## Lab 6-01: Performing Real-Time Data Analysis with Amazon Kinesis

### Service Introduction

Amazon Kinesis is a platform on AWS that collects, processes, and analyzes real-time streaming data. It enables you to ingest large streams of data records in real-time, allowing for immediate data processing and analysis. This service is ideal for scenarios such as real-time analytics, machine learning, and data warehousing.

### Problem

Your business generates large volumes of data from various sources, such as application logs, social media feeds, and IoT devices. You need to analyze this data in real-time to gain immediate insights and make quick decisions, but traditional batch processing methods are too slow and inefficient.

### Solution

Use Amazon Kinesis to create a real-time data pipeline that ingests, processes, and analyzes streaming data. Kinesis Data Streams will collect the data, Kinesis Data Firehose will load it into storage or analytics services, and Kinesis Data Analytics will process it in real-time. This setup enables you to perform real-time data analysis, providing immediate insights and actionable intelligence to drive timely business decisions.

#### Task 1: Create an Amazon Kinesis Data Stream

|  |
| --- |
| 1. In the search bar on top, type **"Kinesis"** to search for the Amazon Kinesis service. 2. Click on the **Amazon Kinesis** result to go directly to the Kinesis service.      1. Click on the **Data streams** button in the sidebar. 2. Click on the **Create data stream** button to launch the wizard to create the Data Stream.      1. Enter the following details for the new Data Stream:  * Data stream name: **TelemetricsStream** * Capacity mode: **On-demand**      1. Click on the **Create data stream** button.      1. Confirm the creation of the new Data Stream. |

#### Task 2: Develop a Lambda Function to produce streaming events

|  |
| --- |
| 1. In the search bar on top, type **"Lambda"** to search for the AWS Lambda service. 2. Click on the Lambda result to go directly to the Lambda service on a new tab.      1. Click on the **Create function** button to create a new AWS Lambda function      1. Enter the following details for the new Lambda function:  * Function name: produceKinesisEvents * Runtime: Python 3.11 (or the latest)      1. Click the Change default execution drop toggle to change the execution role manually. 2. Change the radio button to Use an existing role. 3. Select the IAM role from the dropdown containing the name KinesisLambdaRole, which will provide the necessary privileges to the Lambda function. 4. Click on the **Create function** button to create the function.      1. Once the page has loaded, update the following code into the Lambda function:   import json  import boto3  def lambda\_handler(event, context):  client = boto3.client('kinesis')  data = {  "id": "0",  "latitude": "0",  "longtitude": "0"  }  response = client.put\_record(  StreamName="TelemetricsStream",  PartitionKey="geolocation",  Data=json.dumps(data)  )  return response   1. Click on the **Deploy** button to save and deploy the new version of the Lambda function.      1. Click on the **Test** button to prepare to test the functions execution.      1. Enter a short name for the test event (like test), and click on the **Save** button to save the test event. 2. Click on the **Test** button again to trigger the test.      1. Confirm that the execution results returned with a HTTPStatusCode of 200.      1. If your function returned an error, check the execution role you set when creating your Lambda function. Since the Lambda function sends records to the Kinesis Data Stream, it needs extra permissions; otherwise, it will trigger an error. 2. Return to the Lambda function code and update it with the following to ensure it returns a continuous stream of events:   import json  import boto3  import uuid  import random  import time  def lambda\_handler(event, context):  client = boto3.client('kinesis')  while True:  data = {  "id": str(uuid.uuid4()),  "latitude": random.uniform(-90, 90),  "longtitude": random.uniform(0, 180)  }    response = client.put\_record(  StreamName="TelemetricsStream",  PartitionKey="geolocation",  Data=json.dumps(data)  )  print(response)  time.sleep(random.random())     1. Navigate to the **Configuration** tab of your Lambda Function, and ensure the General configuration sidebar tab is selected. 2. Click on the **Edit** button to change the configuration settings of your Lambda function.      1. Change the Timeout value to 30 seconds, then click on the **Save** button to update the function configuration.      1. Test the function again, and ensure it runs for 30 seconds before timing out. |

#### Task 3: Develop a Lambda Function to consume streaming events

|  |
| --- |
| 1. Open a new tab on the **Lambda Functions** page. 2. Click on the **Create function** button to create a new AWS Lambda function.      1. Enter the following details for the new Lambda function:  * Function name: **consumeKinesisEvents** * Runtime: **Python 3.11 (or the latest)**      1. Click the **Change default execution** drop toggle to manually change the execution role. 2. Change the radio button to Use an existing role. 3. Select the IAM role from the dropdown containing the name KinesisLambdaRole, which will provide the necessary privileges to the Lambda function. 4. Click on the **Create function** button to create the function.      1. Once the page has loaded, click on the **Add trigger** button.      1. From the dropdown, select Kinesis. 2. Enter the following details:  * Kinesis stream: **kinesis/TelemetricsStream** * Batch size: **10**      1. Click on the **Add** button to confirm the creation of the new trigger.      1. Update the following code into the Lambda function:   import json  def lambda\_handler(event, context):  print(json.dumps(event))     1. Click on the **Deploy** button to save and deploy the new version of the Lambda function. 2. Return to the tab containing the **produceKinesisEvents** function.      1. Click on the **Test** button again to trigger the test.      1. Return to the tab containing the consumeKinesisEvents function. 2. Navigate to the Monitoring tab of the Lambda function. 3. Click on the **View CloudWatch logs** button to launch CloudWatch Logs in a new tab.      1. Open the current log stream.      1. Verify that the records are present, with the data encoded in Base64.      1. Return to the tab containing the **consumeKinesisEvents** function. 2. Update the following code into the Lambda function:   import json  import base64  def lambda\_handler(event, context):  records = []    for record in event["Records"]:  data = base64.b64decode(record["kinesis"]["data"]).decode()  records.append(json.loads(data))    output = {  "count": str(len(records)),  "data": records  }    print(json.dumps(output))   1. Click on the **Deploy** button to save and deploy the new version of the Lambda function.      1. Return to the tab containing the **produceKinesisEvents** function. 2. Click on the **Test** button again to trigger the test.      1. Return to the tab containing the **consumeKinesisEvents** function. 2. Navigate to the **Monitoring** tab of the Lambda function. 3. Click on the **View CloudWatch logs** button to launch CloudWatch Logs in a new tab.      1. Open the current log stream.      1. Verify that the records are present, with the data decoded in JSON format containing the various GPS coordinates. |

#### Task 4: Setup Kinesis Data Firehose to send events to an S3 Bucket

|  |
| --- |
| 1. In the search bar on top, type "Kinesis" to search for the Amazon Kinesis service. 2. Click on the Amazon Kinesis result to go directly to the Kinesis service. 3. Click on the Data Firehose button in the sidebar. 4. Click on the **Create delivery stream** button to launch the wizard to create the Data Firehose delivery stream.      1. Enter the following details for the new Data Stream: 2. Source: **Amazon Kinesis Data Streams** 3. Destination: **Amazon S3**      1. Under Kinesis data stream, use Browse to find your **TelemetricsStream**      1. Under S3 bucket, Browse to find your firehosedeliverybucket      1. Set Buffer interval to **60** seconds      1. Click on the **Create Firehose stream** button to confirm the creation of the new Data Firehose.      1. Return to the tab containing the **produceKinesisEvents** function.      1. Navigate to the **Configuration** tab of your Lambda Function, and ensure the General configuration sidebar tab is selected. 2. Click on the **Edit** button to change the configuration settings of your Lambda function.      1. Change the Timeout value to 5 minutes, then click on the **Save** button to update the function configuration. 2. Click on the **Test** button again to trigger the test.      1. In the search bar on top, type "**S3**" to search for the Amazon S3 service. 2. Click on the **Amazon S3** result to go directly to the S3 service.      1. Open the **firehosedeliverybucket** in S3. 2. Wait up to a minute or so for the first events to arrive 3. Navigate through the date-based folders to find the delivered events |