper.connect`) which are no longer supported in the new kafkaConsumer. New kafka consumer configs: http://kafka.apache.org/documentation/#newconsumerconfigs. We need to update our samza configuration table. 1. Delete the old configuration references (systems.system-name.consumer.zookeeper.connect, etc) 2. Stress some of the important features of new kafkaConsumer(`SSL`, `keySerializer`, `valueSerializer`, `max.poll.records`). Our implementation overrides the defaults with user-defined configurations and it will be better to document it.
26. 1. Can you please share the rationale why we want to allow users to define configuration `auto.reset.offset` with predefined kafka constants directly? 2. Previously, `samza` did not allow users to specify value anything other than `largest`, `smallest` for the configuration `auto.reset.offset`. Now if we want to allow users to specify the predefined kafka `auto.reset.offset` configurations, can we update the configuration table documentation in accordance with this implementation.
27. Minor: s/KAFKA_CONSUMER_MAX_POLL_RECORDS_DEFAULT/DEFAULT_KAFKA_CONSUMER_MAX_POLL_RECORDS
28. Might be useful to document the default return value and also log the reset value if it isn't one of `latest` or `smallest`
29. Good to document public factory methods.
30. Can this be private since there's already a `getKafkaSystemConsumerConfig` factory method?
31. I think this line was too long so Intelij divided it into two lines.
32. done.
33. Done.
34. Done.
35. Done.
36. Done.
37. Please create a separate Jira for this. We can do it after this one.
38. In general I agree. But we cannot do easily it yet. For this we need to get rid of SimpleConsumer in the Admin. Which should be a separate PR.
39. Yes, it should be used in getAdmin in the factory! Thanks, will update.
40. The rest of the configs passed through the consumer.* are directly passed to the kafka consumer. This is the only one that requires translation. I think we should allow both new kafka settings, as well as support backward compatible samza values.
41. done.
42. This class doesn't talk directly to brokers anymore, so I thought we should get rid of Broker name.
43. done.
44. It is not a static inner class, so it will require some changes if we decide to extract it. Not sure if it is needed (this is how it was in the original implementation, so I didn't look into refactoring of it).
45. It may happened if Kafka decides throw an exception we don't handle. So may be it is better to throw exception in this case.
46. But 'r' is shorter! :) Changed it to record.
47. to minimize scala to java conversions.
48. I totally agree. I called it new just to differentiate from existing KafkaSystemConsumer. Now that it is deleted it can be removed. May screw up the diffs though..
49. At some point we may want to get rid of the BlockingEnvelopeMap in KafkaSystemConsumer entirely. But not now.
50. We can improve it when we change SystemAdmin to use new kafka consumer.
51. I think it is a good idea. Done.
52. to calculate this value we need to know number of partitions. But we know it only all of them are registered.
53. done.
54. Done.
55. not straight forward, but doable. Right now it is a object method, so using it requires creating an SystemAdmin object. Since it is scala, we cannot just make it static. It needs to be moved to the KafkaSystemAdmin object. And this will requires refactoring of all the places where it is used. This is a probably a good idea anyway, but adds even more changes to this pretty big PR. I've created SAMZA-1867 and will do it after this PR.
56. I think it makes it easier to read and reuse. I left the two which are used in multiple places.
57. it needs access to KafkaSystemConsumer members.
58. Just curious(Please correct me if i'm wrong). 1. Minor: Does it makes sense to rename this to `KafkaConsumerThread` or `KafkaMessageFetchThread` or `KafkaConsumerPollThread`(or some other name). I think the name `KafkaConsumerProxy` implies that this can be used in the place of KafkaConsumer object(with the proxy connotation introducing some additional

functionality on top of `KafkaConsumer`). 2. Does it makes sense to make this class package-private(Since it's used only within KafkaConsumer)?
59. Minor: Do we have to add non-null Precondition checks for some arguments here?
60. At line 124 in this class, we're adding all (k,v) pairs from the injectProps into final config bag. So if the user turns on kafka auto-commit by mistake, we would add it. Do we need to handle that scenario?
61. Minor: s/bootstrapServer/bootstrapServers
62. Minor(Please change if makes sense): Can simplify the return statement with option.getOrElse(). ``` String.format("%s-%s", jobNameOption.getOrElse("undefined_job_name"), jobIdOption.getOrElse("undefined_job_name")); ```
63. Just curious about this synchronized block. We pause and resume the `KafkaConsumer` for the SSP's from this list `topicPartitionsToPause` within the synchronized block. Why do we need to temporarily pause them if all we do is resume them right after poll? Can you please share the significance of this division and pausing, resuming a subset before/after poll?
64. Why this has to wait for 5 mins to stop the proxy thread. Can we set the proxy thread stop timeout to reasonably low value?
65. Minor: s/getNewKafkaSystemConsumer/getKafkaSystemConsumer
66. Would be great to take TimeUnit or duration as the second parameter here. From the arguments, it's not clear what is the timeunit of timeout parameter.
67. Do we need to interrupt the consumerPoll thread here after the timeout expires. Let's say If the thread did not die after this timeout (say got stuck in kafkaConsumer.fetch), what is the line of action?
68. > if Kafka decides throw an exception we don't handle. In that scenario, it will be propogated back to the caller, control-flow won't even reach the processResults method here.
69. Do we need to handle the interrupt in the consumer poll thread? Existing implementation doesn't seem to handle it and the consumer poll thread is interrupted during `KafkaSystemConsumer.stop()`
70. Why do we have the log level guard here. In slf4j/log4j combination, event can be logged unconditionally and message formatting is applied(for parameterized log statements) only based upon the configured log level of the application. To me this check introduces unnecessary complexity, what is the value add we get from this?
71. @shanthoosh: this should be expected behavior right? injected properties should always have highest precedence. injected properties != user-defined properties.
72. +1
73. Don't we explicitly "seek" to a particular offset anyways during register()? In that case, the Kafka configs for offset-reset have no effect?
74. Can we just do a new KafkaSystemConsumer()?
75. +1
76. Samza enforces thresholding based on the # of messages / partition. To achieve this, we pause those partitions for which we already have buffered messages for and un-pause the rest.
77. This change came up during testing and seems like a good thing to keep. CoordinatorStreamManager has its own life cycle, so unnecessary passing a reference to it into another class (with its own life cycle) is not desired.This way we can stop() it here, without worrying about the other reference.
78. 1. Naming came from BrokerProxy, and later LiKafkaConsumerProxy. So it may be a good idea to keep the name to preserve the history. 2. Do you mean keep the class or the constructor package private? I didn't see a lot of package private classes in our code.
79. I'd rather not add even more stuff to this PR. It is too big already.
80. Since we already manage offsets in samza through checkpoint topic, we should not allow users to turn auto-commit on to manage checkpoints in kafka(when using the kafkaSystemConsumer). Kafka-autocommit is unnecessary and could potentially create confusion given that we have our own mechanism to rewind/fast-forward the checkpointed offsets(with checkpoint tool). (Ex: What if the user turns on kafka auto-commit, stops the job and updates offsets stored in kafka. After this user would expect samza to resume from the updated offset in kafka. But the job would resume with the checkpointed offset stored in samza checkpoint topic) I think we should log the warning here if auto-commit is turned on and document in configuration table that it should not be turned on.
81. Interesting question. Generally speaking @vjagadish1989 is correct. But some settings may be required by samza. One of hem is disabling auto commits. So should we enforce it?
82. Since we control the partition assignment strategy in `JobCoordinator` via generated `JobModel`, setting this configuration is unnecessary and can be removed.
83. Done.
84. not so easy. Requires some scala magic. As is - jobConfig.getJobId().getOrElse("default") - doesn't compile.
85. This setting comes into play if the offset is invalid. User can override this behavior.
86. Done.
87. changed to 60 seconds.
88. Done.
89. I think disabling it is the best way forward, unless we've strong enough rationale not to do so.
90. > Do you mean keep the class or the constructor package private? I didn't see a lot of package private classes in our code. I meant, if possible we can make this whole class package private. Looking at the usages, it's doable. General rule of thumb is to choose lowest level of visibility for a class.
91. s/kc/KafkaSystemConsumer/
92. Minor: variable names should follow camel-case convention. s/SSPsToFetch/partitionsToFetch
93. Done.
94. In this case we need it (because there is a loop), but in other cases I've removed it.
95. Done.
96. I guess you are right, but this is just a good coding style - check function's arguments :)
97. done. sspsToFetch
98. Done.
99. Please delete these unnecessary commented out empty code block.

100. Can we please delete this empty method.
101. Why do we log the message here and then throw up. The codeflow that triggers `register(ssp, offset)` already logs any exception. Please delete it if it's redundant.
102. Typo: "SamzaContainer"
103. Why do we need to extend Kafka's ConsumerConfig class?
104. I don't think the "will fetch ALL available messages for all partitions" comment is true. Can you confirm? There still seems to be a default MAX_POLL_RECORDS of 500 in the ConsumerConfig. Also, should this be private?
105. Should provide meaningful param documentation and document what this method does in some detail.
106. Prefer not adding `final` to local variables, unless we do this consistently across the codebase. IMHO it's not worth doing.
107. Prefer choosing the desired behavior and updating this comment instead of leaving TODOs and open questions in code.
108. "setting default key serialization" -> "Setting key serializer" "consumer(for {})" -> "consumer for system {}" Same below.
109. Do we still need to support injectedProperties or can we inline those here?
110. Add newline between methods.
111. Does "\\W" work? https://docs.oracle.com/javase/tutorial/essential/regex/pre_char_classes.html
112. Will be cleaner to pass the autoOffsetReset string explicitly instead of passing all Properties.
113. Maybe just `metrics`?
114. What's this metricName? Is this a prefix? Also, doesn't look like this is used outside the constructor. Do we need this field?
115. IMHO we don't need to mark fields and methods package private explicitly - it's boilerplate and doesn't add value. Would prefer deleting this everywhere.
116. Remove comment or add more details of what this means and how it's used.
117. Prefer using topicPartitionsToOffset, Not worth saving 1 character IMHO. Would also be good to document what these maps represent.
118. Prefer adding meaningful javadocs or not adding them at all, preferably the former.
119. Use this.messageSink etc consistently. Same for proxy.
120. s/it is/it has
121. Do we need to log the proxy object reference? Maybe "Created KafkaConsumerProxy"?
122. Isn't metricName derived from systemName and clientId? Probably don't need to log again. What does KafkaConsumer.toString return?
123. Does this need to be public?
124. Please clean up javadocs for public APIs.
125. Can we remove injectedProperties?
126. Please capitalize first letter in log statements and javadocs everywhere.
127. Should this be a exception?
128. Log consumer id here and in exception message for context.
129. Incorrect javadoc format.
130. Should not refer to proxy thread, that's implementation details of the proxy,
131. Which other exceptions is this referring to? There don't seem to be any recoverable exceptions.
132. Log consumer ID for context. Use the same message in the exception for traceability.
133. Log consumer id everywhere.
134. Strongly prefer combining these messages into a single log line. Same for other logs from this method.
135. Comment does not match code. Also, should this be hardcoded?
136. Is this the right class name? Also, the `+` should not be in the message.
137. Should be before we call super.register.
138. Does this need to be public?
139. Don't think we need to log this. This is not useful for users.
140. `return super.poll()` directly?
141. Use Javadoc format. Don't need a separate block for this. Same at the end.
142. What does this metric represent? Number of registered TopicPartitions? If so, maybe call this registered-topic-partitions?
143. Can the KafkaSystemConsumer create this KafkaConsumer itself?
144. "Created KafkaSystemConsumer"?
145. Can KafkaSystemProducer create Producer instances itself?
146. Indent this block by 2 more so that it's aligned correctly.
147. This class isn't the thread, it has a thread, so the javadoc for this class shouldn't refer to it as a thread. It also doesn't interact with the BEM directly, so should mention the KafkaConsumerSink instead, and add the BEM related documentation to the Sink.
148. Explicit check + throw exception instead of assert.
149. Also, do we need to keep these maps both this class and the Proxy, or can we move them to the proxy?
150. Use Javadoc format.
151. s/is called/should be called
152. Is this still true?
153. Also, incorrect class name.
154. Incorrect class name.
155. We aren't stopping anything here?
156. Move private methods to end of class. Prefer the following order: Constructors, public, protected, package private, private, inner classes.
157. Does this need to be public? If so, add javadoc for public methods.
158. If this map doesn't change after start, this check should be in start instead of here.
159. Include context in exception message. E.g., systemName, clientId etc.
160. Would be better to have a more explicit string (e.g. "systemName={} clientId={}" etc) instead of making users figure out what the "/" separated parts are. Also, what's included in super.toString?

161. Is this logic available in open source? I believe this would be linkedin-kafka-clients specific since this is for large message support?
162. Can remove comment. There's also no 'client' in Kafka, so is not accurate.
163. If we don't handle these exceptions, would it be better to not catch them here and try-catch-log them in the caller instead? Even otherwise, since we don't handle them differently, you could just catch Exception, Log it with context, and rethrow it with context.
164. I don't see the benefit of throwing SamzaException here. Even if you don't do that, you'll get an NPE with the same amount of information. Should either delete this, or log more context in SamzaException that may be helpful for debugging.
165. Use Javadoc format for methods everywhere in this class.
166. Delete unused code.
167. Does this need to be public?
168. Fix todo?
169. @vjagadish1989 Speaking of which, this behavior initially caused a perf degradation that the kafka team had to fix. Do you recall if their fix went into Kafka main?
170. Would prefer to just inline these offset type constants in the method.
171. Can use isBlank instead of isEmpty.
172. This should be in a Samza package.
173. I agree about the utility of allowing users to set auto.reset.offset, but doesn't make sense to allow using Samza's config values for Kafka's config keys. We should to remove this method and pass the user provided consumer config as-is.
174. Can we delete this method?
175. Extract out of the if block and reuse in SamzaException for the else block.
176. Comment should be above @Test annotation.
177. Don't need empty setup method.
178. createConsumer?
179. Should stop the consumer to prevent leaking Consumers during tests.
180. Typo: MockKafkaSystemConsumer
181. Should be warn, since this is not a normal use case.
182. Not sure what this is talking about. Can you clarify what this means?
183. Can delete these comments.
184. you are right, we don't need to. Fixed.
185. Fixed the comment. It is package private because it is used in the tests.
186. I guess consistency is the key. Removed.
187. My understanding, it allows the caller to add some configs, which are not part of the app provided config. Not sure if there are any specific use cases. It is used in some tests, but can easily be gotten rid off.
188. According to the document it does. Changed. Added test. Tested.
189. According to Samza Configuration [page](https://samza.apache.org/learn/documentation/0.14/jobs/configuration-table.html) use should provide Samza values, and we need to translate it. If you, guys, think we need to change - let's do it in a separate, small patch.
190. we should, but not in this patch. It is still used in few places. I can clean up after we commit this patch and systemAdmin one. Filed SAMZA-1899.
191. Agree, removed the question. If bootstrap_servers is not provided under 'consumer' configs, we get it from the 'producer' configs.
192. done.
193. done.
194. done.
195. done.
196. nope. fixed.
197. Fixed.
198. changed wording to '..proxy has stopped'.
199. ok.
200. done.
201. good point. done.
202. done.
203. done.
204. changed the wording, but left synchronization in.
205. done. catch Exception, log, rethrow.
206. Added more context. I don't like NPEs. They look like a missed check for null, but in this case it is not.
207. changed to '//' for private methods.
208. done.
209. done.
210. removed todo. This timeout is different from consumer.request.timeout.ms. So it should be internal to Samza. No sense adding another config at this point.
211. At first I thought it may be a good idea to have 'package private', to show it is not an omission, but, when there are too many of those, it doesn't make sense anymore.
212. added comment.
213. Added documentation and renamed (it is not about 1 char, more about readability). The maps in the Consumer are used for registration, and then transferred to the proxy on start()
214. done. this.* should be used only when the scope is ambiguous.
215. changed the comment.
216. added toString()

217. Fixed the logging, but left in the kafkaConsumer. Since we may have multiple consumers for the same system, this will help to differentiate between them.
218. It is used in KafkaSystemFactory
219. In theory yes. In practice, the stopping sequence are not well debugged and we may end up with calling stop from a few places. There should be no harm in ignoring it in this case.
220. Deleted the comment. The original idea was, on resetting the consumer to first/last message, instead of querying admin for first/last offset, just pass a special value for the offset, and use seekToBeginning()/End() here.
221. All recoverable exceptions are taken care by Kafka consumer. If we get here - we need to throw. Changed the comments.
222. normal operations should be well under the timeout. I don't see a need for a config here.
223. fixed.
224. Done.
225. changed it to systemname:clientid all over the class.
226. Actually, since one can pass a null as the 'cause', no 'if' needed at all.
227. see above.
228. registered is kind of implicit here. But, sure, why not.
229. I've used this static method to avoid awkward scala-java conversions. But this part may change in the other patch (changing SystemAdmin) and I will review it there.
230. this is not related to this patch. Just formatting.
231. Scala formatting is causing many issues with checkStyle. So I prefer to delegate this task to Intelij. If checkstyle fails I will address it.
232. sure.
233. Fixed the typo. Thanks.
234. Right. Fixed.
235. Why does this class need to extend HashMap? If this is intended to be used as a Samza config class, should this be MapConfig instead?
236. Can you clarify what this is for, and how this is different than kafka's ConsumerConfig?
237. No `-` between param name and documentation for consistency with rest of javadocs.
238. "Helper method to create configs for"
239. Maybe: "system name to get consumer configuration for"
240. Maybe: "if consumer bootstrap servers aren't configured, get them from producer configs"
241. Shouldn't jobName and jobId be always available? Would prefer to throw if not instead of defaulting to arbitrary values.
242. Prefer using JobConfig.getJobName
243. Prefer using JobConfig.getJobId, which already default to 1.
244. We shouldn't allow samza values ("largest", "smallest") for kafka consumer configuration - that's a misconfiguration. Would prefer to remove this method.
245. Can remove `/* package private */` everywhere.
246. Can remove comment.
247. Prefer moving to the bottom, even though this is public.
248. `this.toString` only adds the clientId which is not meaningful, so prefer not logging it.
249. "Ignoring InterruptedException while waiting for consumer poll thread to start."
250. The clientId isn't useful in these messages since it just tells us the jobName and jobId, which we'd know anyway if when looking at the logs. Would prefer to remove everywhere.
251. Maybe: "Cannot start KafkaConsumerProxy without any registered TopicPartitions for system %s"
252. Log systemName/this.
253. Stop this KafkaConsumerProxy and wait for at most {@code timeoutMs}.
254. Public method javadoc shouldn't refer to implementation details. Maybe: "maximum time to wait to stop this KafkaConsumerProxy"
255. s/event/even. s/join returns/join may timeout and return Should wait for timeoutMs/2 in both calls if the API documents that timeout is the maximum amount.
256. Maybe; 'must only be called before {@link KafkaConsumerProxy#start} is called.' Also, prefer arranging public methods in the order they're expected to be called in. Here, addTopicPartition, start, stop.
257. Minor: topicPartitionsToSSP.
258. Would be useful to clarify what `next offsets` means here.
259. Minor: "partition %s with offset %s to queue" for consistency.
260. This is already logged above, so can remove this log.
261. kafkaConsumerMetrics.setNumTopicPartitions?
262. What does this comment mean?
263. Include system name in log message. Also, wrap e in a new SamzaException with the same message for context.
264. Maybe: ("Received null 'records' after polling consumer in KafkaConsumerProxy " + this)
265. I think 'already de-serialized' isn't true for this implementation, remove comment.
266. From HashMap javadoc: `When the number of entries in the hash table exceeds the product of the load factor and the current capacity, the hash table is <i>rehashed</i> (that is, internal data structures are rebuilt) so that the hash table has approximately twice the number of buckets.` You should use a higher count ((records.count()/0.75) + 1) or load factor if the intention behind passing count is to avoid a resize.
267. Prefer `messages`
268. Prefer no `final` for local variables.
269. Prefer `ime`.
270. Formatting: add space before `=`, and remove extra space after it.
271. Maybe: "Starting consumer poll thread {} for system {}", consumerPollThread.getName, systemName
272. s/Latency/Lag

273. Maybe: "pollConsumer for {} SSPs: {}", sspToFetch.size(), sspsToFetch
274. This is logged a few lines later, so can remove.
275. s/timeout/timeoutMs
276. Minor: Space between `;` and `//`
277. Prefer inlining this method in populateCurrentLags since it's only used in one place. I think there are too many metrics/lag related methods in this class already.
278. Prefer `currentLagMetric`
279. Prefer inlining single use helper method, since there are too many metrics/lag related methods in this class.
280. Include system name.
281. This is a similar comment to pollConsumer. I'd prefer to clarify how they're related. Also, since pollConsumer is only called within this method, this method should be above pollConsumer in this class.
282. Minor: can remove extra newline.
283. `consumer {}` should be systemName instead.
284. protected/package private methods should be above private methods, right after public methods.
285. I'd propose the following method order for readability: ``` package private Methods: addTopicPartition start isRunning stop getFailureCause private methods: createProxyThreadRunnable initializeLags fetchMessages pollConsumer processResults moveMessagesToTheirQueue populateCurrentLags (inline populateMetricNames) updateMetrics (inline getLatestLag) getRecordSize ```
286. I think these methods can all be package private (I mean remove `public`, not add the /* package private */ annotation) since the only user should be KafkaSystemConsumer
287. Prefer logging systemName explicitly instead of this. In this class this can refer to the thread or this class depending on where it's used, so would prefer to not use it in logs to avoid confusion.
288. Typo: "BlockingEnvelopeMap"
289. "and their offsets"
290. Include description for javadocs. Maybe: "Create a KafkaSystemConsumer for the provided {@code systemName}"
291. "application config"
292. "metrics for this KafkaSystemConsumer"
293. No '-' in param docs. ". Should be sufficient to say "clock system clock"
294. This shouldn't be here since the consumer is being passed in.
295. Is the space separator in metricName intentional? What is this used for? If this is prefix for metrics, do we need clientId in it - it's just the jobName + jobId.
296. Would prefer to log everything in this class as "Created KafkaConsumerProxy {} for systemName {}" for consistency with rest of the codebase.
297. Don't need clientId. Also see comment about log message format above.
298. Can this be done in KafkaConsumerProxy#start instead? Would be better to consolidate interactions with the KafkaConsumer in one class.
299. This should this be an exception, or at least an error log.
300. Should be error.
301. Include context in exception message.
302. Should this be an error?
303. Can this entire method be in KafkaConsumerProxy to consolidate consumer management in one class (as the name KafkaConsumerProxy implies).
304. "numPartitions"
305. Would prefer to combine these log lines in one message at the end of this method if possible. Currently this adds log noise every time a new consumer is created, which is often.
306. Use javadoc format.
307. Same comment as other class: prefer ordering methods logically by how they're used. E.g., it makes more sense to order them as 'register, start, poll, stop' etc than the current order of 'start, stop, register, poll'. Also, prefer ordering methods by visibility: public, protected then private methods.
308. this class is used to initialize kafka's KafkaConsumer, so which expects Map<String, Object). We cannot use kafka's ConsumerConfig because it is packaged private for kafka packages.
309. As mentioned above, since we cannot use ConsumerConfig directly, we need a class to pass to KafkaConsumer.
310. This was done to match current functionality. I guess, we should change it to be consistent with getClientId logic, which does throw.
311. These are the correct 'samza' values. We cannot remove them if we want to be backward compatible (https://samza.apache.org/learn/documentation/0.14/jobs/configuration-table.html)
312. this is the only differentiator between different proxies.
313. we can have multiple consumers and multiple proxies. I think it can be helpful to see if a log comes from different proxies/consumers.
314. changed the comment.
315. removed the code.
316. I think this code was removed.
317. I will remove client id here.
318. will put this change for later.
319. removed.
320. made it error.
321. May be. I don't think we should do it in this patch. I will add a jira for it.
322. will do in later patch.

jira_issues:

jira_issues_comments: