

# Api Gateway Implementation Report!

**Prepared by:** Falilat Owolabi

**Date:** jan 19th 2026

**Cloud provider:** AWS

**Test Environment:** AWS console and Postman

**Testing Window:** jan 14th – jan 19th 2026

**Type:** secure integration Simulation (Self-led, Part of API Security Capstone)

## Format

This report shows implementation of an API gateway using AWS cloud service provider which integrates some security mechanisms to protect the API endpoint use in testing which include:

- Https integration
- Secured with jwt token

## 1. Executive Summary

This report documents the steps taken to implement AWS APIGateway with a simple API endpoint, it shows how some security configuration can be implemented to mitigate against security vulnerability and protect the API endpoint through the API gateway.

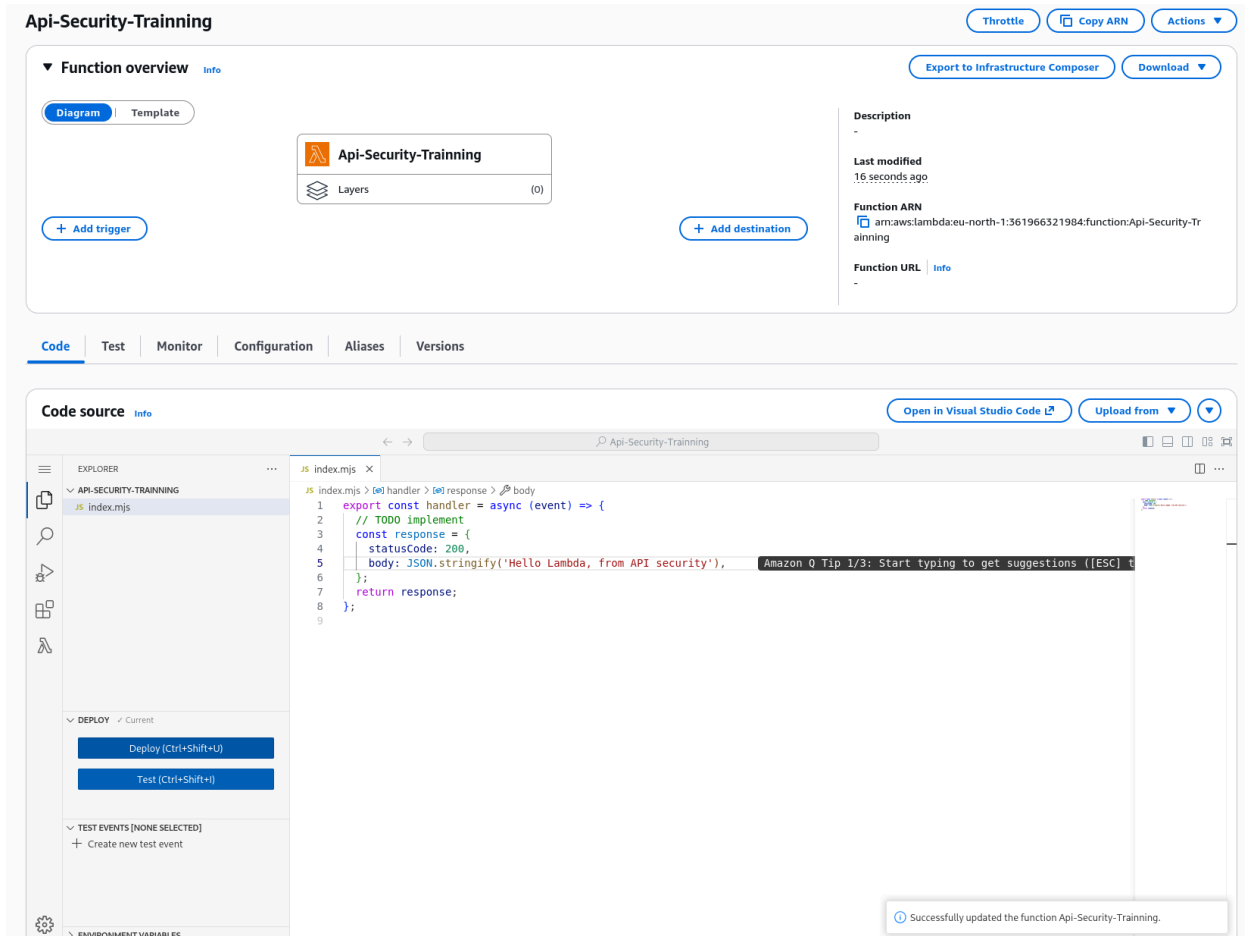
Additionally, through frontend JWT Authorization and TLS integrated, a protected API gateway was achieved.

## Methodology

The testing approach solely use the AWS console for setting up and implementation and POSTMAN was used for testing the result of the configuration

## Lambda Function

An API gateway with an API endpoint on AWS involves having a function/logic you can test your implementation with and AWS has a console that allows us to easily create a Lambda function that we can integrate into our API called Lambda function. After going through the necessary steps in AWS console to create a lambda function, I was able to create a simple function that can be used as shown below:



Lambda console after creating lambda function.

Testing the lambda function to be sure

Executing function: succeeded (logs [L](#))

Details

```
{
  "statusCode": 200,
  "body": "\u000aHello Lambda, from API security\u000a"
}
```

Summary

<b>Code SHA-256</b> b3QWX8x/00A4HrdVuueY7EqOUAvTgddGR5SbMx2w=	<b>Execution time</b> 1 second ago
<b>Function version</b> \$LATEST	<b>Request ID</b> c131bfc1-7f7d-45b3-af2a-f665ebb3fbaf
<b>Duration</b> 8.46 ms	<b>Billed duration</b> 151 ms
<b>Resources configured</b> 128 MB	<b>Max memory used</b> 79 MB
<b>Init duration</b> 142.30 ms	
<b>Log output</b>	

The area below shows the last 4 KB of the execution log. [Click here](#) to view the corresponding CloudWatch log group.

```
START RequestId: c131bfc1-7f7d-45b3-af2a-f665ebb3fbaf Version: $LATEST
END RequestId: c131bfc1-7f7d-45b3-af2a-f665ebb3fbaf
REPORT RequestId: c131bfc1-7f7d-45b3-af2a-f665ebb3fbaf  Duration: 8.46 ms    Billed Duration: 151 ms Memory Size: 128 MB    Max Memory Used: 79 MB    Init Duration: 142.30 ms
```

Test event Info

CloudWatch Logs Live Tail Save Test

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

Create new event

Edit saved event

Invocation type

☒ Synchronous

Executes the Lambda function and blocks until receiving the function's response, with a maximum timeout of 15 minutes. Returns function output or error details directly to the calling application.

☐ Asynchronous

Enqueues the Lambda function for execution and returns immediately with a request ID. Function processes independently, with results optionally sent to a configured destination like SQS, SNS, or EventBridge.

Event name

firs-test

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

# Build with Rest API

The next step is to build a rest Api, that we will use with our lambda function, the flow below shows

## Create REST API [Info](#)

### API details

#### ☒ New API

Create a new REST API.

#### ☐ Clone existing API

Create a copy of an API in this AWS account.

#### ☐ Import API

Import an API from an OpenAPI definition.

#### ☐ Example API

Learn about API Gateway with an example API.

### API name

Rest-API

### Description - optional

testing for API Security

### API endpoint type

Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

Regional

### Security policy - [new](#) [Info](#)

Transport Layer Security (TLS) protects data in transit between a client and server. The security policy also determines the cipher suite options that clients can use with your API.

SecurityPolicy\_TLS13\_1\_2\_2021\_06

### Endpoint access mode - [new](#) [Info](#)

Provide additional governance for your APIs.

#### ☒ Basic

Allow all clients to access the API

#### ☐ Strict (recommended)

Enforce Server Name Indication (SNI) validation

### IP address type [Info](#)

Select the type of IP addresses that can invoke the default endpoint for your API.

#### ☒ IPv4

Supports only edge-optimized and Regional API endpoint types.

#### ☐ Dualstack

Supports all API endpoint types.

[Cancel](#)

[Create API](#)

Creating restAPI flow with the integration of TLS, to make sure communication between the client and gateway is secured.

After that, I created a resource and also create a method that will be used for the api, where I am able to integrate the lamda function created above, tested that it is working fine and afterwards, I have the Output below ready to be deployed

API Gateway > APIs > Resources - Rest-API (sx2kxmpqf0)

Successfully edited method request for "/ANY". Redeploy your API for the update to take effect.

### Resources

[Create resource](#)

- /
- /api-gateway-demo
- ANY

### /Api-gateway-demo - ANY - Method execution

ARN: [arn:aws:execute-api:eu-north-1:361966321984:xx2kxmpqf0/\\*/\\*/\\*-api-gateway-demo](#) Resource ID: [rpy0d0m](#)

[API actions](#) [Deploy API](#) [Update documentation](#) [Delete](#)

Client → Method request → Integration request → Lambda integration

← Method response ← Integration response (Proxy integration)

Method request | Integration request | Integration response | Method response | **Test**

### Test method

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

**Method type**  
GET

**Query strings**  
param1=value1&param2=value2

**Headers**  
Enter a header name and value separated by a colon (:). Use a new line for each header.  
header1:value1  
header2:value2

**Client certificate**  
No client certificates have been generated.

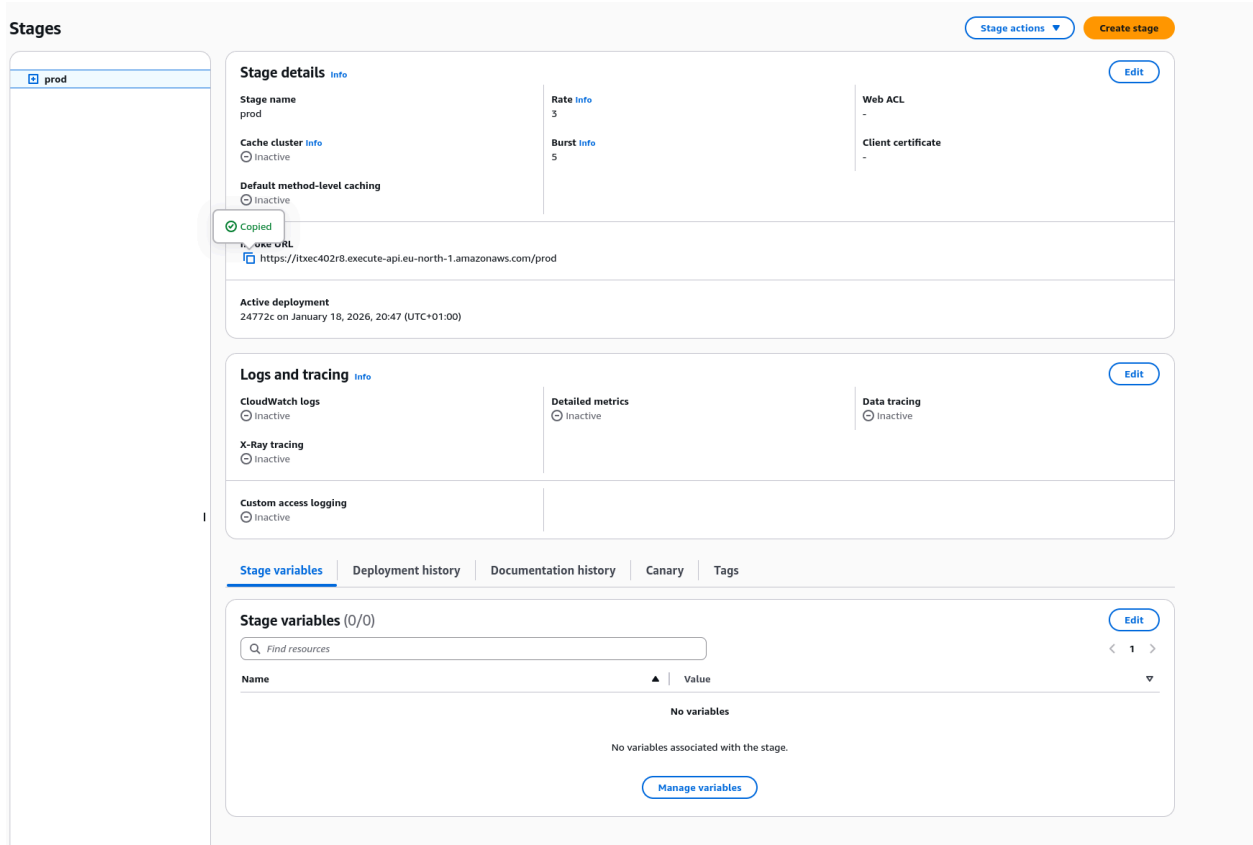
[Test](#)

**/Api-gateway-demo - ANY method test results**

Request	Latency ms	Status
/api-gateway-demo	377	200

**Response body**  
"Hello Lambda, from API security"

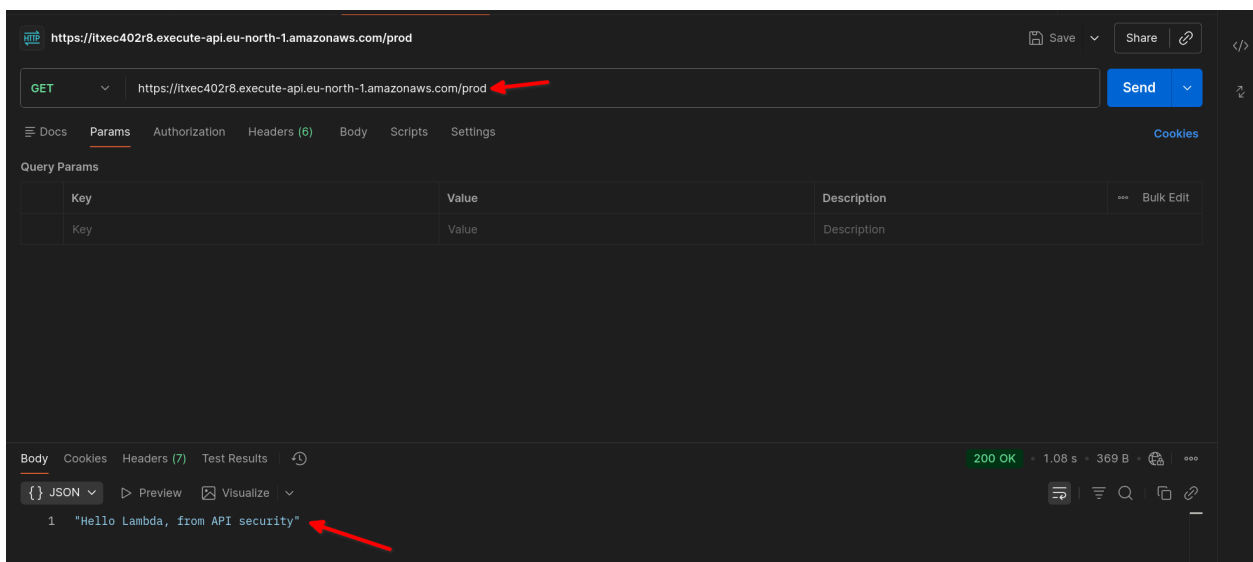
**Response headers**  
{  
 "X-Amzn-Trace-Id": "Root=1-696d337a-29e8836fba4679aa3ee83651;Parent=5cf28ab139115513;Sampled=0;Lineage=1:bc4ed09:0"  
}



The deployment is done to the prod stage as seen above.

## Testing The the invoke url in post man

Now that the API gateway has been deployed, I then copied the invoke url and past in postman to confirm it can be access properly



## Using JWT as the AuthorizationToken to Access the endpoint

Now that the API gateway is working well, the next phase is to add an Authorizer that can help secure it from being accessed by unauthorised user which will lead us to having to create an authorizer lambda function that will be added to our Rest API

**Create function** [Info](#)  
Choose one of the following options to create your function.

☒ **Author from scratch**  
Start with a simple Hello World example.


☐ **Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.


☐ **Container image**  
Select a container image to deploy for your function.

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.  
  
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

**Runtime** [Info](#)  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.  

Node.js 24.x  Last fetched 19/01/2026, 07:26:53

**Durable execution - new** [Info](#)  
Enable durable execution to simplify building resilient multi-step applications that checkpoint progress and resume after interruptions. Supports Python and Node.js runtimes. [View pricing](#)   
☐ Enable

**Architecture** [Info](#)  
Choose the instruction set architecture you want for your function code.  

☐ arm64  
☒ x86\_64

**Permissions** [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

[► Change default execution role](#)

[► Additional configurations](#)  
Use additional configurations to set up networking, security, and governance for your function. These settings help secure and customize your Lambda function deployment.

[Cancel](#) [Create function](#)

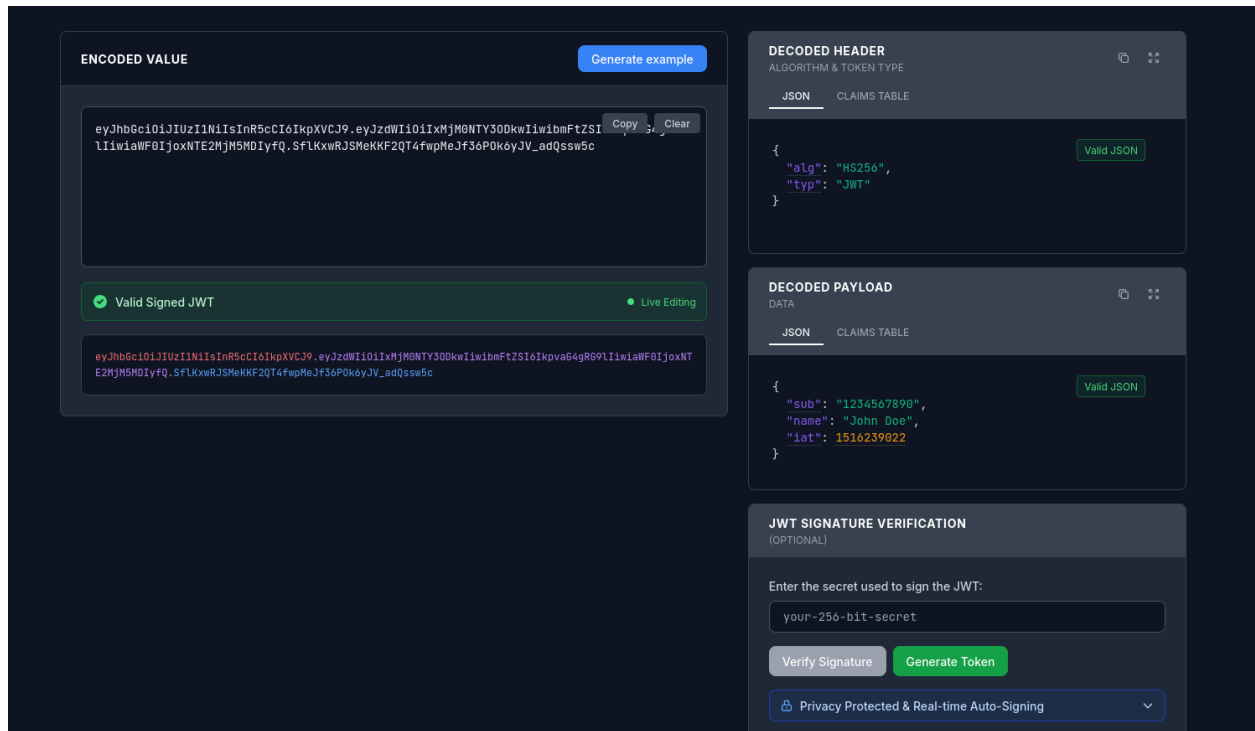
Creating the authorization lambda function

```

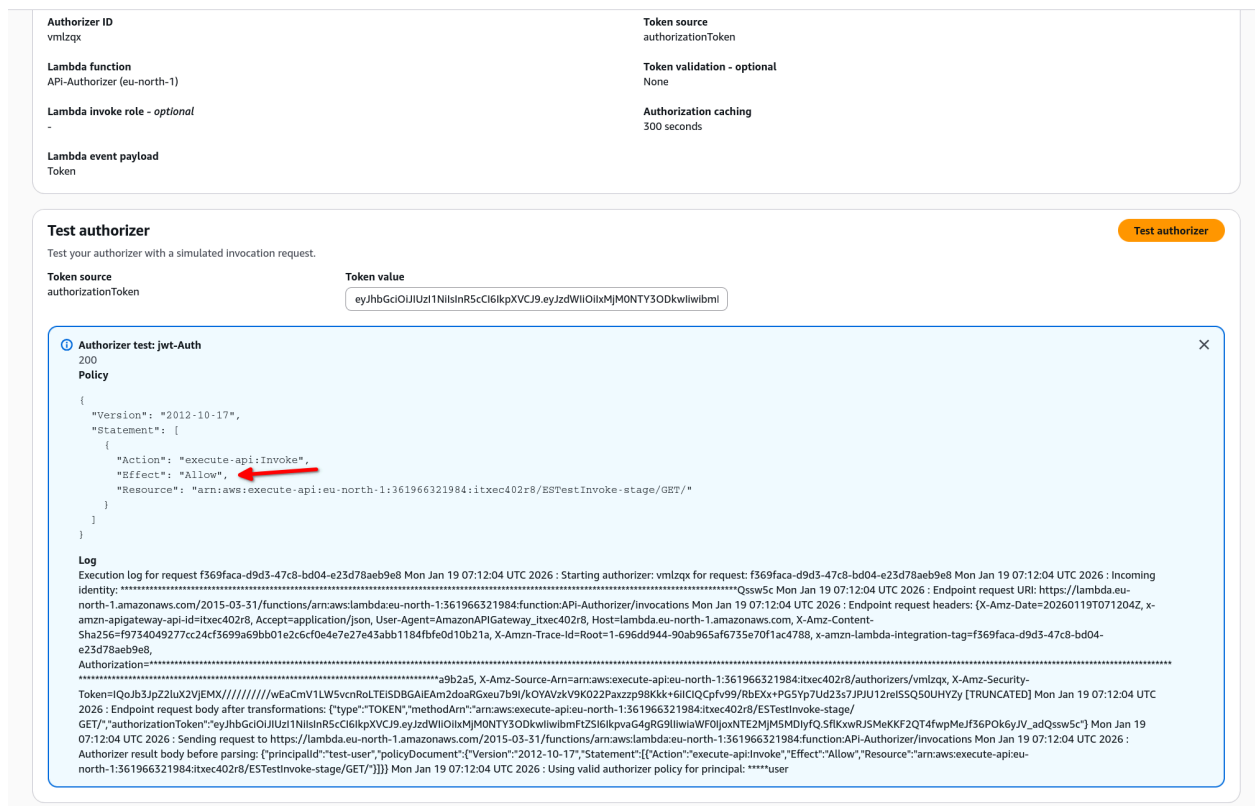
js index.mjs > handler > handler > token
1  exports.handler = async (event) => {
2      const token = event.authorizationToken;
3      // In a real scenario, validate the token (e.g., JWT verification, database lookup)
4      if (token === "secretToken") {
5          return generatePolicy('user', 'Allow', event.methodArn);
6      } else {
7          return generatePolicy('user', 'Deny', event.methodArn);
8      }
9  };
10
11  const generatePolicy = (principalId, effect, resource) => {
12      return {
13          principalId,
14          policyDocument: {
15              Version: '2012-10-17',
16              Statement: [{
17                  Action: 'execute-api:Invoke',
18                  Effect: effect,
19                  Resource: resource
20              }]
21          },
22          // Optional: Pass data to the backend Lambda function via 'context'
23          // context: {
24          //     "userId": "123"
25          // }
26      };
27  };
28

```

I got this code online when I searched, edit it to suit my purpose and here I realised I need a token and I use [xjwt.io](https://xjwt.io) to generate a new token



I copied the token from here and past it in my lambda function, deployed it and tested it to confirm it is working as expected



Back to the API gateway console, in my rest API created earlier, I clicked on



Authorizer to add the lambda function created for authorizer, to the ApiGateway as seen below

**Create authorizer** [Info](#)

**Authorizer details**

**Authorizer name**  
JWT-Auth

**Authorizer type** [Info](#)  
Choose to authorize your API calls using one of your Lambda functions or a Cognito User Pool.  
☒ Lambda  
☐ Cognito

**Lambda function**  
Provide the Lambda function name or alias. You can also provide an ARN from another account.  
eu-north-1

**Grant API Gateway permission to invoke your Lambda function**  
When you save your changes, API Gateway updates your Lambda function's resource-based policy to allow this API to invoke it.

**Lambda invoke role - optional**  
Specify an optional role API Gateway will use to make requests to your authorizer. For optimal API performance it is strongly recommended to activate Regional STS in the region where your API is located.

**Lambda event payload**  
Choose token to send a single header that contains an authorization token. Choose request to send all request parameters.  
☒ Token  
☐ Request

**Token source**  
Enter the header that contains the authorization token.

**Token validation - optional**  
Enter a regular expression to validate tokens.

☒ Authorization caching

**TTL (1-3600 seconds)**

[Cancel](#) [Create authorizer](#)

After the creation of Authorization in the API gateway, I then go back to the AP-gateway-demo and I changed the Authorization from Non to the JWT-Auth I created and redeployed the API gateway as shown below

Resources

Create resource

/

ANY

/Api-gateway-demo

API actions

Deploy API

Update documentation

Delete

/ - ANY - Method execution

ARN: `arn:aws:execute-api:eu-north-1:361966321984:itxec402r8/*/*/*`

Resource ID: `e67r1qas0l`

Client → Method request → Integration request → Lambda integration

← Method response ← Integration response (Proxy integration) ←

Method request | Integration request | Integration response | Method response | Test

Method request settings

Authorization: jwt-Auth (highlighted with a red arrow)

Request validator: Validate body, query string parameters, and headers

API key required: False

SDK operation name: Generated based on method and path

Request paths (0)

Name | Caching

No request paths

No request paths defined

URL query string parameters (0)

Name | Required | Caching

No URL query string parameters

After this I then copied the invoke url and paste in postman for testing to see if it will be denied without the right Authorization token and allowed with the right Authorization token as seen below respectively.

https://itxec402r8.execute-api.eu-north-1.amazonaws.com/prod

PATCH | https://itxec402r8.execute-api.eu-north-1.amazonaws.com/prod

Send

Docs | Params | Authorization | Headers (7) | Body | Scripts | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

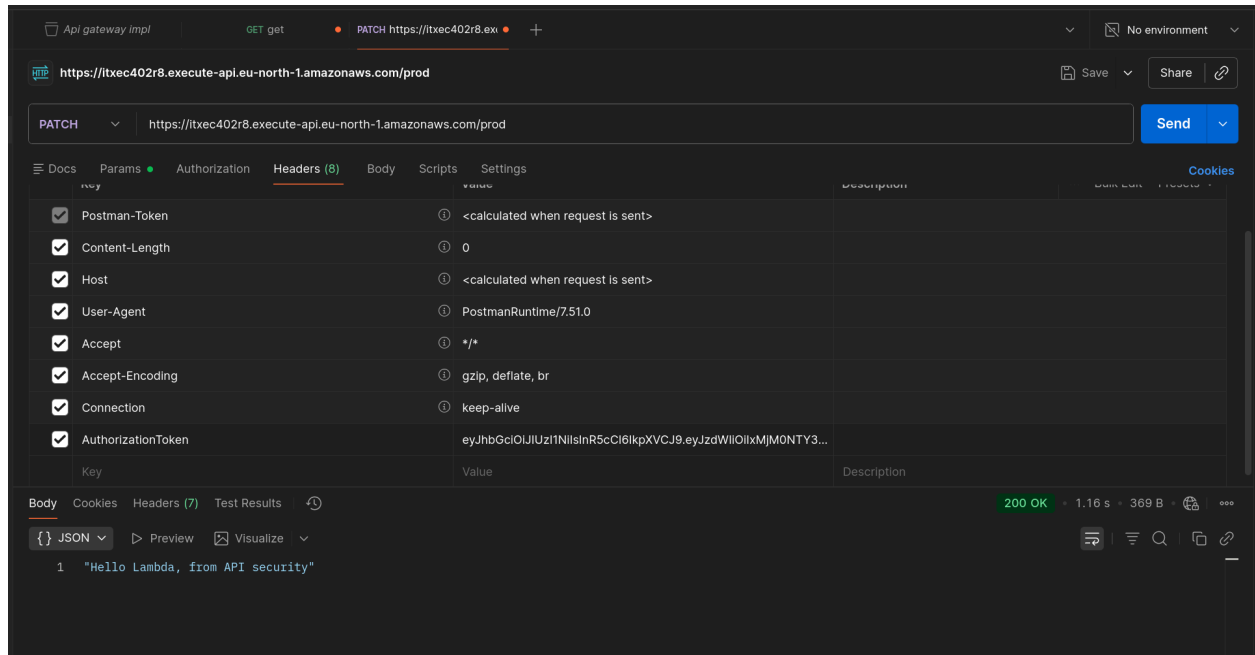
1 Ctrl+Alt+P to Ask AI

Body | Cookies | Headers (7) | Test Results

401 Unauthorized | 683 ms | 299 B

JSON | Preview | Pass the correct auth credentials

```
1 {
2   "message": "Unauthorized"
3 }
```



## Conclusion

The Implementation of API gateway introduced me to the cloud space and how services are rolling out features to save APIs and infrastructure . The protection each service has put in place and from a security perspective, what could go wrong if the protection is not in place in the API, API gateway in my opinion added another layer of protection to data in the cloud, protecting malicious injection in the system.

## Challenges faced:

I struggled, understanding how to go about it. After so much research I was able to come up with this implementation even with the fact that I could not implement all features expected to be implemented but in the end I have a better understanding of the cloud service and what my role will be as an API tester.