Integration with Kinesis

What is Kinesis?

- Out-of-the-box streaming tool from Amazon
- Designed to maintabe easier to maintain than Kafka, but operates under Amazon Software Licence (ASL).
- Uses “shards" as units of Streaming instead of “RDDs"

Steps to Using Kinesis In Applications

Linking

- For Python applications, you will have to add the below library and its dependencies when deploying an application

```
groupId = org.apache.spark
artifactId = spark-streaming-kinesis-asl_2.11
version = 2.2.0
```

- Full details are given in the Spark Programming guide at https://spark.apache.org/docs/2.2.0/streaming-programming-guide.html#linking
- ***Note that by linking to this library, you will include Amazon-licensed code in your application.***

Steps to Using Kinesis In Applications

Programming

- In the streaming application code, import `KinesisUtils` and `InitialPositionInStream` and create an output Dstream.
- You can also specify the key and value classes and their corresponding decoder classes using variations of createStream

```
from pyspark.streaming.kinesis import KinesisUtils,
      InitialPositionInStream


kinesisStream = KinesisUtils.createStream(streamingContext, [Kinesis app
      name], [Kinesis stream name], [endpoint URL], [region name],
      [initial position], [checkpoint interval],
      StorageLevel.MEMORY_AND_DISK_2)
```

Steps to Using Kinesis In Applications

Deployment

- `spark-submit` is used to launch application
- For Python applications which lack SBT/Maven project management, `spark-streaming-kinesis-asl_2.11` and its dependencies can be directly added to `spark-submit` using `--packages`

```
./bin/spark-submit --packages org.apache.spark:spark-streaming-kinesis-
      asl_2.11:2.2.0 ...
```

Things to Remember

- Kinesis data processing is ordered per partition and occurs at-least once per message.
- Multiple applications can read from the same Kinesis stream. Kinesis will maintain the application-specific shard and checkpoint info in DynamoDB.
- A single Kinesis stream shard is processed by one input DStream at a time.
- A single Kinesis input DStream can read from multiple shards of a Kinesis stream by creating multiple KinesisRecordProcessor threads.
- Multiple input DStreams running in separate processes/instances can read from a Kinesis stream.

Things to Remember (cont.)

- You never need more Kinesis input DStreams than the number of Kinesis stream shards.
- Horizontal scaling is achieved by adding/removing Kinesis input DStreams - up to the total number of Kinesis stream shards per the previous point.
- The Kinesis input DStream will balance the load between all DStreams - even across processes/instances.
- There is no correlation between the number of Kinesis stream shards and the number of RDD partitions/shards created across the Spark cluster during input DStream processing.