Serverless AWS Project:

# Building a CRUD Serverless API with AWS Lambda, API Gateway, and a DynamoDB with cloudwatch and SNS notifications.

# Problem Statement:

-A client has built a user registration web application for a company. The front-end is written in JavaScript and has to function with web API; while the backend is written in Python. The client wants this application to be deployed on AWS using serverless service options for the backend. Also, the client wants to monitor the application and get notified in case they are errors or too much traffic.

# Solution Statement:

I am going to create an API with endpoints so that the frontend can use it, and also, create a lambda to manage the request from the API gateway and a dynamodb to store data. The lambda function will also communicate with the dynamodb. To store the frontend JavaScript code, I am going to create an Amazon S3 bucket to hold the code and an Amazon CloudFront distribution to serve the website globally; while Amazon Route 53 will manage the domain name, and AWS Certificate Manager will provide a valid SSL/TLS certificate. Lastly, AWS Cloudwatch is going to manage the logs.

# Some Endpoints for the project:

-Health check: This endpoint will be used to study the applications' health. It has a GET method to return a 200 response when called.

-User: This endpoint will support GET, POST, DELETE, and PUT methods. It will be used to create, modify, delete, or display users' information.

-Users: This endpoint will be used to list all the users in the database.

# Services to be used:

-AWS Lambda

-AWS API Gateway

-DynamoDB

-Amazon S3

-Amazon CloudFront

-Amazon Route 53

-AWS Certificate Manager

- AWS SNS

- AWS Cloudwatch

# Lambda function description:

-The Lambda function is going to create, reads, updates, and deletes from DynamoDB. Also, the function uses events from API Gateway to determine how to interact with DynamoDB.

# Creating a DynamoDB database

Graphical user interface, text, application, email

Description automatically generated

# Creating an IAM role so that the lambda function will use to access the database

Graphical user interface, text, application, email

Description automatically generated

# Role permissions

Table

Description automatically generated

# Creating a Lambda function

Graphical user interface, text, application, email

Description automatically generated

# Files for the Lambda process

Graphical user interface, application

Description automatically generated

# Create and Configure the API gateway

Graphical user interface, text, application, email

Description automatically generated

# Create various endpoints for the API

Graphical user interface, text, application

Description automatically generated

# After Deploying the API, now it’s time to host the JavaScript application on S3

Graphical user interface, text, application, email

Description automatically generated

# Configuring the S3 bucket to host the static website

Graphical user interface, application

Description automatically generated

# S3 static website URL

Graphical user interface, text, application, email

Description automatically generated

# Accessing the website with the S3 URL

Graphical user interface, table

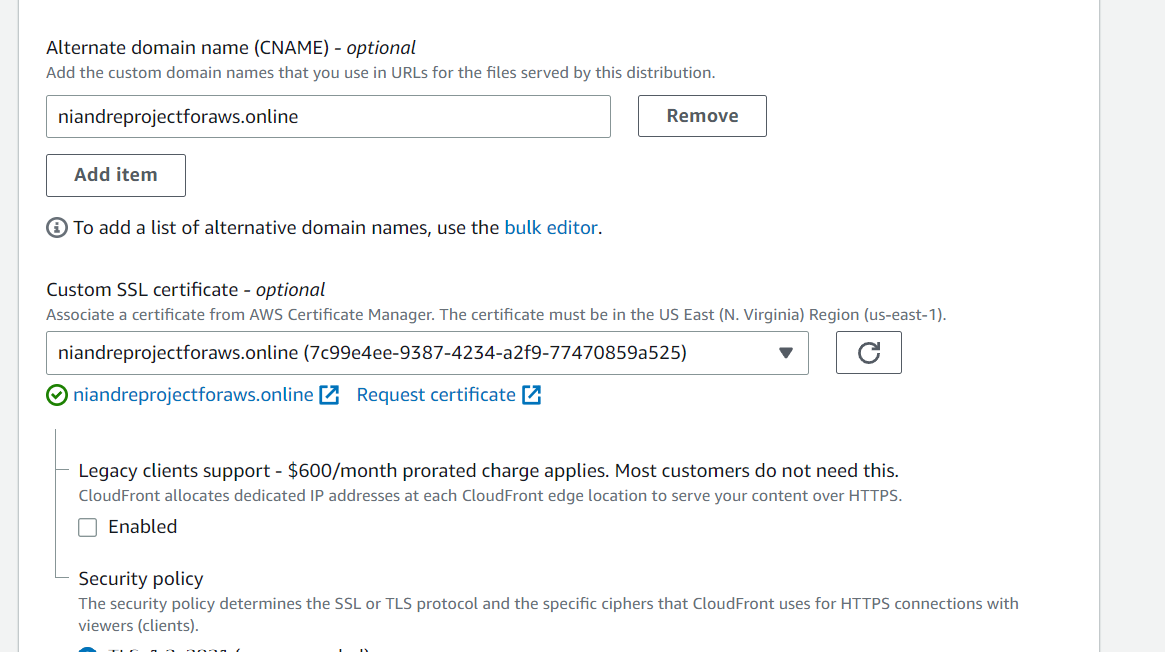
Description automatically generated with medium confidence

# Creating a CloudFront distribution for the site

Graphical user interface, text, application, email

Description automatically generated

# Selecting a previous CNAME record that I created for the website



# CloudFront distribution with the alternative name

Graphical user interface, text, application, email

Description automatically generated

# Accessing the website with the alternative name

Table

Description automatically generated

# Now, we need to integrate Amazon CloudWatch

# Creating an IAM role for logging into CloudWatch

Graphical user interface, text, application

Description automatically generated

# Setting up logs for the API gateway

Application

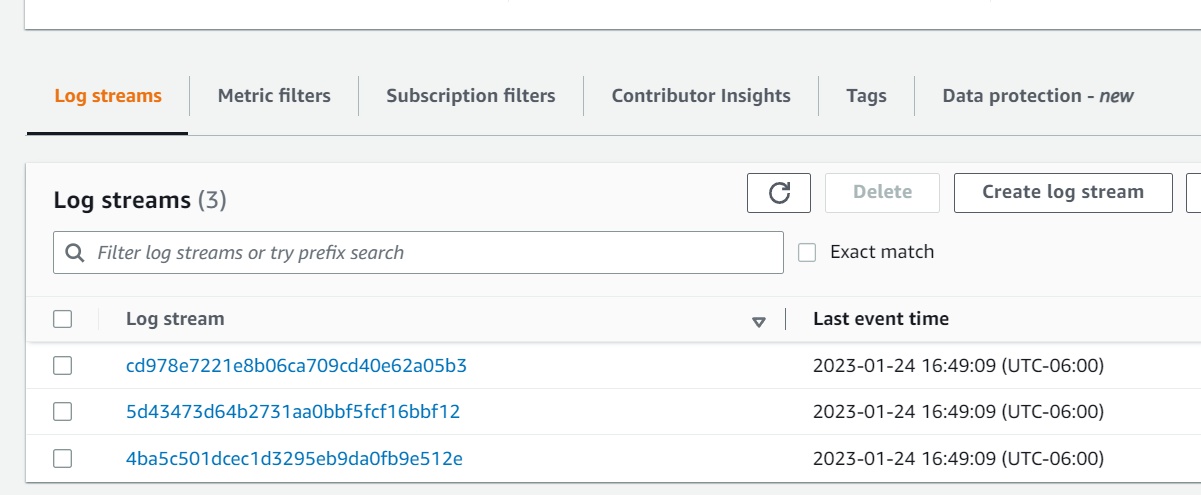
Description automatically generated

# Creating a log destination group in CloudWatch for the logs

Graphical user interface, text, application, email

Description automatically generated

# I created a new user, and I was able to generate these logs



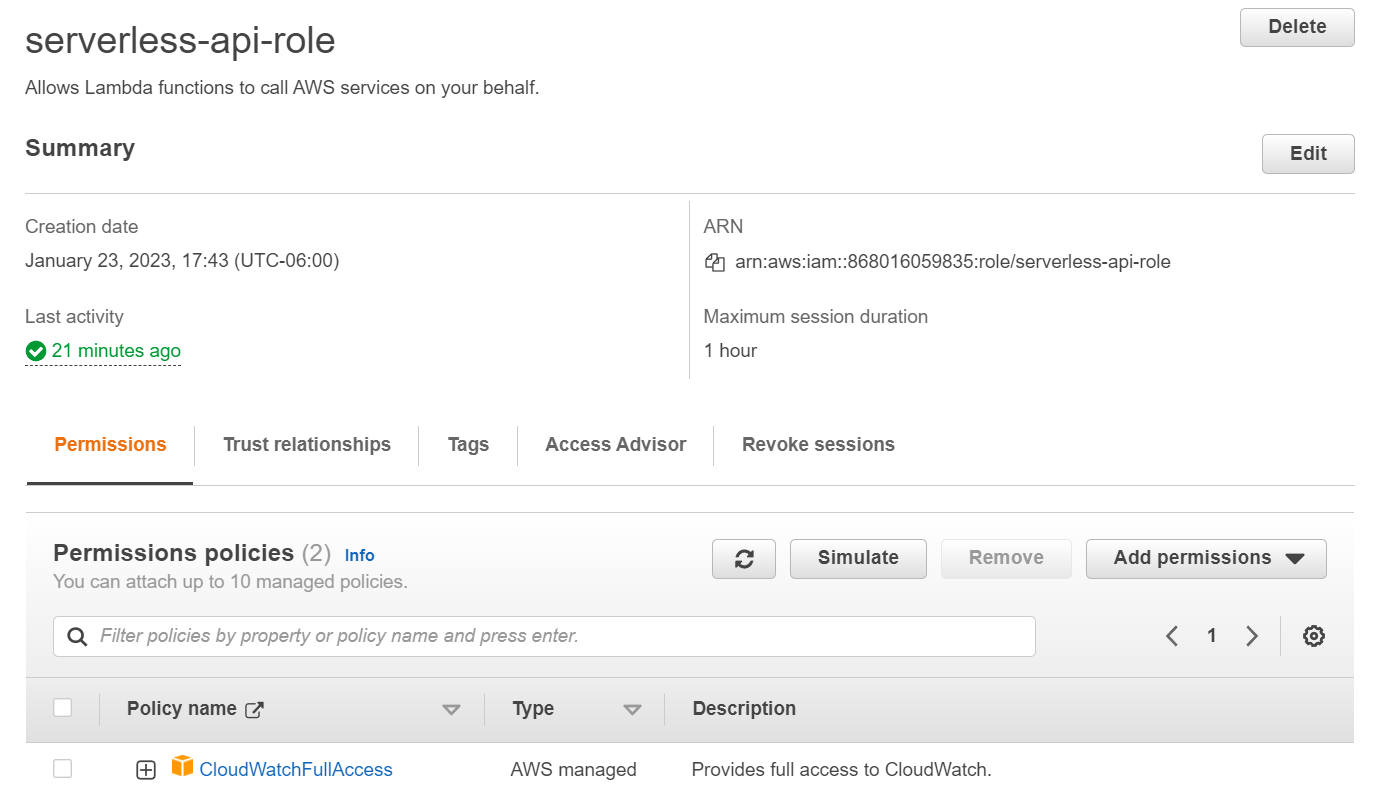
# Looking through some of the log streams

Graphical user interface, text, application, email

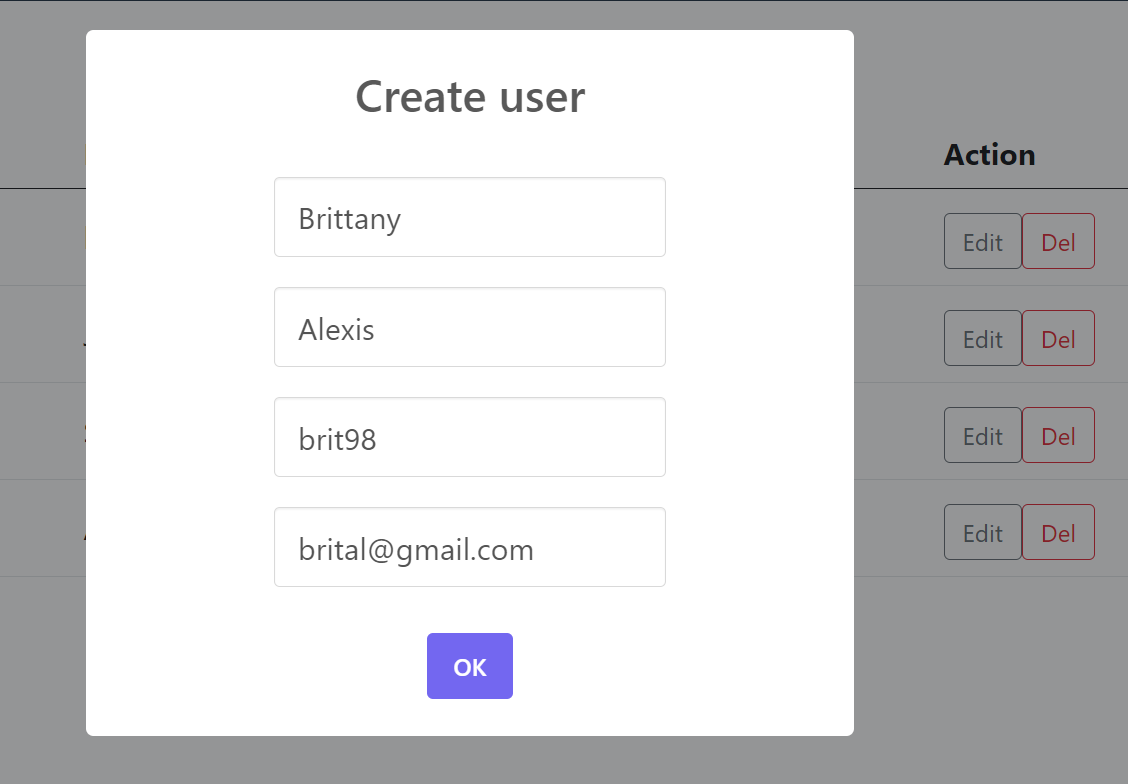
Description automatically generated

# Configure CloudWatch logging for the AWS Lambda

# Verifying to make sure that the role of the Lambda function has a policy



# Create a new user and see if the log was sent to the log group



Table

Description automatically generated

# Viewing the Log group for the Lambda function in CloudWatch

Graphical user interface, text, application, email

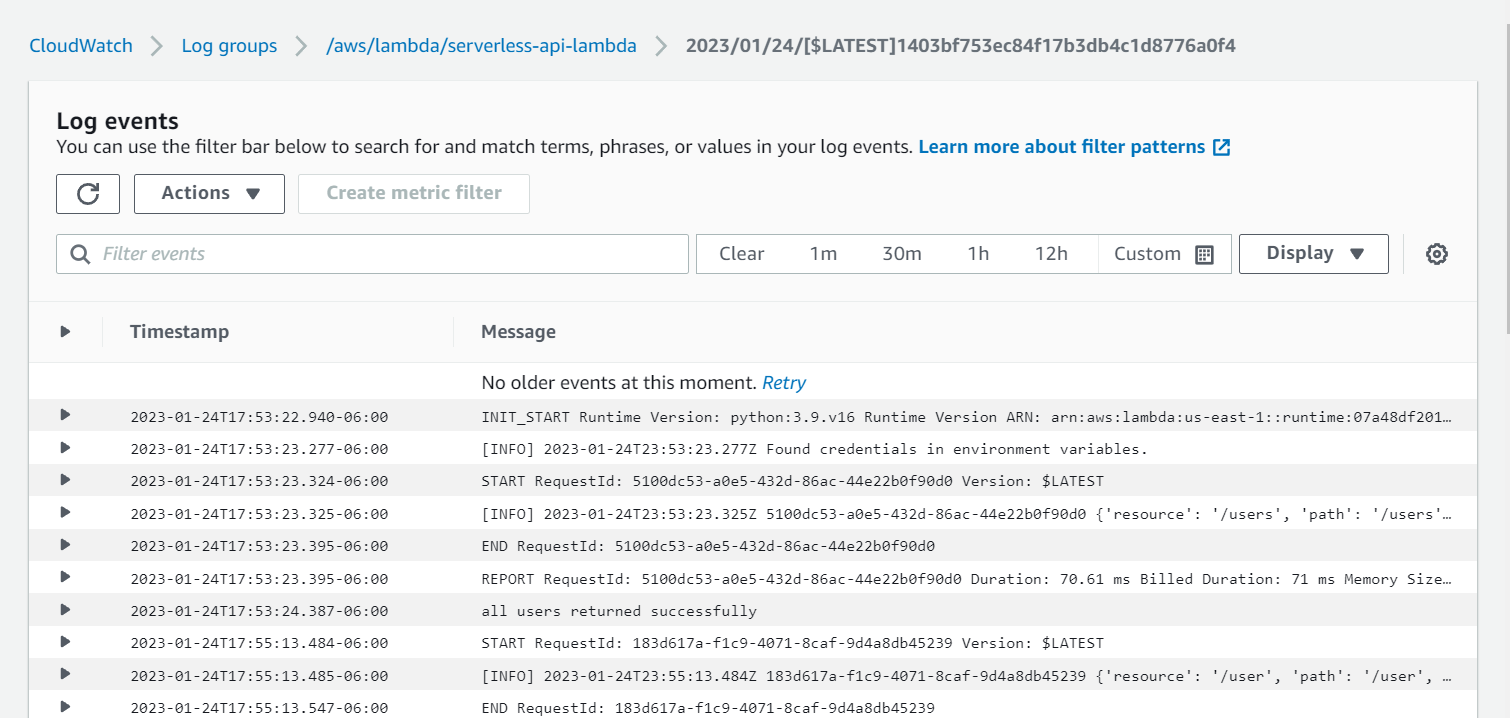
Description automatically generated

# Viewing some of the logs from the lambda function: After viewing this log, I realize that I had an error in the Lambda function. I used the keyword “sys” in the code to print out a message, but I forgot to import the sys module in the function.

Graphical user interface, text, application, email

Description automatically generated

-After fixing the above error, I rerun the function and the problem was fixed. Below is the output of a new log.



# Configure CloudWatch logs for DynamoDB

When I created the DynamoDB, CloudWatch come with the newer create database because CloudWatch monitors take the data from DynamoDB and process it into readable metrics. Now, I just need to retrieve CloudWatch data from a table that was created in DynamoDB.

# In the CloudWatch Metrics section, we can see our DynamoDB

Graphical user interface, text

Description automatically generated

# DynamoDB has different metrics. I need to select the Table Metric

Graphical user interface, text, application

Description automatically generated

# These are some of the metrics

Graphical user interface, text, application

Description automatically generated

# Graph output

Graphical user interface, text, application, email

Description automatically generated

# At this moment, the client is going to use the basic analytics for bucket size and several objects, which are free and are calculated automatically to save money. However, if the client wants to view more detailed analytics for S3, the client will need to turn on the request metrics and the cost of the service can add up.