Item 102
git_comments:

1. * * Use STS to assume an IAM role for temporary session-based authentication. Will use configured * long-lived credentials for authorizing to STS itself (either the default provider chain * or a configured keypair). STS will validate the provided external ID with the one defined * in the trust policy of the IAM role to be assumed (if one is present). * * @param roleArn ARN of IAM role to assume via STS * @param sessionName Name to use for the STS session * @param externalId External ID to validate against assumed IAM role's trust policy * @return Reference to this [[SparkAWSCredentials.Builder]]
2. * * Use a basic AWS keypair for long-lived authorization. * * @note The given AWS keypair will be saved in DStream checkpoints if checkpointing is * enabled. Make sure that your checkpoint directory is secure. Prefer using the * [[http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html#credentials-default default provider chain]] * instead if possible. * * @param accessKeyId AWS access key ID * @param secretKey AWS secret key * @return Reference to this [[SparkAWSCredentials.Builder]]
3. * Returns DefaultAWSCredentialsProviderChain for authentication.
4. * * Returns AWSStaticCredentialsProvider constructed using basic AWS keypair. Falls back to using * DefaultCredentialsProviderChain if unable to construct a AWSCredentialsProviderChain * instance with the provided arguments (e.g. if they are null).
5. * * Serializable interface providing a method executors can call to obtain an * AWSCredentialsProvider instance for authenticating to AWS services.
6. * * Builder for [[SparkAWSCredentials]] instances. * * @since 2.2.0
7. scalastyle:on
8. * * Returns the appropriate instance of [[SparkAWSCredentials]] given the configured * parameters. * * - The long-lived credentials will either be [[DefaultCredentials]] or [[BasicCredentials]] * if they were provided. * * - If STS credentials were provided, the configured long-lived credentials will be added to * them and the result will be returned. * * - The long-lived credentials will be returned otherwise. * * @return [[SparkAWSCredentials]] to use for configured parameters
9. * * Return an AWSCredentialProvider instance that can be used by the Kinesis Client * Library to authenticate to AWS services (Kinesis, CloudWatch and DynamoDB).
10. * 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.
11. * * Returns an STSAssumeRoleSessionCredentialsProvider instance which assumes an IAM * role in order to authenticate against resources in an external account.
12. * * Use STS to assume an IAM role for temporary session-based authentication. Will use configured * long-lived credentials for authorizing to STS itself (either the default provider chain * or a configured keypair). * * @param roleArn ARN of IAM role to assume via STS * @param sessionName Name to use for the STS session * @return Reference to this [[SparkAWSCredentials.Builder]]
13. scalastyle:off
14. * * Creates a [[SparkAWSCredentials.Builder]] for constructing * [[SparkAWSCredentials]] instances. * * @since 2.2.0 * * @return [[SparkAWSCredentials.Builder]] instance
15. * 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.
16. * * Basic test to ensure that the KinesisDStream.Builder interface is accessible from Java.
17. * * Sets the name of the Kinesis stream that the DStream will read from. This is a required * parameter. * * @param streamName Name of Kinesis stream that the DStream will read from * @return Reference to this [[KinesisInputDStream.Builder]]
18. * * Builder for [[KinesisInputDStream]] instances. * * @since 2.2.0
19. * * Create a new instance of [[KinesisInputDStream]] with configured parameters and the provided * message handler. * * @param handler Function converting [[Record]] instances read by the KCL to DStream type [[T]] * @return Instance of [[KinesisInputDStream]] constructed with configured parameters
20. * * Sets the initial position data is read from in the Kinesis stream. Defaults to * [[InitialPositionInStream.LATEST]] if no custom value is specified. * * @param initialPosition InitialPositionInStream value specifying where Spark Streaming * will start reading records in the Kinesis stream from * @return Reference to this [[KinesisInputDStream.Builder]]
21. Required params
22. * * Sets the AWS region to construct clients for. Defaults to "us-east-1" if no custom value * is specified. * * @param regionName Name of AWS region to use (e.g. "us-west-2") * @return Reference to this [[KinesisInputDStream.Builder]]
23. * * Creates a [[KinesisInputDStream.Builder]] for constructing [[KinesisInputDStream]] instances. * * @since 2.2.0 * * @return [[KinesisInputDStream.Builder]] instance
24. Params with defaults
25. * * Sets the KCL application name to use when checkpointing state to DynamoDB. This is a * required parameter. * * @param appName Value to use for the KCL app name (used when creating the DynamoDB checkpoint * table and when writing metrics to CloudWatch) * @return Reference to this [[KinesisInputDStream.Builder]]
26. * * Sets the StreamingContext that will be used to construct the Kinesis DStream. This is a * required parameter. * * @param ssc [[StreamingContext]] used to construct Kinesis DStreams * @return Reference to this [[KinesisInputDStream.Builder]]
27. * * Sets how often the KCL application state is checkpointed to DynamoDB. Defaults to the Spark * Streaming batch interval if no custom value is specified. * * @param interval [[Duration]] specifying how often the KCL state should be checkpointed to * DynamoDB. * @return Reference to this [[KinesisInputDStream.Builder]]
28. * * Sets the [[SparkAWSCredentials]] to use for authenticating to the AWS Kinesis * endpoint. Defaults to [[DefaultCredentialsProvider]] if no custom value is specified. * * @param credentials [[SparkAWSCredentials]] to use for Kinesis authentication
29. * * Sets the storage level of the blocks for the DStream created. Defaults to * [[StorageLevel.MEMORY_AND_DISK_2]] if no custom value is specified. * * @param storageLevel [[StorageLevel]] to use for the DStream data blocks * @return Reference to this [[KinesisInputDStream.Builder]]
30. * * Sets the [[SparkAWSCredentials]] to use for authenticating to the AWS DynamoDB * endpoint. Will use the same credentials used for AWS Kinesis if no custom value is set. * * @param credentials [[SparkAWSCredentials]] to use for DynamoDB authentication
31. * * Create a new instance of [[KinesisInputDStream]] with configured parameters and using the * default message handler, which returns [[Array[Byte]]]. * * @return Instance of [[KinesisInputDStream]] constructed with configured parameters
32. * * Sets the StreamingContext that will be used to construct the Kinesis DStream. This is a * required parameter. * * @param jssc [[JavaStreamingContext]] used to construct Kinesis DStreams * @return Reference to this [[KinesisInputDStream.Builder]]
33. * * Sets the AWS Kinesis endpoint URL. Defaults to "https://kinesis.us-east-1.amazonaws.com" if * no custom value is specified * * @param url Kinesis endpoint URL to use * @return Reference to this [[KinesisInputDStream.Builder]]
34. * * Sets the [[SparkAWSCredentials]] to use for authenticating to the AWS CloudWatch * endpoint. Will use the same credentials used for AWS Kinesis if no custom value is set. * * @param credentials [[SparkAWSCredentials]] to use for CloudWatch authentication

git_commits:

1. **summary:** [SPARK-19911][STREAMING] Add builder interface for Kinesis DStreams
message: [SPARK-19911][STREAMING] Add builder interface for Kinesis DStreams ## What changes were proposed in this pull request? - Add new KinesisDStream.scala containing KinesisDStream.Builder class - Add KinesisDStreamBuilderSuite test suite - Make KinesisInputDStream ctor args package private for testing - Add JavaKinesisDStreamBuilderSuite test suite - Add args to KinesisInputDStream and KinesisReceiver for optional service-specific auth (Kinesis, DynamoDB and CloudWatch) ## How was this patch tested? Added ```KinesisDStreamBuilderSuite``` to verify builder class works as expected Author: Adam Budde <budde@amazon.com> Closes #17250 from budde/KinesisStreamBuilder.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-19911][STREAMING] Add builder interface for Kinesis DStreams
   body: ## What changes were proposed in this pull request? - Add new KinesisDStream.scala containing KinesisDStream.Builder class - Add KinesisDStreamBuilderSuite test suite - Make KinesisInputDStream ctor args package private for testing - Add JavaKinesisDStreamBuilderSuite test suite - Add args to KinesisInputDStream and KinesisReceiver for optional service-specific auth (Kinesis, DynamoDB and CloudWatch) ## How was this patch tested? Added ```KinesisDStreamBuilderSuite``` to verify builder class works as expected

github_pulls_comments:

1. Open questions I'd like feedback on: * Should the ```KinesisUtils.createStream()``` methods be marked as deprecated? * Should the ```KinesisUtils.createStream()``` methods be refactored to use the builder? * ~Should I add full docs for each method (e.g. including ```@param``` lists)?~ EDIT: has been added * Does the file name and class name I've added seem reasonable? * Is making the ctor args to ```KinesisInputDStream``` package private for testing reasonable? I'd like to also extend this to allow configuring CloudWatch and DynamoDB-specific authorization which I imagine will be quite helpful to users. ~I'm trying to decide if I should do this as a separate PR or just roll it in here.~ EDIT: has been added to this PR Pinging @brkyvz and @srowen from #16744
2. **[Test build #74342 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/74342/testReport)** for PR 17250 at commit [`8552caf`](https://github.com/apache/spark/commit/8552caf84dbd5b43a5a8446b10e4ccf9b295f924). * This patch **fails Spark unit tests**. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * ` class Builder[T: ClassTag](`
3. Forgot to stop the ```StreamingContext``` added in ```KinesisDStreamBuilderSuite```. Updated the code to stop the context after all tests have run.
4. **[Test build #74351 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/74351/testReport)** for PR 17250 at commit [`bcb7667`](https://github.com/apache/spark/commit/bcb76677b9fa58e5b96920d9faaa338183cca26d). * This patch passes all tests. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * ` class Builder[T: ClassTag](`
5. **[Test build #74379 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/74379/testReport)** for PR 17250 at commit [`a604dc5`](https://github.com/apache/spark/commit/a604dc5952a1c9939d433371abc969670a8ff6ab). * This patch passes all tests. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * ` class Builder[T: ClassTag](`
6. Found a bit more time to work on this. Changes made: * Added implementations of ```KinesisDStream.builder()``` that take ```JavaStreamingContext``` instead of ```StreamingContext``` * Added ```JavaKinesisDStreamBuilderSuite``` to test that the builder interface is accessible in Java * Added support for DynamoDB and CloudWatch-specific authentication parameters * Made the documentation for ```KinesisDStream.Builder``` more thorough
7. **[Test build #74418 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/74418/testReport)** for PR 17250 at commit [`8aeef08`](https://github.com/apache/spark/commit/8aeef08410999910e864d1de07f6eae21fd5409a). * This patch passes all tests. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * ` class Builder[T: ClassTag](`
8. @brkyvz Thanks for taking a look! Re: major feedback: 1. Sounds reasonable to me. I don't have strong feeling here. 1. Will do. 1. Sounds good. 1. I agree that method overloading isn't a great and could lead to a similar scenario as ```KinesisUtils.createStream()``` in the long run. I kept it here since I figured that the scope was rather limited and I wanted to avoid having to deal with too many builders but I think you raise a good point. I'll look into adding a credentials builder. Ideally we wouldn't have to worry about Java/Python interoperability and could just use a Scala method with optionals or default args to solve this but I guess we have to play the hands we're dealt :) I can move the builder to the companion object of ```KinesisInputDStream``` (this is actually where I had this placed originally). I like the idea of the private constructor as well although I don't think this will be possible until ```KinesisUtils.createStream()``` is refactored to use the builder pattern.
9. @brkyvz Actually, now that I think about it, do we need to make ```messageHandler``` a constructor arg since ```Builder``` is a generic class? There's probably a way we could get around this but I'd imagine it would be pretty complex...
10. Good point @budde. I can think of two options: 1. Leave it as a constructor param 2. Make the `Builder` class non-generic and have the `build` function take the message handler: ```scala class Builder { def build(): KinesisInputDStream[Array[Byte]] def buildWithMessageHandler[T](f: Record => T): KinesisInputDStream[T] } ``` It's a matter of taking it as the first parameter or the final parameter. There are other ways to do it as well, but will throw runtime exceptions instead of at compile time. cc @rxin for input on APIs
11. @brkyvz I think if we're eliminating the constructor arguments then the second approach you've proposed might make more sense. I can't think of anything cleaner.
12. @budde Do you think you can update this PR? The 2.2 branch will be cut on Monday (2017-03-18).
13. @brkyvz A conference took up a lot of my time last week but I should have it updated later today
14. @brkyvz PR has been updated, apologies for the delay. I've added ```SerializableCredentialsProvider.Builder```, which I'm willing to hear suggestions for a better name on. I wanted to stay away from something like ```AWSCredentials.Builder``` so as to avoid confusion with similarly-named classes in the AWS Java SDK.
15. **[Test build #74901 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/74901/testReport)** for PR 17250 at commit [`d6afaef`](https://github.com/apache/spark/commit/d6afaef4099d9b20d0d1257ec5942d1bb5b868af). * This patch passes all tests. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * ` class Builder ` * ` class Builder `
16. @brkyvz Updated the PR to remove ```defaultMessageHandler()``` from ```KinesisUtils``` in favor of keeping this method in ```KinesisInputDStream```. My thought here was that this would be a better place for it since we've put ```KinesisUtils``` on the deprecation path.
17. **[Test build #74909 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/74909/testReport)** for PR 17250 at commit [`3cc2df8`](https://github.com/apache/spark/commit/3cc2df82f5ae6b4e3a21c2549a8855f5bd8a3eb6). * This patch passes all tests. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * ` class Builder ` * ` class Builder `
18. @brkyvz Updated per your feedback. Most significant change is renaming ```SerializableCredentialsProvider``` to ```SparkAWSCredentials``` (as well as renaming its subclasses) and refactoring its builder as you've suggested.
19. **[Test build #75065 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/75065/testReport)** for PR 17250 at commit [`337b6ba`](https://github.com/apache/spark/commit/337b6ba575c969417566ee65664c93916e6923d4). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
20. @brkyvz Updated per your feedback, thanks for taking a thorough look. I also renamed the ```longLivedCredsProvider``` of ```STSCredentials``` to just ```longLivedCreds``` to match the updated naming conventions.
21. Thanks a lot for the quick turnaround @budde ! Could you also contribute to the docs as well with the new builder API? https://spark.apache.org/docs/latest/streaming-kinesis-integration.html
22. **[Test build #75111 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/75111/testReport)** for PR 17250 at commit [`6f11978`](https://github.com/apache/spark/commit/6f11978326e59e4d383d7a64eef981b917364af6). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
23. @brkyvz Sure, want me to add it to this PR or open a new one?
24. Thanks! new PR would be easier!
25. LGTM pending tests. Thanks a lot for this PR @budde PS It's okay to make new commits, you don't have to squash commit every time :)
26. **[Test build #75170 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/75170/testReport)** for PR 17250 at commit [`5315f1e`](https://github.com/apache/spark/commit/5315f1e9f200be7f97d7838c2d64bc6bfb208ecb). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
27. **[Test build #75172 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/75172/testReport)** for PR 17250 at commit [`03f91da`](https://github.com/apache/spark/commit/03f91dadb3878aa47f2a134e3e1b8d46aadd3b47). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
28. Merging to master
29. @brkyvz Awesome, thanks for reviewing this!

github_pulls_reviews:

1. I feel that builders rarely have constructor params. I understand you wanted to make these the `required` options, but I would just rather have the builder have a `zero-param` constructor, and it checks for the required fields upon `build()`. What do you think?
2. I would also make this a `required` field, otherwise people will face confusing issues when they start 2 streams from the same Spark application.
3. Would be great to document what happens when both region and endpoint is set, but are referring to different regions
4. could you link `InitialPositionInStream` for simplicity
5. We shouldn't default to the Spark app name
6. Kinda feels unnecessary?
7. I would get rid of the constructor params, and check for required values here, i.e. 1. `ssc` 2. `streamName` 3. `appName`
8. Have the builder take `setStreamingContext`, one which takes `StreamingContext` and the other `JavaStreamingContext`
9. let's also have a `setMessageHandler` function as well
10. nit: move `=` to line above
11. why the underscore?
12. With this many methods for the credential provider, I feel we need a credential provider builder. I wouldn't want to re-enter everything again between `dynamoDb` and `cloudWatch`. If I want to keep them separate from the `kinesis` credentials
13. Declaring ```val storageLevel``` collides with [```DStream.storageLevel```] (https://github.com/apache/spark/blob/d5b1d5fc80153571c308130833d0c0774de62c92/streaming/src/main/scala/org/apache/spark/streaming/dstream/DStream.scala
14. Will do
15. Will do
16. Will be made required builder arguments
17. I'll expand the builder to recognize required params and remove them as ctor args
18. I'll look at introducing a credential builder
19. Yeah, I mostly put this here for the sake of completeness. Should be able to remove this if we introduce a credential builder and replace it with a single method that just takes a ```SerializableCredentialsProvider``` instance to use for the service. A user could always just manually pass ```DefaultCredentialsProvider``` if they really want to for some reason.
20. We'll make this required
21. Will do
22. I'll take a look and see. To be honest supplying both of these has always felt pretty redundant to me. The AWS SDK has changed a bit in how it handles endpoints and regions as well, so it may also be worth revisiting how ```KinesisReceiver``` uses these params. Long term, it may also be nice to allow for different endpoints to be specified for Kinesis, DynamoDB and CloudWatch (I think the KCL should support this...)
23. Sounds reasonable. Feel free to push back on any other defaults as well-- figured these would just be a starting point.
24. This is probably the first builder class I've implemented so I'll defer to your judgment here :)
25. would you mind just re-using the code in `KinesisUtils` instead of copying the code?
26. How about keeping it here and refactoring ```KinesisUtils``` to use it? I think this is what I was intending to do originally, just forgot to update the code.
27. nit: move `=` to the line above
28. hmm. This isn't a great name for a user facing API, the user shouldn't have to care about if the provider is serializable or not, that's an implementation detail. I understand your concerns with the `AWSCredentials` name collisions. However, I think it's the best name there is.
29. anyone who provides an `accessKeyId` should also provide a `secretKey` therefore I would take both together. `.withKeys(awsAccessKey: String, awsSecretKey: String)`
30. same here: ```scala def withSts(roleArn: String, sessionName: String) def withSts(roleArn: String, sessionName: String, externalId: String) ```
31. I guess you no longer need this
32. I wouldn't call these `Stable` just yet :) Let's call it evolving for one release cycle
33. ditto, let's call it evolving for now
34. sounds good
35. How about simply `CredentialProvider`
36. You're right, thanks for catching it
37. Will fix. Sorry I keep doing this :-/
38. For sure. Honestly I just cribbed these annotations from ```SparkSession.Builder``` so I appreciate you letting me know what the proper convention is.
39. Will do
40. I agree we should definitely come up with a better name here. What about ```SparkAWSCredentials```? Obviously it's not as succinct as ```AWSCredentials``` but I think it's a clear name that avoids collisions. I'm okay with ```CredentialsProvider``` otherwise.
41. I'll rework this builder to take multiple arguments for the long-lived keypair and STS
42. Will do
43. I guess `SparkAWSCredentials` also work
44. The `private`s are unnecessary since the class itself is private. You can either: 1) remove `private[kinesis]` 2) make it `private val` and then use `PrivateMethodTester` in the tests. I'm fine either way.
45. `SparkAWSCredentials`
46. ditto
47. ditto
48. where does this come from? `LocalJavaStreamingContext`? If so, I wouldn't stop it. In fact, you can use `Mockito` to create a mock `JavaStreamingContext` if you like
49. since we're not really starting a stream, and testing API's, we should just mock it.
50. I would split this into 3 very small tests: `should raise an exception if StreamingContext is missing` `should raise an exception if stream name is missing` `should raise an exception if checkpoint app name is missing`
51. nit: could you make this a single line?
52. nit: single line please.
53. I know we didn't have it before, but could you also check that after deserialization, they're equivalent? ```scala val creds = BasicCredentials("x", "y") assert(Utils.deserialize[BasicCredentials](Utils.serialize(creds)) === creds) ```
54. My bad, thanks
55. Will fix and do a "grep -r 'SerialziableCredentialsProvider' *" to make sure this isn't appearing anywhere else
56. I'll probably just go the mock route then and ignore the context all together. I was seeing a bunch of "Spark context is already running" error messages when I tried to run all of the streaming tests before I added this
57. Will do
58. Sounds good
59. I'll fix it. This happened since ```DefaultCredentialsProvider``` was shortened to ```DefaultCredentials``` so I'll try to check for other places where a multiline statement can be rolled up into a single line
60. Will do
61. Will do
62. Yeah, it's really just there so that I could access the values directly from the test. I'll look into using ```PrivateMethodTester```, thanks for the suggestion
63. If you keep it `val` it should be fine.
64. Figured out why I didn't catch this before-- apparently I can't spell "serializable". Ugh!
65. Yeah, was just thinking that'd be a lot simpler. I'll go that route. Thanks!
66. Using a mock here and in the other test might not be very practical after all-- looks like the ```DStream``` constructor hooks into ```StreamingContext```. We would at least need to mock it's [```getState()``` method] (https://github.com/apache/spark/blob/d5b1d5fc80153571c308130833d0c0774de62c92/streaming/src/main/scala/org/apache/spark/streaming/dstream/DStream.scala as well as [mocking a ```SparkContext``` along with its local properties] (https://github.com/apache/spark/blob/d5b1d5fc80153571c308130833d0c0774de62c92/streaming/src/main/scala/org/apache/spark/streaming/dstream/DStream.scala *Edit: this might not be as bad as I thought-- I'll keep trying the mock approach*
67. that's fine then. As long as we don't break the environment for other tests, do proper clean up, it should be fine

68. Looks like this will need to be left as-is. In the current test implementation we check that ```checkpointInterval``` isn't a required option and its default value is obtained via ```ssc.graph.batchDuration```, which we won't be able to mock because ```DStreamGraph``` is final.
69. Do you want to add the note here as well? ``` * @note The given AWS credentials will get saved in DStream checkpoints if checkpointing * is enabled. Make sure that your checkpoint directory is secure. ```
70. Done
71. nit: `Make sure that your checkpoint directory is secure. Prefer using the [https://link.to.amazon.docs default credential provider chain]] if possible`
72. The link in this case will be quite long-- URL just by itself pushes it over the 100 char limit: ```[[http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html#credentials-default default credential provider chain]]``` Do you know if there's a way to safely split this into multiple lines? Should I just turn style checks off for this comment?
73. Feel free to add ``` // scalastyle:off // scalastyle:on ``` around the doc
74. Done

jira_issues:

jira_issues_comments: