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.

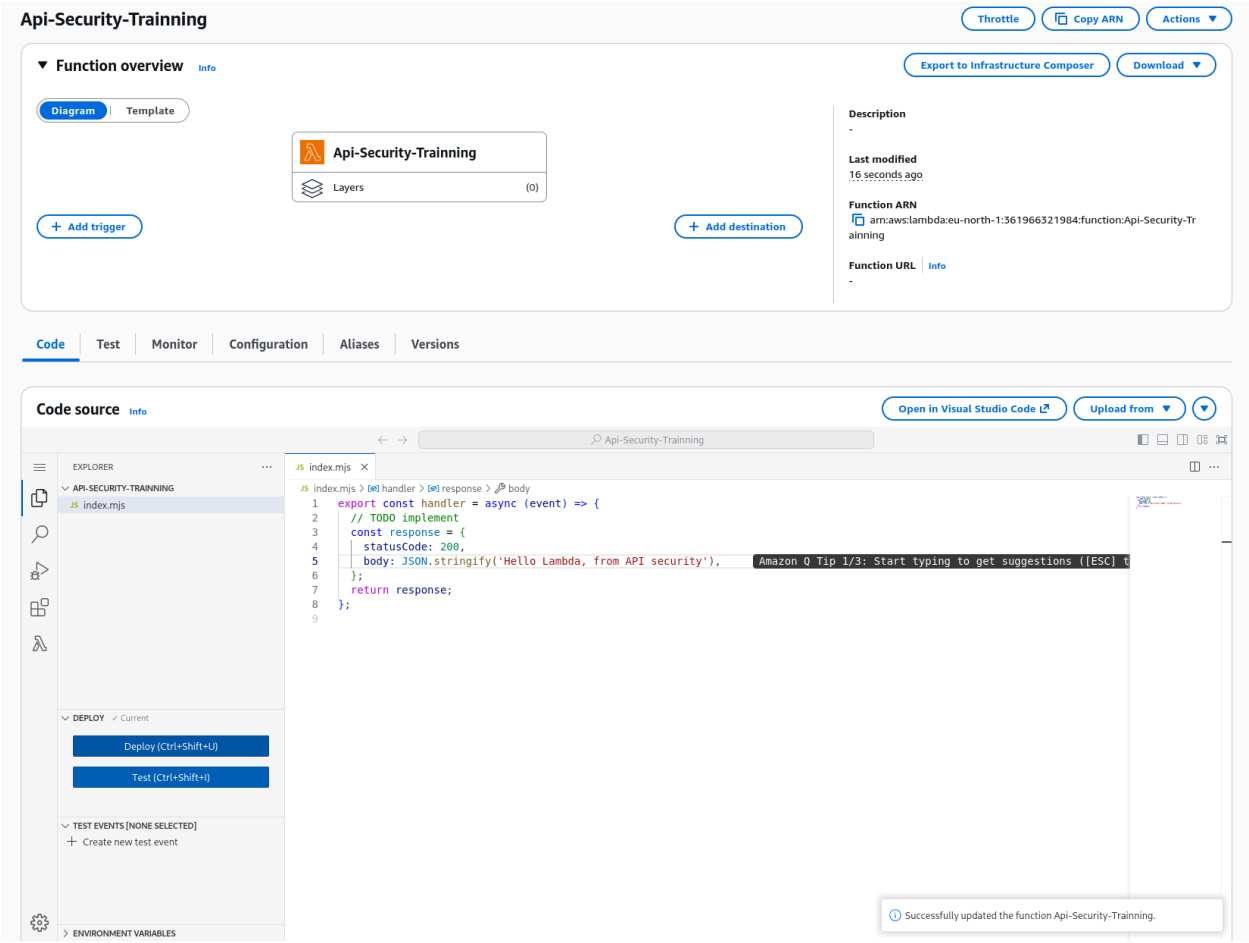
Additionally, through frontend JWT Authorization and TLS integrated, a protected API gateway was achieved.

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

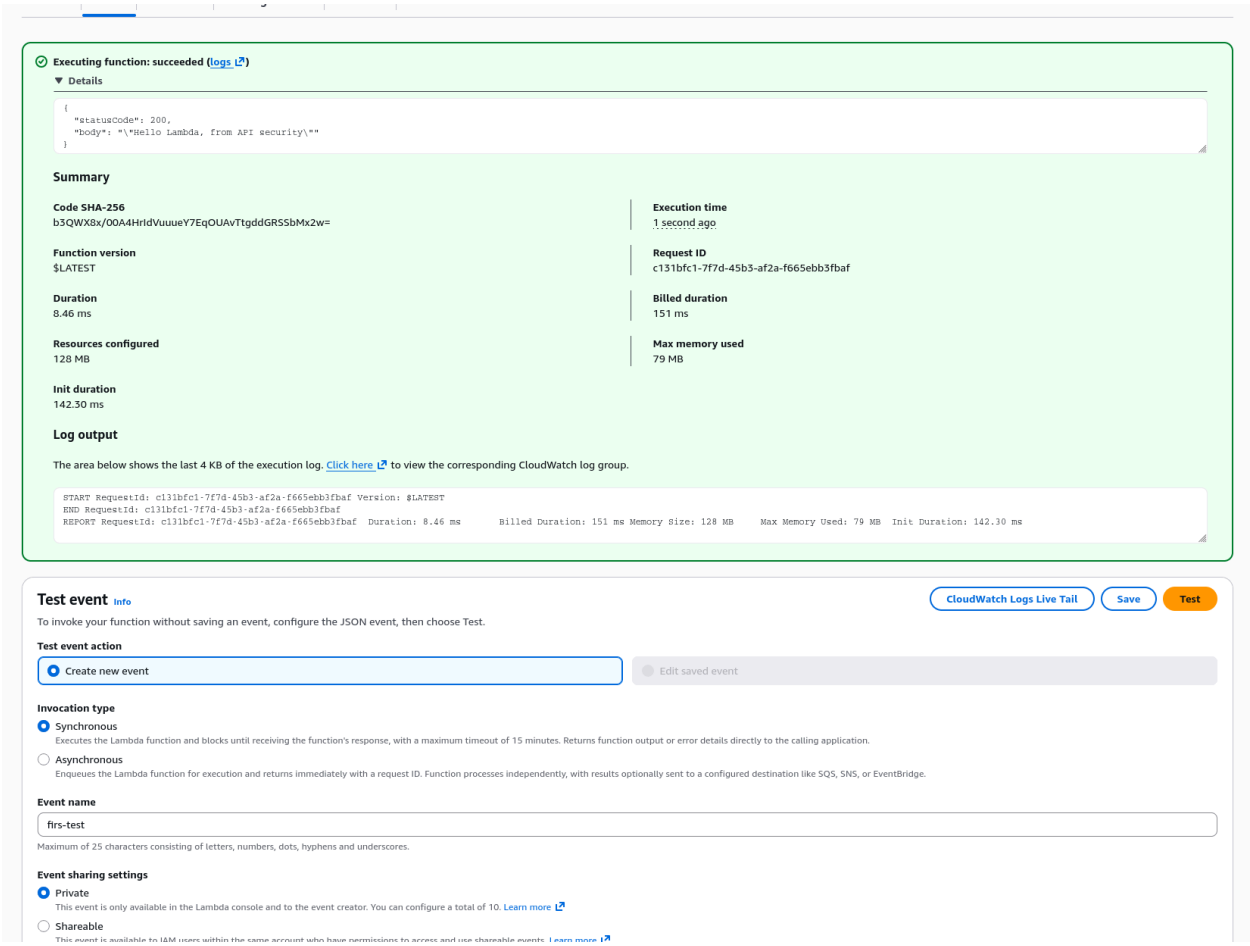
Lambda Function

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:



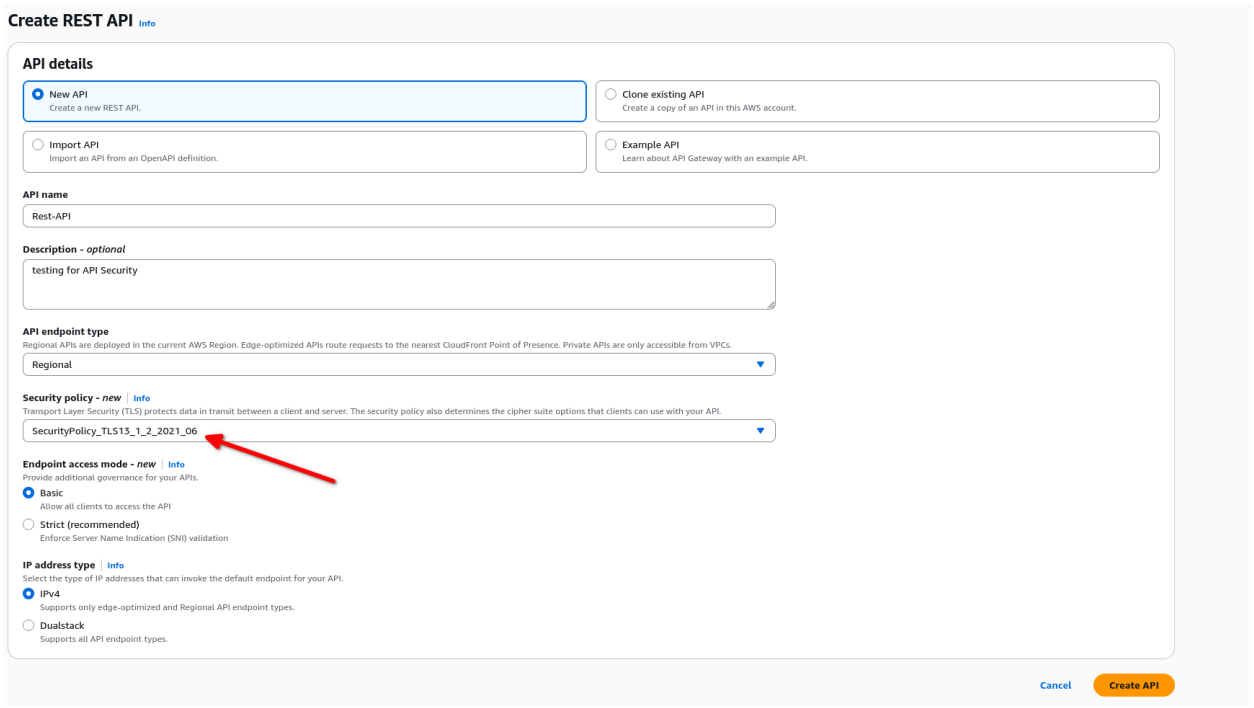
Lambda console after creating lamba function.

Testing the lambda function to be sure



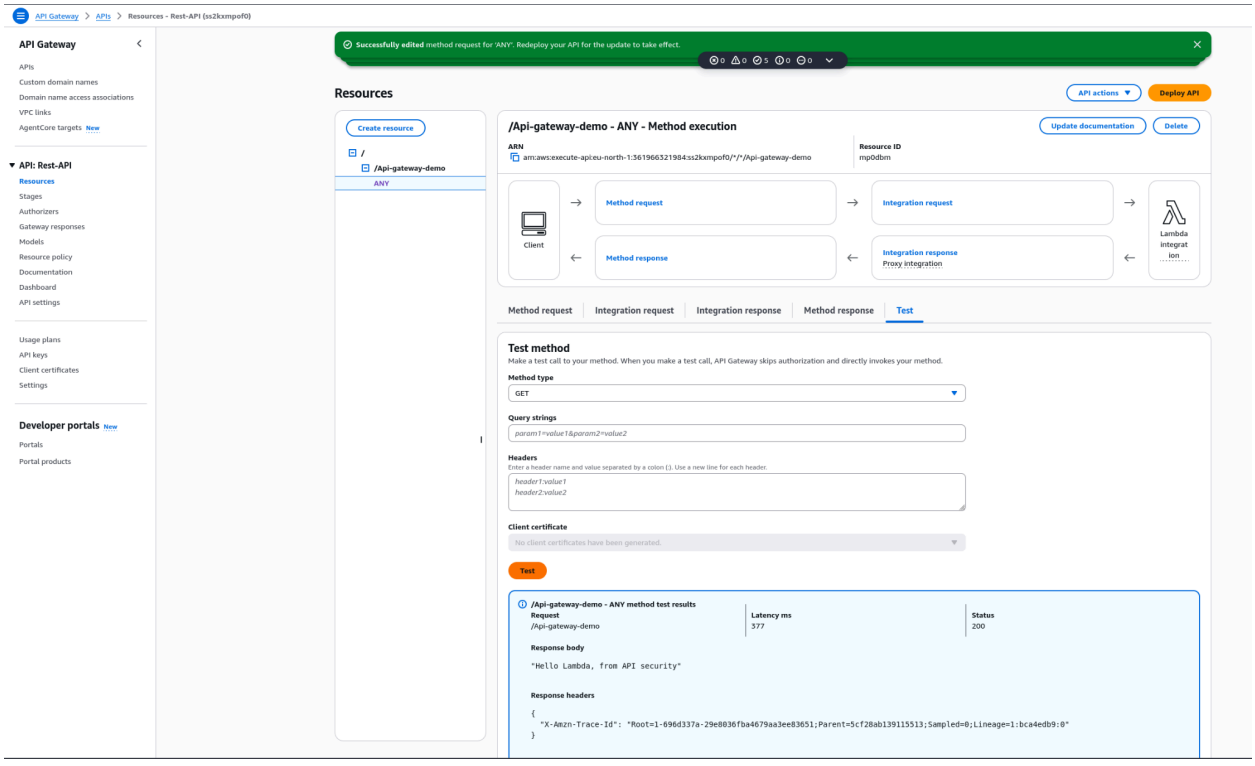
Build with Rest API

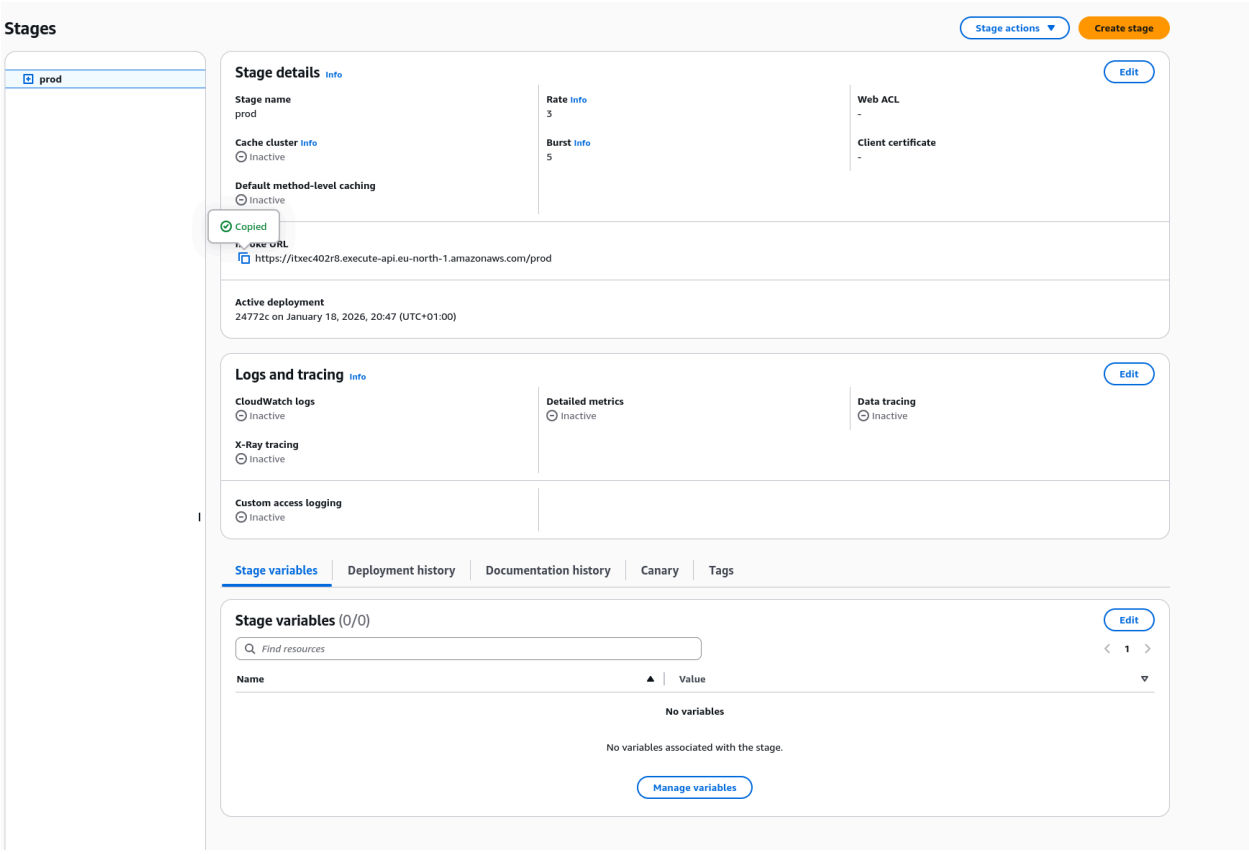
The next step is to build a rest Api, that we will use with our lambda function, the flow below shows



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

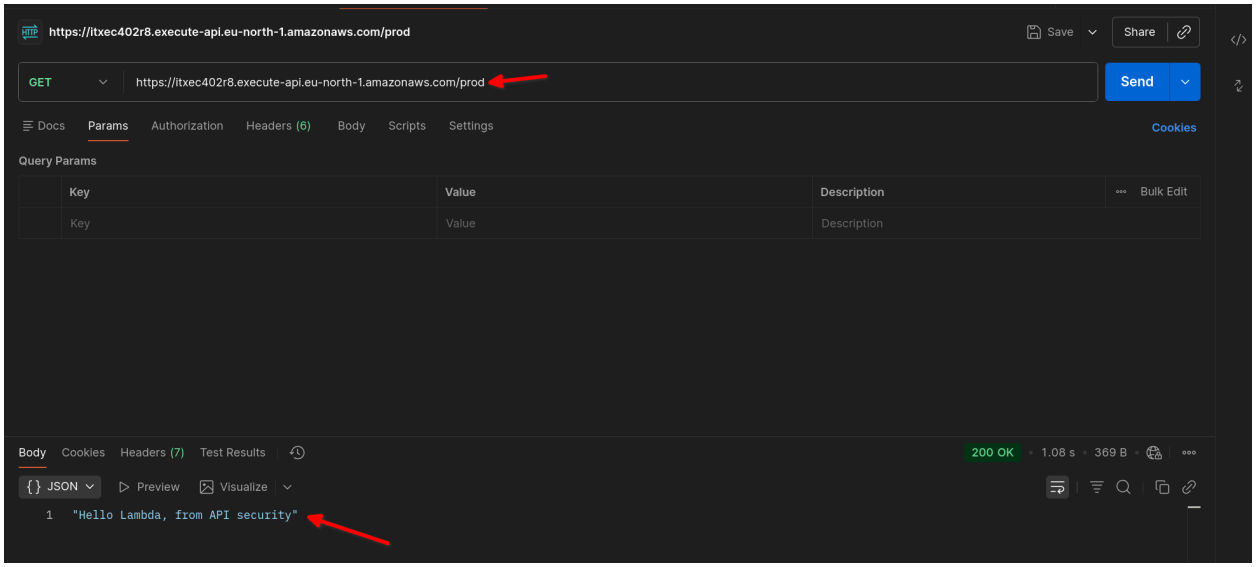




The deployment is done to the prod stage as seen above.

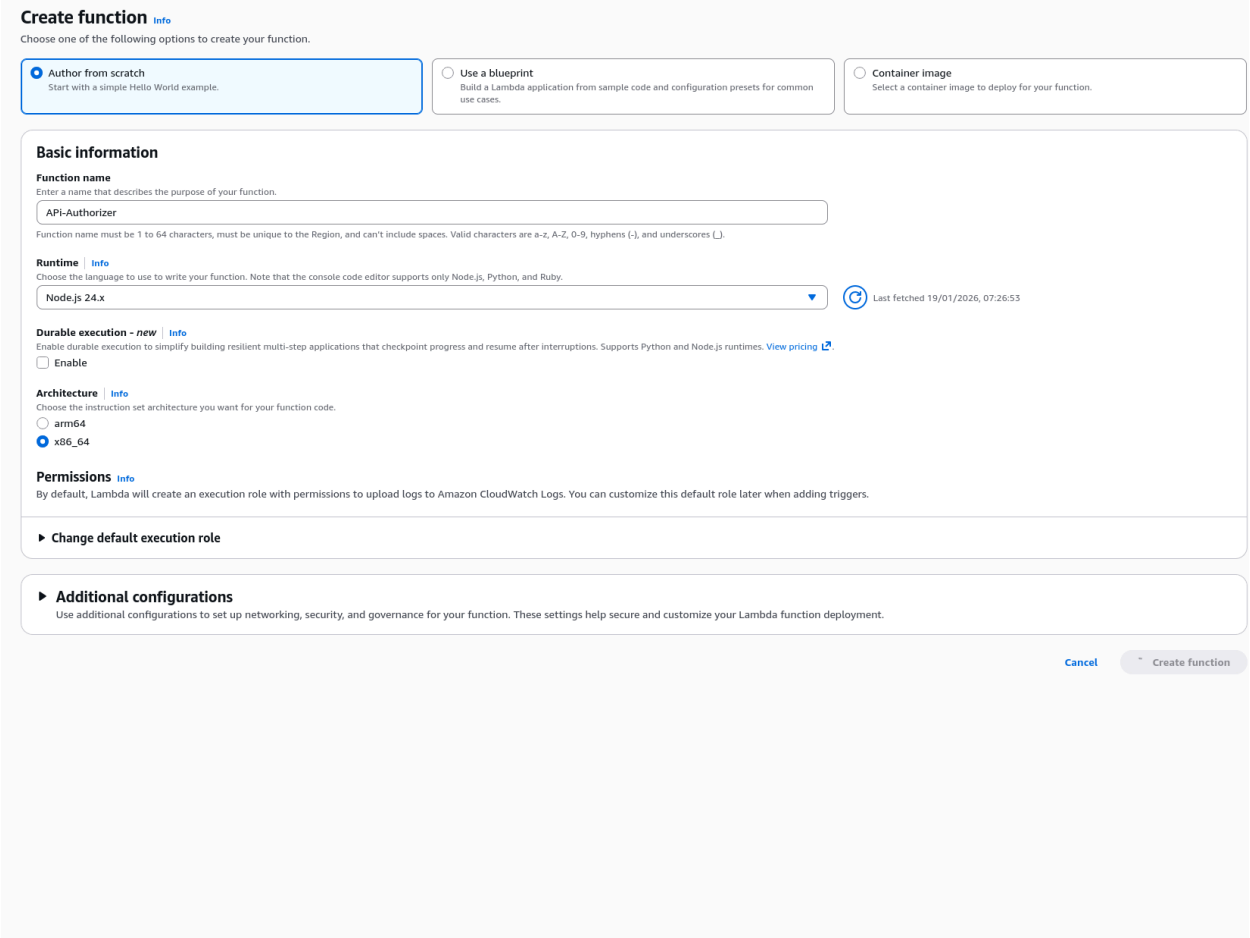
Testing The the invoke url in post man

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



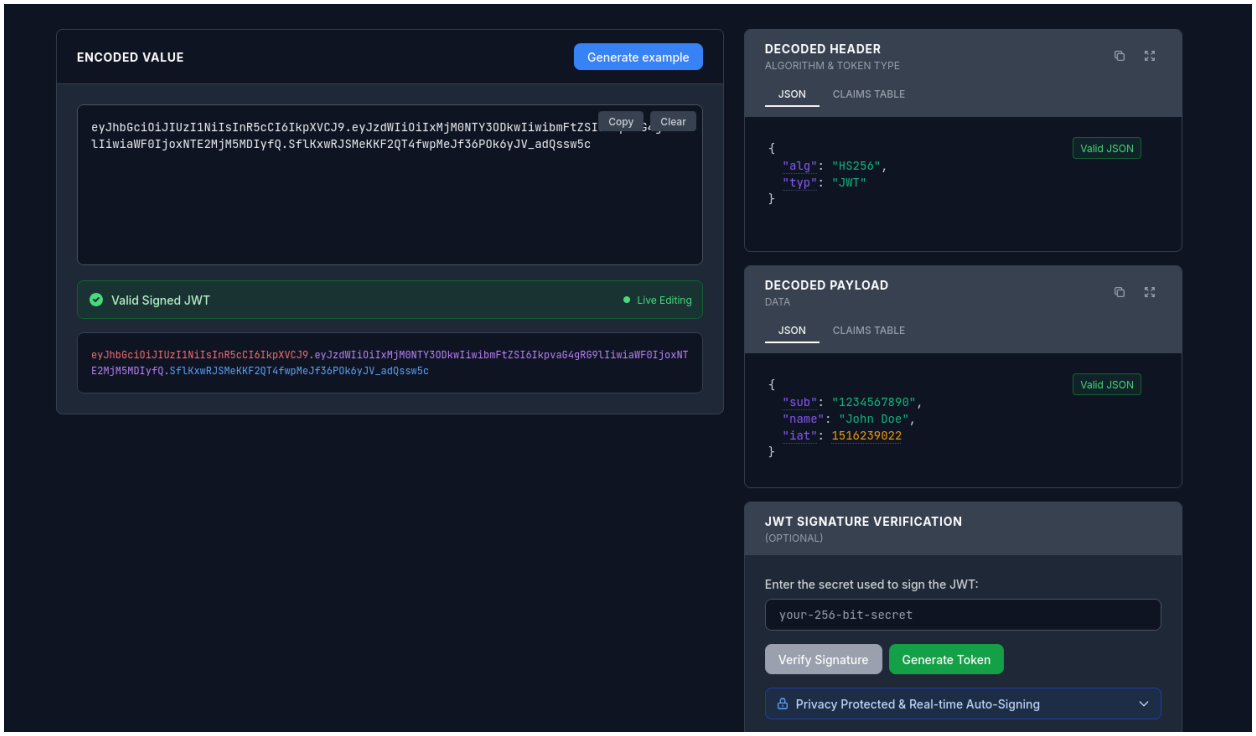
Creating the authorization lambda function

index.mjs > handler > handler > token

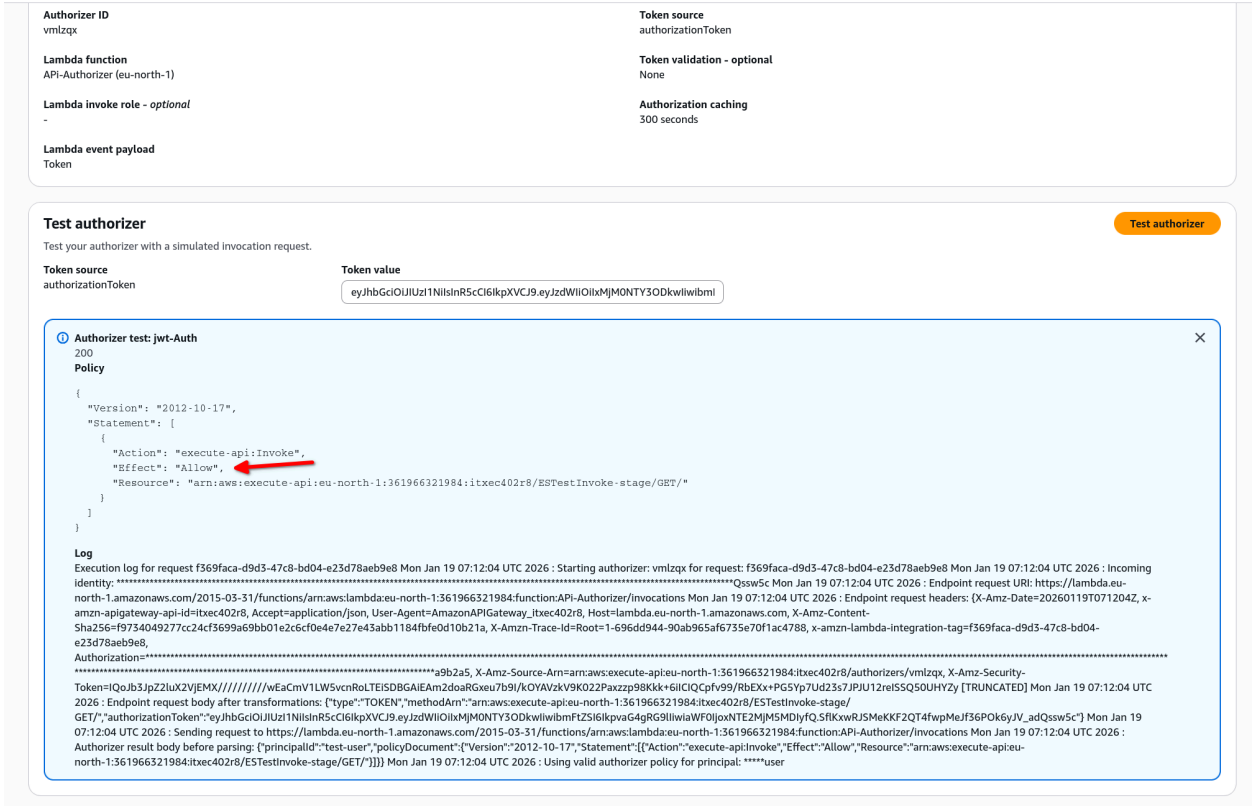
```
1   exports.handler = async (event) => {
2     const token = event.authorizationToken;
3     // In a real scenario, validate the token (e.g., JWT verification, database lookup)
4     if (token === "secretToken") {
5       return generatePolicy('user', 'Allow', event.methodArn);
6     } else {
7       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
```

Amazon Q Tip 1/3: Start typing to get

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 to generate a new token

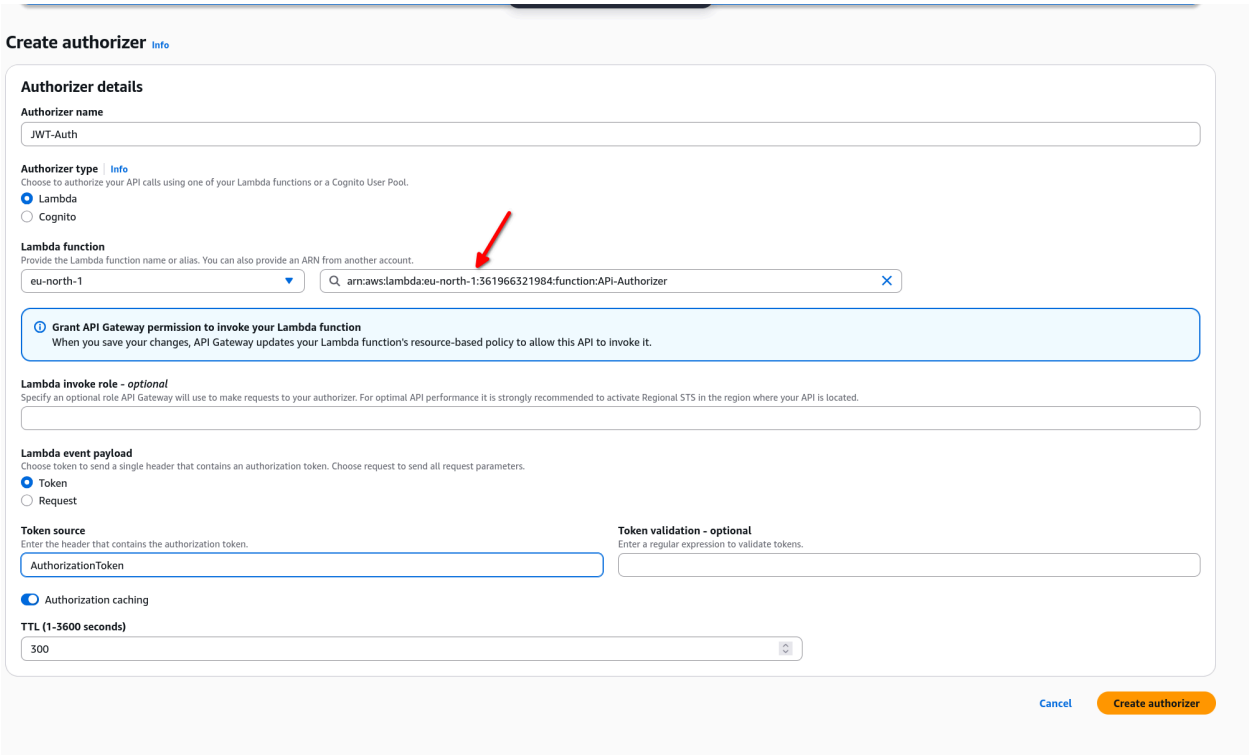


I copied the token from here and past it in my lambda function, deployed it and tested it to confirm it is working as expected

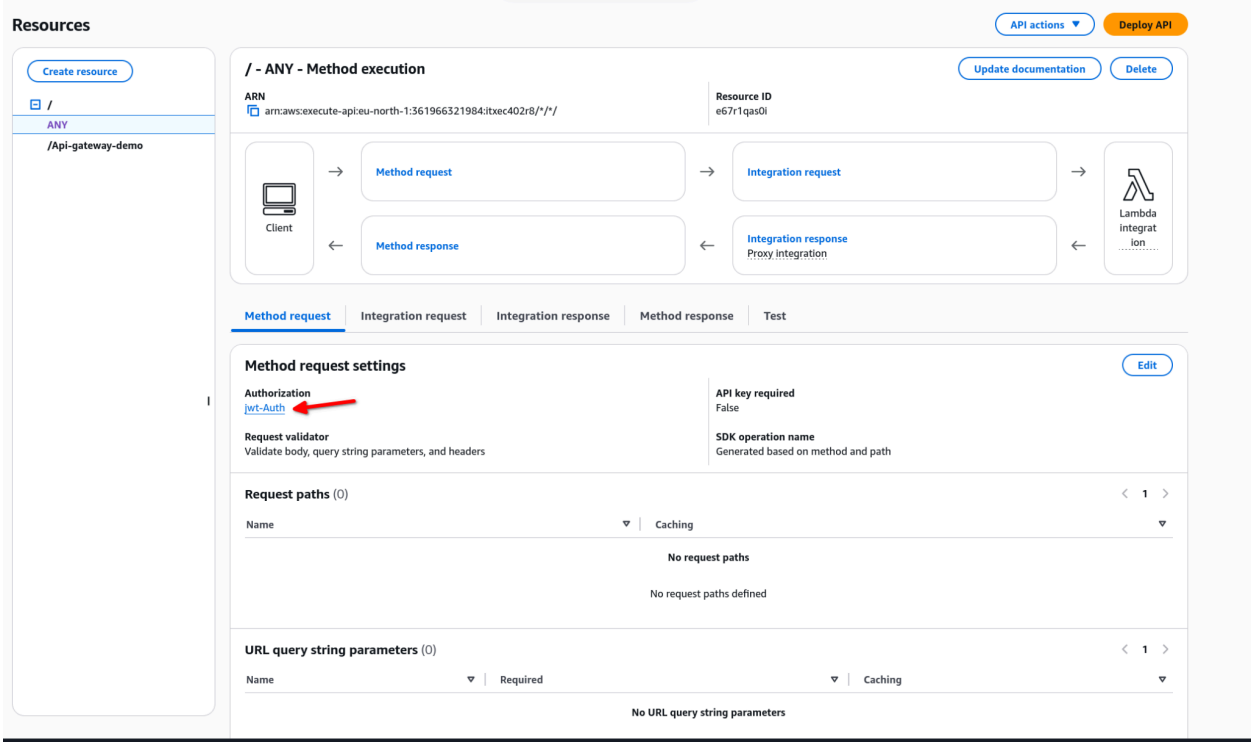


Back to the API gateway console, in my rest APi created earlier, I clicked on

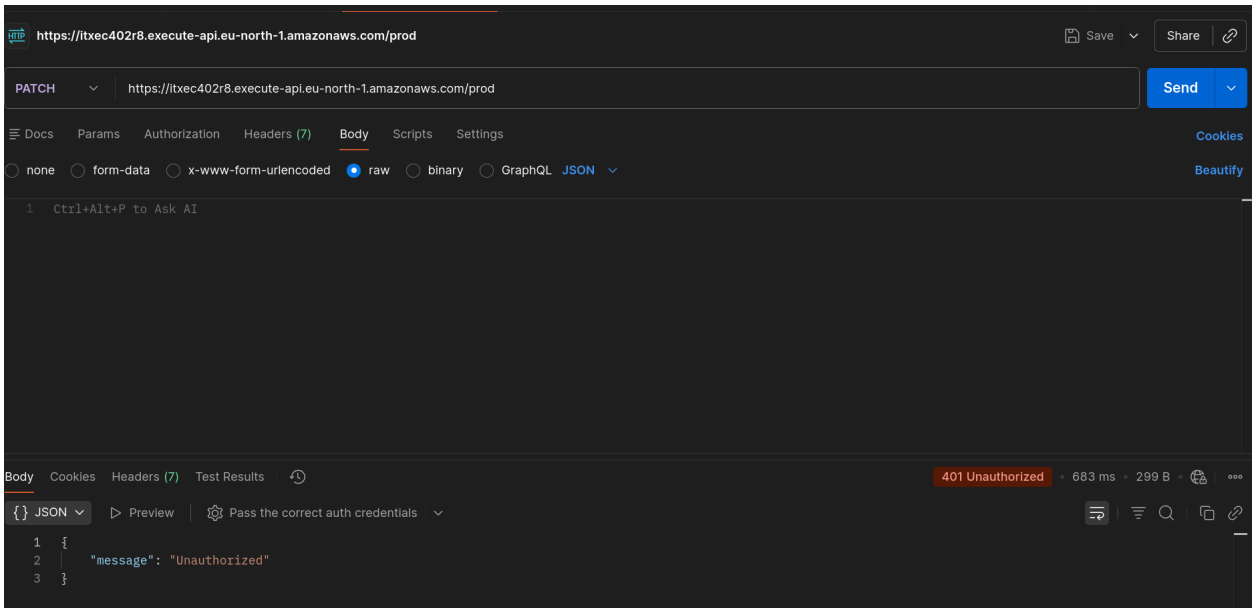
Authorizer to add the lambda function created for authorizer, to the ApiGateway as seen below

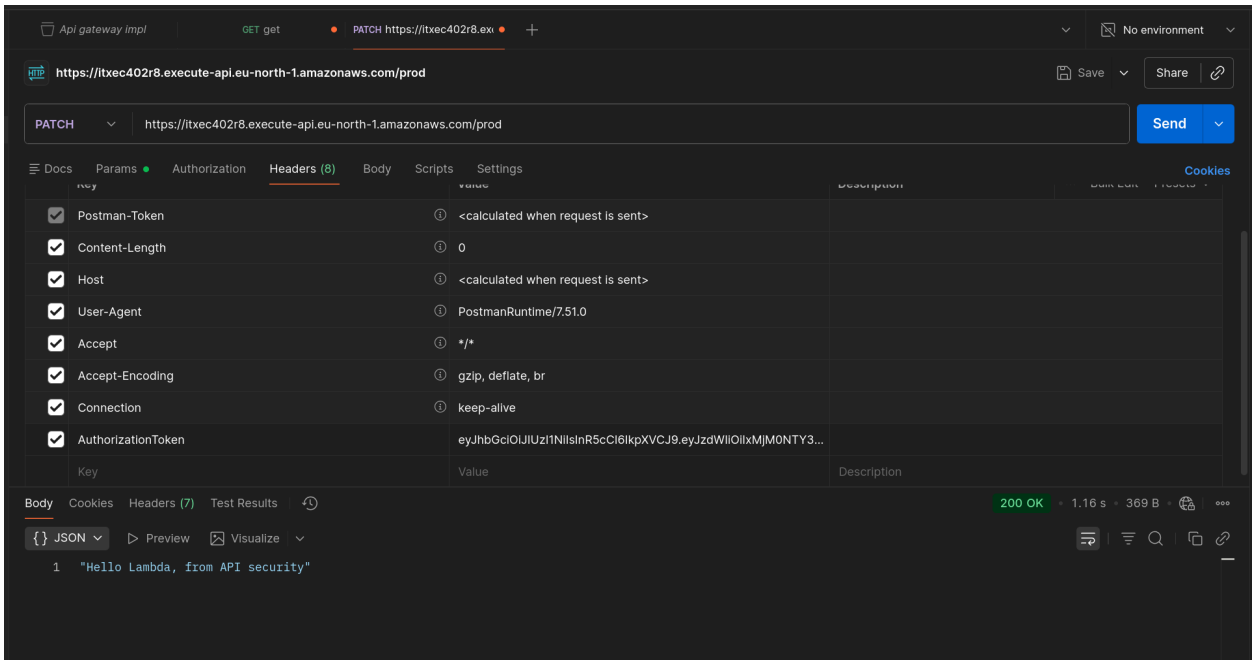


After the creation of Authorization in the API gateway, I then go back to the AP-gateway-demo and I changed the Authorization from Non to the JWT-Auth I created and redeployed the API gateway as shown below



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.





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.