

## Step 1: Connect to AWS Account with Account ID:

```
.9/site-packages (from boto==1.35.76->awscli) (1.26.20)
Requirement already satisfied: pyasn1>=0.1.3 in ./micromamba/lib/python3.9/site-
packages (from rsa<4.8,>=3.1.2->awscli) (0.5.1)
Requirement already satisfied: six>=1.5 in ./micromamba/lib/python3.9/site-packa
ges (from python-dateutil<3.0.0,>=2.1->boto==1.35.76->awscli) (1.16.0)
Downloading awscli-1.36.17-py3-none-any.whl (4.5 MB)
----- 4.5/4.5 MB 5.5 MB/s eta 0:00:00
Downloading docutils-0.16-py2.py3-none-any.whl (548 kB)
----- 548.2/548.2 kB 5.2 MB/s eta 0:00:00
Downloading rsa-4.7.2-py3-none-any.whl (34 kB)
Installing collected packages: rsa, docutils, awscli
  Attempting uninstall: rsa
    Found existing installation: rsa 4.9
    Uninstalling rsa-4.9:
      Successfully uninstalled rsa-4.9
Successfully installed awscli-1.36.17 docutils-0.16 rsa-4.7.2
(base) santosh@Santoshs-MacBook-Air ~ % aws --version
aws-cli/1.36.17 Python/3.9.18 Darwin/22.4.0 boto/1.35.76
(base) santosh@Santoshs-MacBook-Air ~ % aws configure

AWS Access Key ID [None]: 034362057519
AWS Secret Access Key [None]: *****
Default region name [None]: us-east-1
Default output format [None]: json
```

## Step 2: Run the code “producer.py” and sent the real time data to AWS:

```
ased policy allows the kinesis:PutRecord action
(base) santosh@Santoshs-MacBook-Air CL TASK1 % python producer.py
Sending data: {'sensor_id': 60, 'temperature': 20.55, 'humidity': 39.24, 'timestamp': 1733588110.66237}
Sending data: {'sensor_id': 23, 'temperature': 20.67, 'humidity': 55.26, 'timestamp': 1733588117.967923}
Sending data: {'sensor_id': 32, 'temperature': 22.52, 'humidity': 42.79, 'timestamp': 1733588119.3801322}
Sending data: {'sensor_id': 44, 'temperature': 29.59, 'humidity': 54.49, 'timestamp': 1733588120.7543418}
Sending data: {'sensor_id': 72, 'temperature': 23.87, 'humidity': 48.68, 'timestamp': 1733588122.1416478}
Sending data: {'sensor_id': 9, 'temperature': 29.99, 'humidity': 53.86, 'timestamp': 1733588123.484572}
Sending data: {'sensor_id': 95, 'temperature': 25.76, 'humidity': 47.59, 'timestamp': 1733588124.907493}
Sending data: {'sensor_id': 67, 'temperature': 27.93, 'humidity': 37.89, 'timestamp': 1733588126.346263}
Sending data: {'sensor_id': 48, 'temperature': 28.57, 'humidity': 52.79, 'timestamp': 1733588127.7768378}
Sending data: {'sensor_id': 76, 'temperature': 26.9, 'humidity': 33.56, 'timestamp': 1733588129.2105548}
Sending data: {'sensor_id': 84, 'temperature': 25.29, 'humidity': 32.66, 'timestamp': 1733588130.642389}
Sending data: {'sensor_id': 36, 'temperature': 29.3, 'humidity': 33.45, 'timestamp': 1733588132.0775309}
```

## Step 3: Login to AWS and step into kenosis and create a lamb function to get the input data stream and store it in a S3 bucket

The screenshot shows the AWS Lambda console for the **KinesisProcessor** function. The **Code source** tab is active, displaying the `lambda_function.py` file in the console editor. The code imports `json` and `boto3`, initializes an `s3_client`, and defines a `lambda_handler` function that processes Kinesis records and saves them to an S3 bucket named `my-transformed-data-bucket`.

```
1 import json
2 import boto3
3
4 s3_client = boto3.client('s3')
5 BUCKET_NAME = 'my-transformed-data-bucket'
6
7 def lambda_handler(event, context):
8     transformed_data = []
9
10    for record in event['Records']:
11        # Decode and parse Kinesis record
12        payload = json.loads(record['kinesis']['data'])
13        print(f"Received payload: {payload}")
14
15        # Transform: Add geo-location (simulated example)
16        payload['geo_location'] = f"Lat-{payload['sensor_id']},Lon-{payload['sensor_id']}"
17        transformed_data.append(payload)
18
19    # Save transformed data to S3
20    filename = f"transformed_data_{int(time.time())}.json"
21    s3_client.put_object(
22        Bucket=BUCKET_NAME,
23        Key=filename,
24        Body=json.dumps(transformed_data)
25    )
26    print(f"Saved transformed data to S3: {filename}")
27
```

The left sidebar shows the **EXPLORER** view with the **KINESISPROCESSOR** folder expanded, showing `lambda_function.py`. Below it, the **DEPLOY [UNDEPLOYED CHANGES]** section indicates that there are undeployed changes and provides buttons for **Deploy** and **Test**. The **TEST EVENTS** section shows that no test events have been created yet, with a button to **Create test event**.

On the right, the **Info** and **Tutorials** tabs are visible. The **Tutorials** tab contains a section titled **Create a simple web app** with a list of steps: 

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

 and a **Start tutorial** button.

The screenshot shows the AWS Lambda console for the **KinesisProcessor** function, displaying the **Execution Results** for a test event named `KinesisTestEvent. The lambda_function.py` code is visible in the background, showing the `lambda_handler` function.

The **Execution Results** section shows the following details:

- Status:** Succeeded
- Test Event Name:** KinesisTestEvent
- Response:**

```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```
- Function Logs:**

```
START RequestId: c467a47e-c133-488e-8abb-71f8b6c93c56 Version: $LATEST
END RequestId: c467a47e-c133-488e-8abb-71f8b6c93c56
REPORT RequestId: c467a47e-c133-488e-8abb-71f8b6c93c56 Duration: 1.92 ms Billed Duration: 2 ms Memory Size: 128 MB
Max Memory Used: 30 MB
```
- Request ID:** c467a47e-c133-488e-8abb-71f8b6c93c56

The left sidebar shows the **EXPLORER** view with the **KINESISPROCESSOR** folder expanded, showing `lambda_function.py`. Below it, the **DEPLOY [UNDEPLOYED CHANGES]** section indicates that there are undeployed changes and provides buttons for **Deploy** and **Test**. The **TEST EVENTS** section shows that a test event named `KinesisTestEvent` has been created.

On the right, the **Info** and **Tutorials** tabs are visible. The **Tutorials** tab contains a section titled **Create a simple web app** with a list of steps: 

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

 and a **Start tutorial** button.

Navigation: Lambda > Functions > KinesisProcessor

### KinesisProcessor

Throttle Copy ARN Actions

The trigger sensor-datastream was successfully added to function KinesisProcessor. The trigger is in a disabled state.

#### Function overview

Diagram Template

KinesisProcessor

Layers (0)

Kinesis

+ Add trigger

+ Add destination

Export to Infrastructure Composer Download

Description

Last modified 35 minutes ago

Function ARN arn:aws:lambda:us-east-1:034362057519:function:KinesisProcessor

Function URL Info

Code Test Monitor Configuration Aliases Versions

#### Code source

Upload from

You are using the new console editor. Switch to the old editor

EXPLORER

KINESISPROCESSOR

lambda\_function.py

lambda\_function.py

27

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

#### Info

#### Tutorials

Learn how to implement common use cases in AWS Lambda.

#### Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Learn more

Start tutorial

## Step 4: Create a Database and table inside it to store the transformed data in S3 buckets, set an IAM and edit the manage preferences

Navigation: AWS Glue > Crawlers > samplecrawler

### samplecrawler

Last updated (UTC) December 7, 2024 at 16:39:23 Run crawler Edit Delete

#### Crawler properties

Name samplecrawler	IAM role AWSGlueServiceRole-testrole	Database transformeddatadb	State READY
Description	Security configuration	Lake Formation configuration	Table prefix
Maximum table threshold			

Advanced settings

#### Crawler runs (1)

The list of crawler runs for this crawler.

Filter data Filter by a date and time range

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
December 7, 2024 at 16:39:31	-	01 min 03 s	Running	-	-

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Step 5: Now Step into to Amazon Athena and connect the Database and start querying to retrieve data (Data Analysis)

The screenshot displays the Amazon Athena Query Editor interface in a web browser. The browser's address bar shows the URL: `us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#query-editor/history/07ca2be0-ceac-4428-809b-08ba816b9be4`. The interface includes a top navigation bar with the AWS logo and a search bar. Below this, a banner message states: "Athena now supports typeahead text suggestions to speed up query development. Typeahead suggestions are turned on by default. You can change this setting in query editor preferences." The main workspace is divided into three sections. On the left, the "Data" sidebar contains dropdown menus for "Data source" (set to "AwsDataCatalog"), "Catalog" (set to "None"), and "Database" (set to "transformeddatadb"). Below these are "Tables and views" with a "Create" button and a search bar. A list of tables shows "testcrawler" under the "Tables (1)" section. The central area is the "Query editor" where "Query 2" is active. It contains a SQL query: 

```
1 SELECT sensor_id, COUNT(*) as event_count
2 FROM "AwsDataCatalog"."transformeddatadb"."testcrawler"
3 GROUP BY sensor_id
4 ORDER BY event_count DESC
5 LIMIT 10;
```

 Below the query editor are buttons for "Run again", "Explain", "Cancel", "Clear", and "Create". To the right of these buttons, it indicates "Reuse query results up to 60 minutes ago". Below the query editor, the "Query results" tab is selected, showing a status bar with "Completed", "Time in queue: 75 ms", "Run time: 372 ms", and "Data scanned: 0.55 KB". There are "Copy" and "Download results" buttons. The results table has a search bar and shows one row with columns "sensor\_id" and "event\_count". The "event\_count" value is 24. The footer of the page includes "CloudShell", "Feedback", and copyright information for Amazon Web Services, Inc. or its affiliates, along with links for "Privacy", "Terms", and "Cookie preferences".

**Data**

Data source: AwsDataCatalog

Catalog: None

Database: transformeddatadb

Tables and views: Create

Filter tables and views

Tables (1): testcrawler

Views (0)

**Query 2**

```
1 SELECT sensor_id, COUNT(*) as event_count
2 FROM "AwsDataCatalog"."transformeddatadb"."testcrawler"
3 GROUP BY sensor_id
4 ORDER BY event_count DESC
5 LIMIT 10;
```

Run again Explain Cancel Clear Create

Reuse query results up to 60 minutes ago

**Query results** Query stats

Completed Time in queue: 75 ms Run time: 372 ms Data scanned: 0.55 KB

Copy Download results

Search rows

#	sensor_id	event_count
1		24

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 6: use QuickSight for analysis the datasets from Athena and connect to the database

