<html><head></head><body><pre style="word-wrap: break-word; white-space: pre-wrap;">from pyspark
import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.storagelevel import StorageLevel
from pyspark.streaming.kinesis import KinesisUtils, InitialPositionInStream
import datetime
import json
from pyspark.sql import SQLContext, Row
from pyspark.sql.types import *
import commands
import pytz
import time
originalTimeZone = "Etc/UTC"
targetTimeZone = "America/Los_Angeles"
current_timestamp=str(pytz.timezone(targetTimeZone).localize(datetime.datetime.fromtimestamp(tim
e.time())).astimezone(pytz.timezone(originalTimeZone)))
aws_region = 'us-east-1'
kinesis_stream = 'EurekaScreenSharingEventStream'
kinesis_endpoint = 'https://kinesis.us-east-1.amazonaws.com/'
kinesis_app_name = 'EurekaScreenSharingEventStreamAppTest'
kinesis_initial_position = InitialPositionInStream.TRIM_HORIZON
kinesis_checkpoint_interval = 10
spark_batch_interval = 10
spark_streaming_context = StreamingContext(SparkContext(), spark_batch_interval)
#spark_streaming_context = StreamingContext(sc, spark_batch_interval) when running from a
pyspark session
kinesis_stream =
KinesisUtils.createStream(spark_streaming_context,kinesis_app_name,kinesis_stream,kinesis_endpoi
nt,aws_region,kinesis_initial_position,kinesis_checkpoint_interval,StorageLevel.MEMORY_AND_DISK)
def process(time, rdd):
    return_code=commands.getstatusoutput('hadoop fs -rm -r
/user/mayanka/KinesisSparkIntegrationTest')
    rdd.saveAsTextFile("/user/mayanka/KinesisSparkIntegrationTest/")

ScreenSharingEvent=sqlContext.read.format('com.databricks.spark.xml').options(rowTag='ns2:screen
SharingEvent').load('/user/mayanka/KinesisSparkIntegrationTest/')
    print("============printSchema============")
    ScreenSharingEvent.printSchema()

SessionID_df=ScreenSharingEvent.select(ScreenSharingEvent["ns2:sessionId"]).withColumnRenamed("c
ol","sessionid")
    print("===========registertable===========")
    SessionID_df.registerTempTable("temp_session_id")
    SessionCount_df=sqlContext.sql("select count(*) as
SessionCount,date_format(current_timestamp(),'MM/dd/yyyy HH:mm:ss') as CurrentTimeStamp from
temp_session_id")
    print("==========writeToRedshift==========")
    SessionCount_df.write.format("com.databricks.spark.redshift").option("url",
"jdbc:redshift://redshiftpoc.avabi.expertcity.com:5439/dev?
user=mayankayush&password=wenK5DJnzfX4s3QL").option("dbtable","temp_sessions_counts").option
("tempdir", "s3a://citrixsaasdata-
dev/KinesisSparkIntegrationTest/").option("aws_iam_role","arn:aws:iam::984551231764:role/mapr-
qa").mode("append").save()

kinesis_stream.foreachRDD(process)
kinesis_stream.pprint()
spark_streaming_context.start()
spark_streaming_context.awaitTermination()

#pyspark --jars /usr/lib/spark/external/lib/spark-streaming-kinesis-asl-assembly_2.11-
2.1.0.jar,/usr/lib/spark/external/lib/amazon-kinesis-client-
1.7.4.jar,/usr/lib/spark/external/lib/RedshiftJDBC4-1.2.1.1001.jar --packages
com.databricks:spark-xml_2.11:0.4.1,com.databricks:spark-redshift_2.10:3.0.0-
preview1,com.databricks:spark-avro_2.11:3.2.0 --driver-cores 2


#spark-submit --jars /usr/lib/spark/external/lib/spark-streaming-kinesis-asl-assembly_2.11-
2.1.0.jar,/usr/lib/spark/external/lib/amazon-kinesis-client-
1.7.4.jar,/usr/lib/spark/external/lib/RedshiftJDBC4-1.2.1.1001.jar --packages
com.databricks:spark-xml_2.11:0.4.1,com.databricks:spark-redshift_2.10:3.0.0-
preview1,com.databricks:spark-avro_2.11:3.2.0 /home/mayanka/kinesis-spark-redshift.py --deploy-

mode cluster --driver-cores 2
</pre></body></html>