



AWS SOLUTION ARCHITECT COURSE-END PROJECT

ASHUTOSH SRIVASTAVA

I2ASHUI2@GMAIL.COM

[WWW.LINKEDIN.COM/IN/ASHUTOSH-SRIVASTAVA-I2ASHUI2](https://www.linkedin.com/in/ashutosh-srivastava-i2ashui2)

PROBLEM STATEMENT AND MOTIVATION

CONTENT

- Real-World Scenario:
TELEMAX is a company expanding its network into underserved markets globally.
They provide innovative communications hardware for efficient networking links.
- Need:
Deploy an effective NoSQL-based data warehousing architecture.
Handle real-time data for future analysis to optimize network topologies.
- Solution Approach:
Utilize AWS cloud services to build a real-time data management system.

SUMMARY OF STEPS

Step 1: Create AWS Kinesis Data Stream

Setup Kinesis stream to handle incoming data.

Step 2: Create AWS Lambda Function for Data Ingestion

Develop Lambda function to pull data from a URL and stream it to Kinesis.

Step 3: Add Python Layer to Lambda Function

Include necessary libraries like requests in a Lambda layer.

Step 4: Deploy and Verify Kinesis Data Stream

Check if data is streaming correctly using Kinesis Data Viewer.

Step 5: Create Lambda Function for Data Loading

Develop Lambda function to read from Kinesis and write to DynamoDB.

Step 6: Configure Permissions for Lambda Functions

Assign appropriate roles and permissions to Lambda functions.

Step 7: Monitor CloudWatch Logs

Review logs to ensure Lambda functions are running correctly and debug issues.

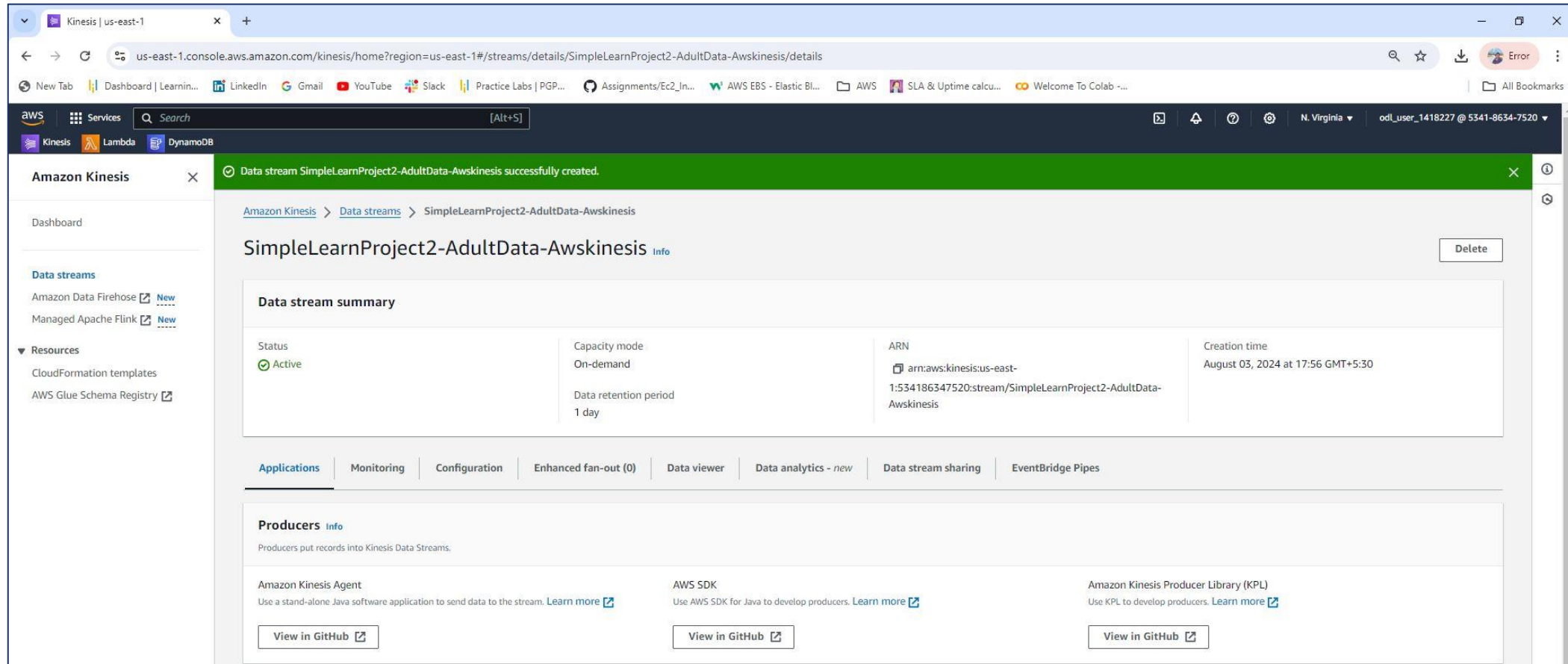
Step 8: Verify Data in DynamoDB

Check DynamoDB to confirm data is loaded and accessible.

STEP 1 : AWS KINESIS DATA STREAM CREATION

Created AWS Kinesis Data Stream named: SimpleLearnProject2-AdultData-Awskinesis

- Screenshot: [Include screenshot of Kinesis Data Stream creation]



STEP 2: CREATE PYTHON CODE ON AWS LAMBDA FUNCTION

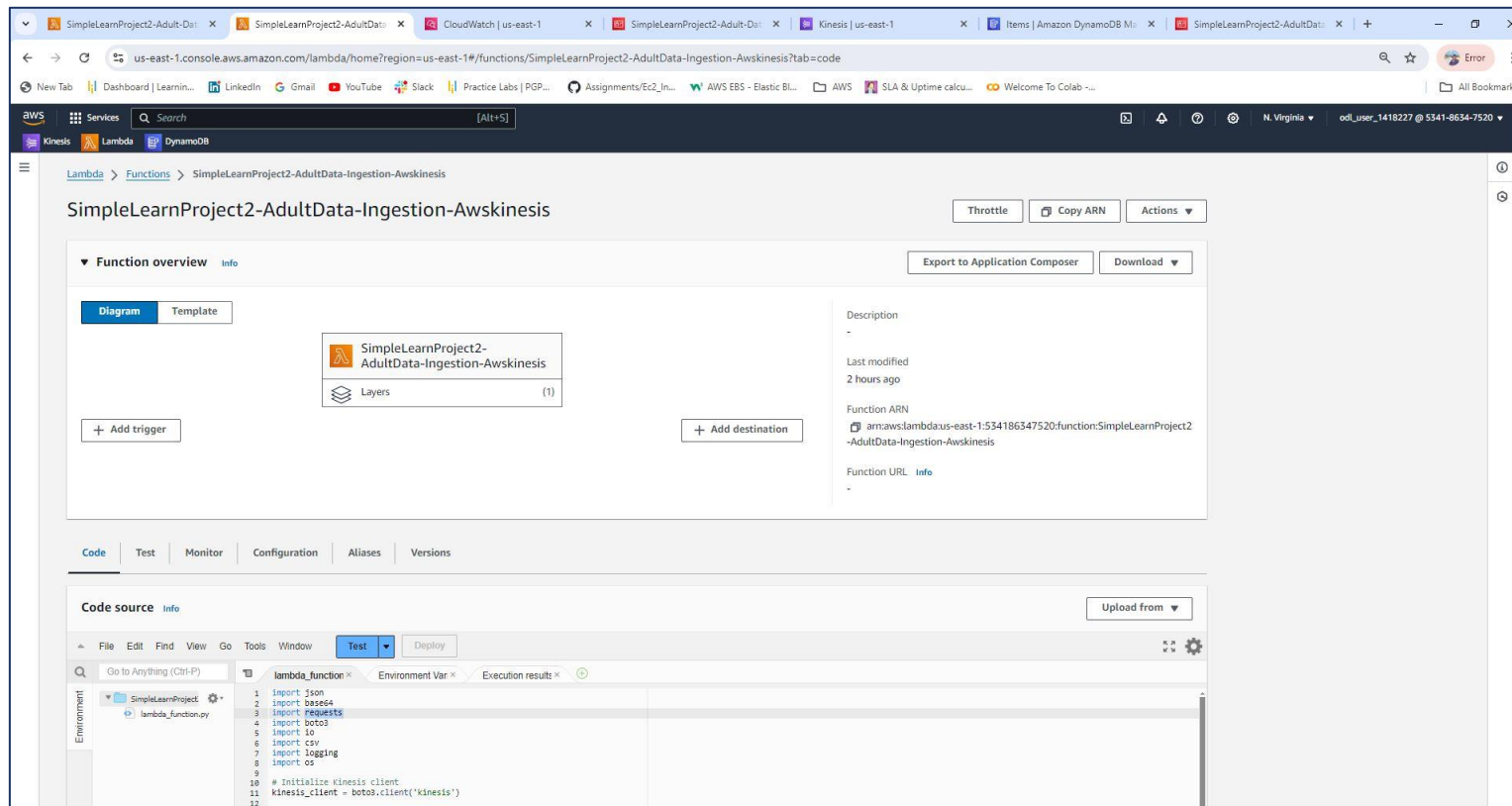
Lambda function to ingest data from URL:

<https://archive.ics.uci.edu/ml/machine-learningdatabases/adult/adult.data>

Details:

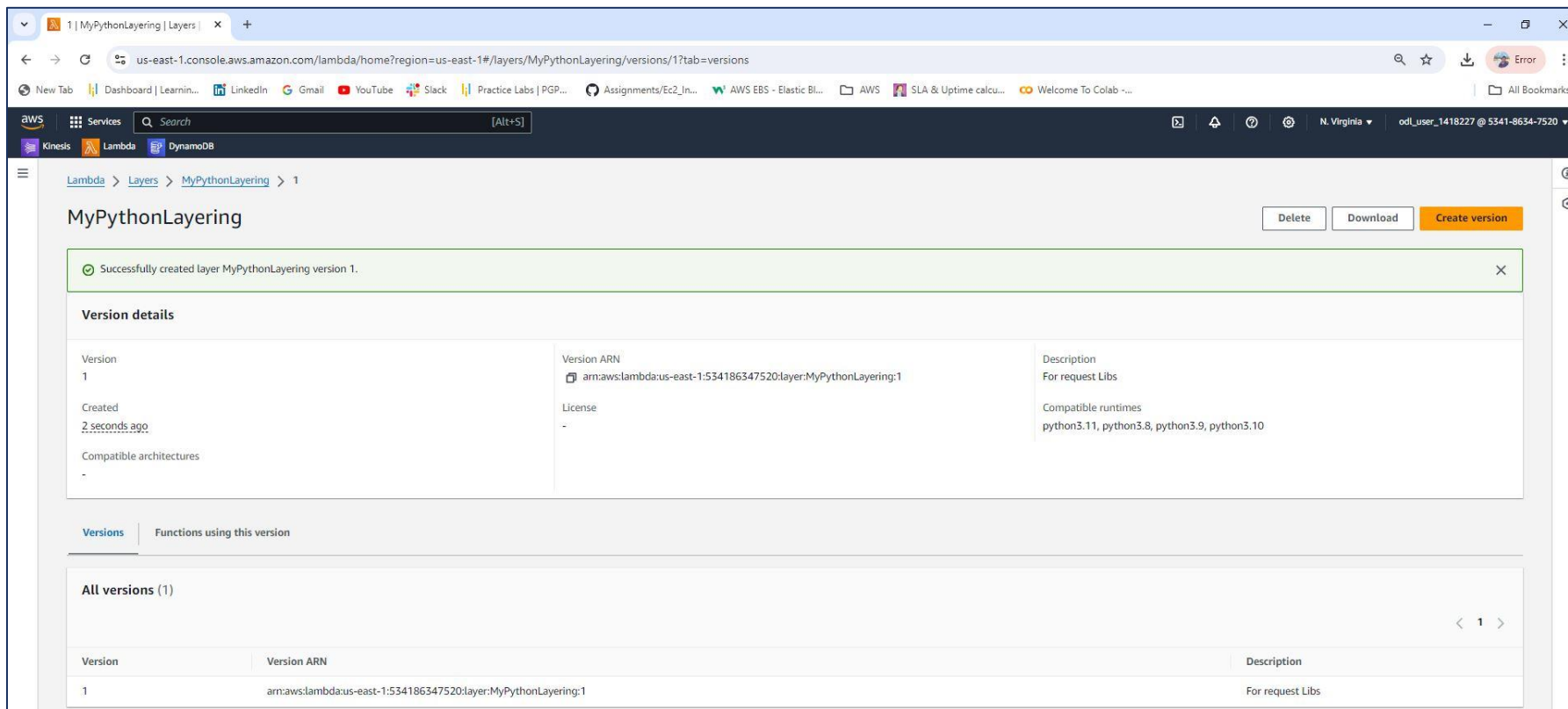
Function processes data and streams it to Kinesis

Screenshot: [Include screenshot of Lambda function code]



STEP 3: AWS LAMBDA FUNCTION WITH PYTHON LAYER

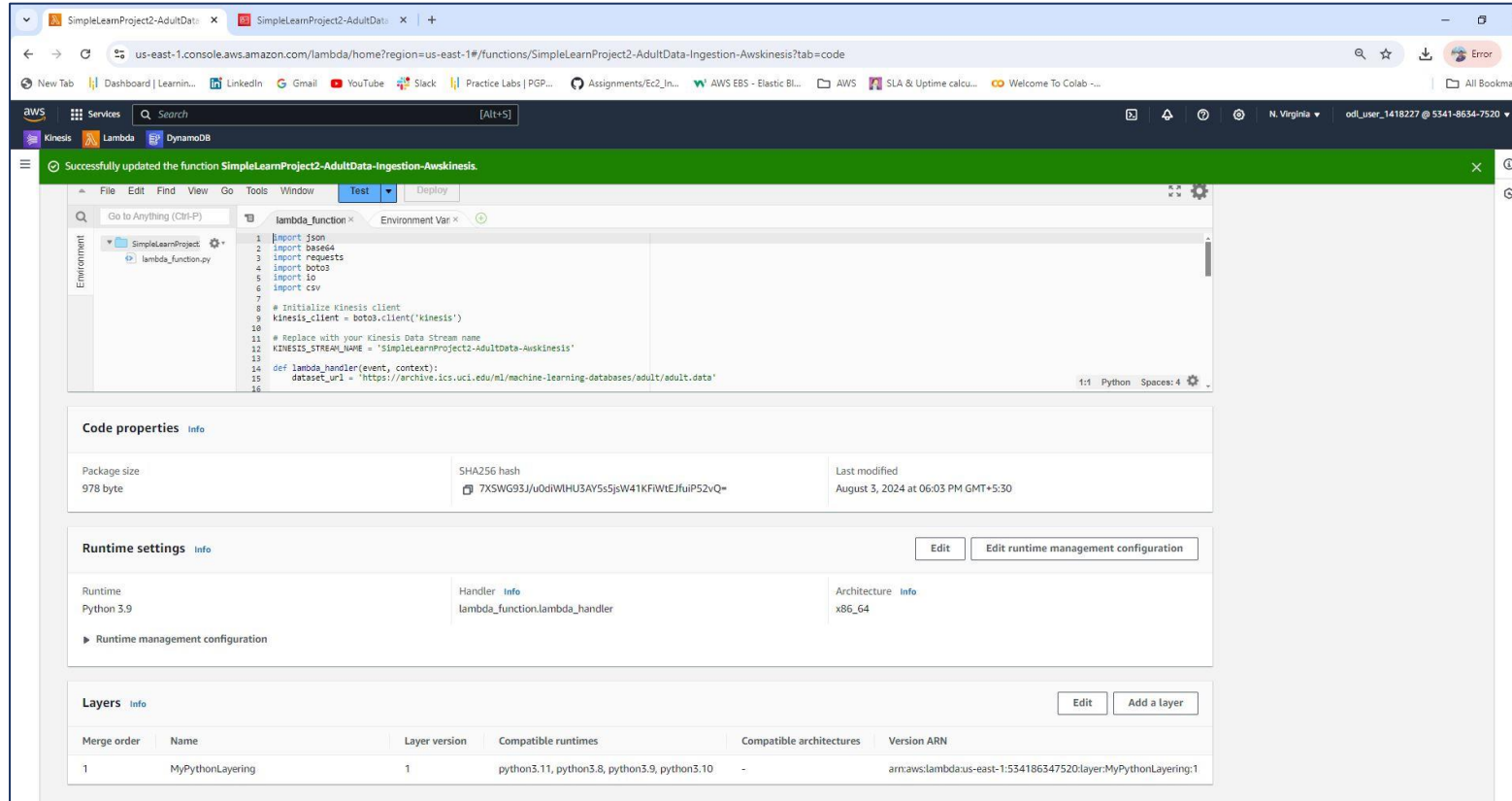
- Lambda function name: SimpleLearnProject2-AdultData-Ingestion-Awskinesis
- Added Python layer for requests library (not available by default)
- Screenshots:



STEP 3: AWS LAMBDA FUNCTION WITH PYTHON LAYER

CONTINUE

- Layer added to Lambda Function
- Screenshots:



Successfully updated the function SimpleLearnProject2-AdultData-Ingestion-AwsKinesis.

```
1 import json
2 import base64
3 import requests
4 import boto3
5 import io
6 import csv
7
8 # Initialize Kinesis client
9 kinesis_client = boto3.client('kinesis')
10
11 # Replace with your Kinesis Data Stream name
12 KINESIS_STREAM_NAME = 'SimpleLearnProject2-AdultData-AwsKinesis'
13
14 def lambda_handler(event, context):
15     dataset_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data'
16
```

Code properties

Package size 978 bytes	SHA256 hash 7XSWG93JJU0diWHU3AY5s5jsW41KFIWEJfuiP52vQ=	Last modified August 3, 2024 at 06:03 PM GMT+5:30
---------------------------	---	--

Runtime settings

Runtime Python 3.9	Handler lambda_function.lambda_handler	Architecture x86_64
-----------------------	---	------------------------

Layers

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
1	MyPythonLayering	1	python3.11, python3.8, python3.9, python3.10	-	arn:aws:lambda:us-east-1:534186347520:layer:MyPythonLayering:1

STEP 4: DEPLOY & VERIFY KINESIS DATA STREAM

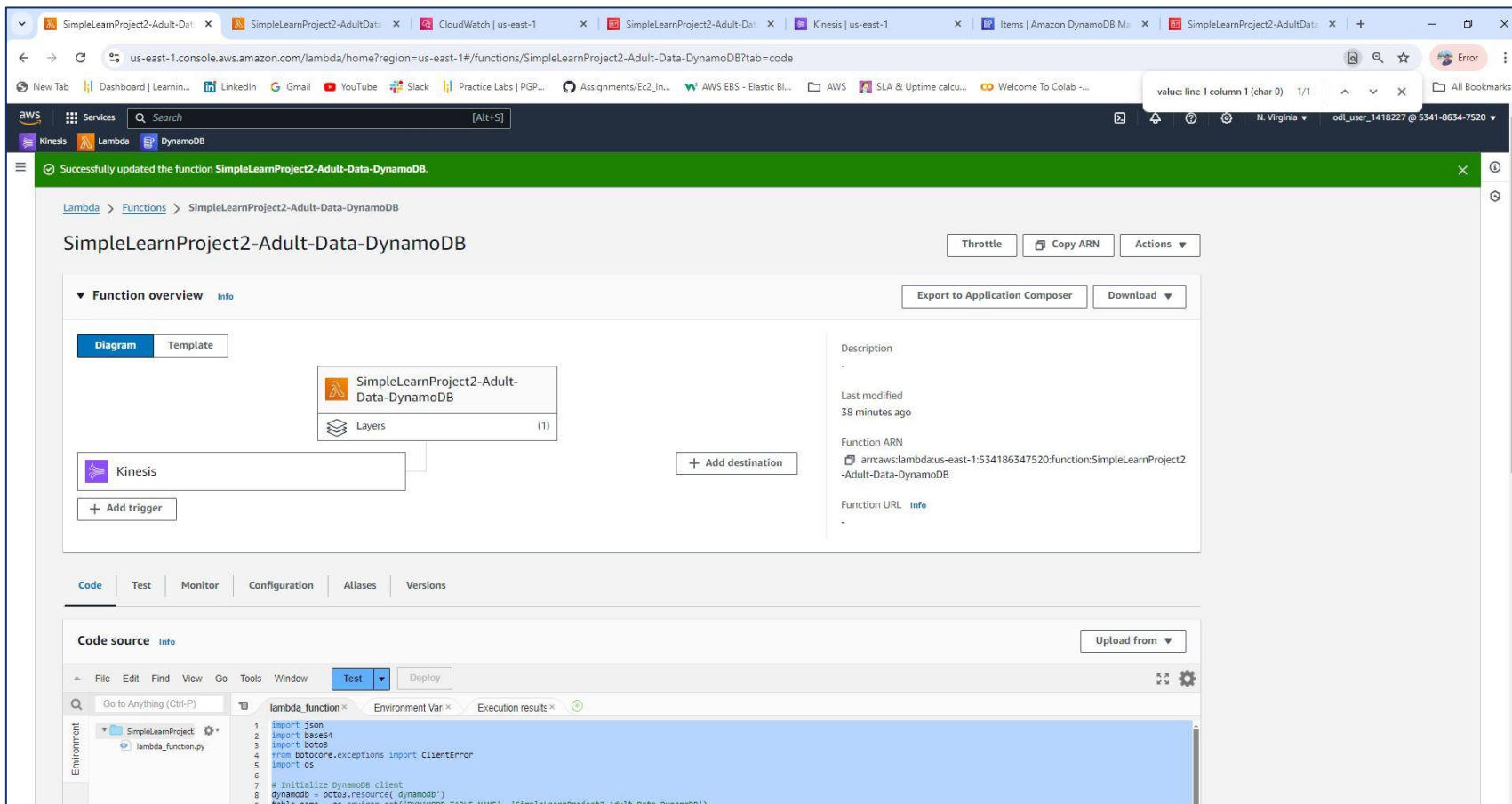
- Deployed Lambda function and verified data streaming using Kinesis Data Viewer

The screenshot displays the AWS Kinesis console for the data stream 'SimpleLearnProject2-Adult-Data-Awskinesis'. A green banner at the top indicates the stream was successfully created. The 'Data stream summary' shows the stream is 'Active' with an 'On-demand' capacity mode and a 1-day data retention period. The 'Data viewer' tab is active, showing a list of records. The records are filtered by the partition key 'Adult' and the starting position is set to 'Trim horizon'.

Partition key	Data	Approximate arrival timestamp	Sequence number
Adult	WylzOSlslClgU3RhGUKz292liwglIA3NzUxNlslClgQmFjaGVsb3JzliwglIAxM...	August 03, 2024 at 18:14:30 GMT+5:30	49654533283212540466227229883041312522442951080164720658
Adult	Wyl1MClslClgU2VzZ11bXAtbm90LWluYyIsClgODMzMTEILCAIEJHY2hibG9...	August 03, 2024 at 18:14:30 GMT+5:30	49654533283212540466227229883042521448262565709339426834
Adult	WylzOCslslClgUHJpdmF0ZSIsClgMjE1NjQ2liwglIBUy1ncmFklwglIA5liwglI...	August 03, 2024 at 18:14:30 GMT+5:30	49654533283212540466227229883043730374082180338514133010
Adult	Wyl1MylslClgUHJpdmF0ZSIsClgMjM0NzliwglIAxMXR0liwglIA3liwglIBNY...	August 03, 2024 at 18:14:30 GMT+5:30	49654533283212540466227229883044939299901794967688839186

STEP 5: AWS LAMBDA FUNCTION FOR DATA LOADING

- Created Lambda function to read from Kinesis and load data into DynamoDB Lambda function name: SimpleLearnProject2-Adult-Data-DynamoDB-LambdaFunction



STEP 6: ADD PERMISSIONS TO LAMBDA FUNCTIONS

- Assigned necessary permissions for both Lambda functions, Below for Ingestion AWS Kinesis

The screenshot displays the AWS IAM console interface. A green notification banner at the top states "Policy was successfully attached to role." The main heading is "SimpleLearnProject2-AdultData-Ingestion-Awskinesis-role-22n4m5vm". The "Summary" section shows the role's creation date as August 03, 2024, 17:57 (UTC+05:30) and its ARN as arn:aws:iam::534186347520:role/service-role/SimpleLearnProject2-AdultData-Ingestion-Awskinesis-role-22n4m5vm. The "Permissions" tab is active, showing two attached policies: "AmazonKinesisFullAccess" (AWS managed) and "AWSLambdaBasicExecutionRole-bcc46902-d9f8-4cdc-b0da-2af97f9181..." (Customer managed). The "Permissions boundary" is noted as "not set".

Identity and Access Management (IAM)

Policy was successfully attached to role.

SimpleLearnProject2-AdultData-Ingestion-Awskinesis-role-22n4m5vm

Summary

Creation date
August 03, 2024, 17:57 (UTC+05:30)

ARN
arn:aws:iam::534186347520:role/service-role/SimpleLearnProject2-AdultData-Ingestion-Awskinesis-role-22n4m5vm

Last activity
-

Maximum session duration
1 hour

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (2)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonKinesisFullAccess	AWS managed	1
AWSLambdaBasicExecutionRole-bcc46902-d9f8-4cdc-b0da-2af97f9181...	Customer managed	1

Permissions boundary (not set)

Generate policy based on CloudTrail events

STEP 6: ADD PERMISSIONS TO LAMBDA FUNCTIONS

- Assigned necessary permissions for both Lambda functions, Below for Ingestion AWS DynamoDB

The screenshot displays the AWS IAM console interface. The left sidebar shows the 'Identity and Access Management (IAM)' menu with options like Dashboard, Access management, Access reports, and Related consoles. The main content area shows the details for the role 'SimpleLearnProject2-Adult-Data-DynamoDB-role-nhlowc3z'. The 'Summary' section includes the creation date (August 03, 2024, 18:18 UTC+05:30), last activity (29 minutes ago), ARN, and maximum session duration (1 hour). The 'Permissions' tab is selected, showing a list of permissions policies. The table below lists the attached policies:

Policy name	Type	Attached entities
AmazonDynamoDBFullAccess	AWS managed	1
AmazonKinesisReadOnlyAccess	AWS managed	1
AWSLambdaBasicExecutionRole-52d71170-38f0-4f0e-a5b4-a51748c069...	Customer managed	1

Below the table, there is a section for 'Permissions boundary' (not set) and a link to 'Generate policy based on CloudTrail events'.

STEP 8: VERIFY DATA IN DYNAMODB TABLE

- Verified data was successfully loaded into DynamoDB table SimpleLearnProject2-Adult-Data-DynamoDB

The screenshot shows the AWS Management Console interface for the DynamoDB table SimpleLearnProject2-Adult-Data-DynamoDB. The table is located in the us-east-1 region. The console displays the table's metadata and a list of 54 items returned from a scan operation. The items are displayed in a table format with columns for record_id, age, capital_gain, capital_loss, education, education_num, fnlgt, hours_per_week, income, marital_status, native_country, occupation, race, and relationship.

record_id (String)	age	capital_gain	capital_loss	education	education_num	fnlgt	hours_per_week	income	marital_status	native_country	occupation	race	relationship
52_209842_HS-grad	52	0	0	HS-grad	9	209842	45	>50K	Married-civ-spouse	United-States	Exec-manag...	White	Husband
35_76845_9th	35	0	0	9th	5	76845	40	<=50K	Married-civ-spouse	United-States	Farming-fish...	Black	Husband
37_280464_Some-colli...	37	0	0	Some-college	10	280464	80	>50K	Married-civ-spouse	United-States	Exec-manag...	Black	Husband
24_172987_Bachelors	24	0	0	Bachelors	13	172987	50	<=50K	Married-civ-spouse	United-States	Tech-support	White	Husband
19_168294_HS-grad	19	0	0	HS-grad	9	168294	40	<=50K	Never-married	United-States	Craft-repair	White	Own-child
40_193524_Doctorate	40	0	0	Doctorate	16	193524	60	>50K	Married-civ-spouse	United-States	Prof-specialty	White	Husband
39_215646_HS-grad	39	0	0	HS-grad	9	215646	40	<=50K	Divorced	United-States	Handlers-cle...	White	Not-in-family
31_507875_9th	31	0	0	9th	5	507875	43	<=50K	Married-civ-spouse	United-States	Machine-op-...	White	Husband
44_128354_Masters	44	0	0	Masters	14	128354	40	<=50K	Divorced	United-States	Exec-manag...	White	Unmarried
45_386940_Bachelors	45	0	1408	Bachelors	13	386940	40	<=50K	Divorced	United-States	Exec-manag...	White	Own-child

CONCLUSION

- Successfully deployed a real-time data management solution on AWS.

Achievements:

- Streamed data using AWS Kinesis
- Processed and loaded data into DynamoDB using Lambda functions Verified data flow and integrity



THANK YOU

I2ASHUI2@GMAIL.COM

[WWW.LINKEDIN.COM/IN/ASH
UTOSH-SRIVASTAVA-I2ASHUI2](https://www.linkedin.com/in/ashutosh-srivastava-i2ashui2)

