Yash Mutatkar **Date: 5 Nov 2024**

Task: Build a Serverless Function with Lambda and API Gateway

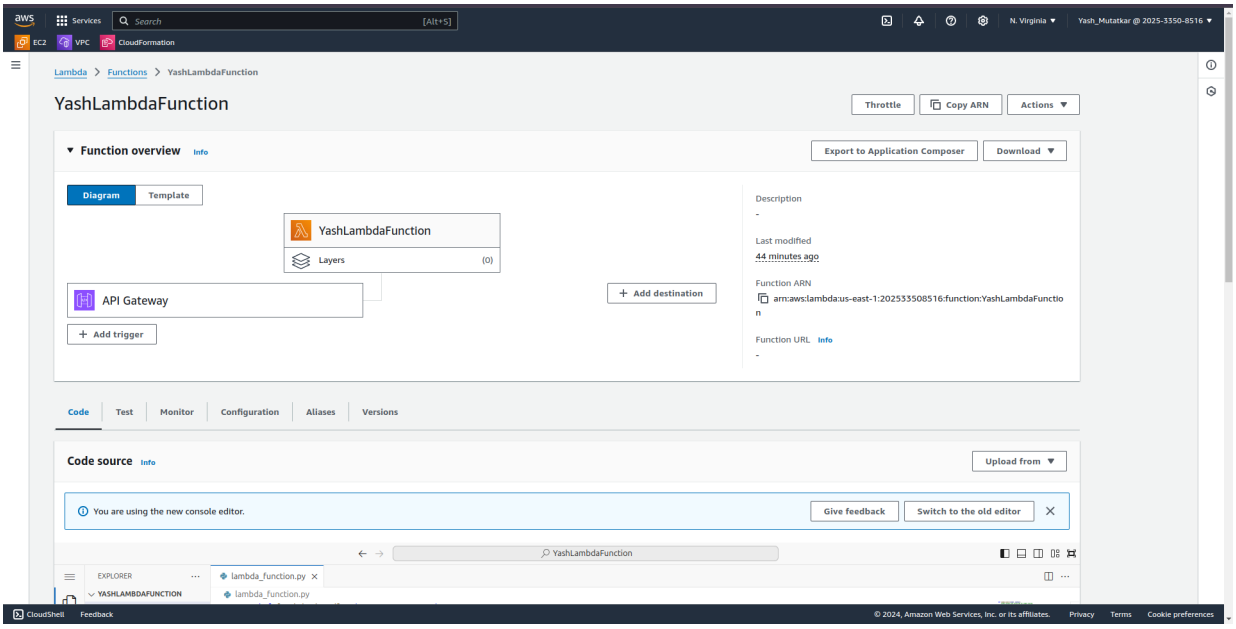
I recently took on the task of creating a serverless application using AWS, and I'm excited to share my journey!

Step 1: Creating an AWS Account

First, I headed over to the AWS website to Login in my account.

Step 2: Setting Up My Lambda Function

- **Navigating to Lambda**: In the AWS Management Console, I searched for "Lambda" and clicked on it. This is where the magic happens!
- **Creating the Function**: I clicked the "Create function" button and opted to "Author from scratch." I named my function **YashLambdaFunction** and selected **Python 3.9** as the runtime. I kept the default settings for everything else and clicked **"Create function."**



- **Writing the Code**: In the "Function code" section, I replaced the default code with my own:

```
    def lambda_handler(event, context):
# HTML content with inline CSS for simple styling
html_content = """
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Hello from Lambda</title>
<style>
body {
font-family: Arial, sans-serif;
```

```
background-color: #f0f0f5;
color: #333;
text-align: center;
padding-top: 50px;
}
.container {
display: inline-block;
padding: 20px;
border: 2px solid #4CAF50;
border-radius: 10px;
background-color: #e6ffe6;
}
h1 {
color: #4CAF50;
}
</style>
</head>
<body>
<div class="container">
<h1>Hello World,This is me Yash Mutatkar!</h1>
<p>This is my first serverless function with HTML response!</p>
</div>
</body>
</html>
"""
return {
'statusCode': 200,
'headers': {
'Content-Type': 'text/html'
},
'body': html_content
}
```

This simple code would return a friendly greeting when called.

- **Deploying the Function**: After writing the code, I hit the “Deploy” button to make my function live.

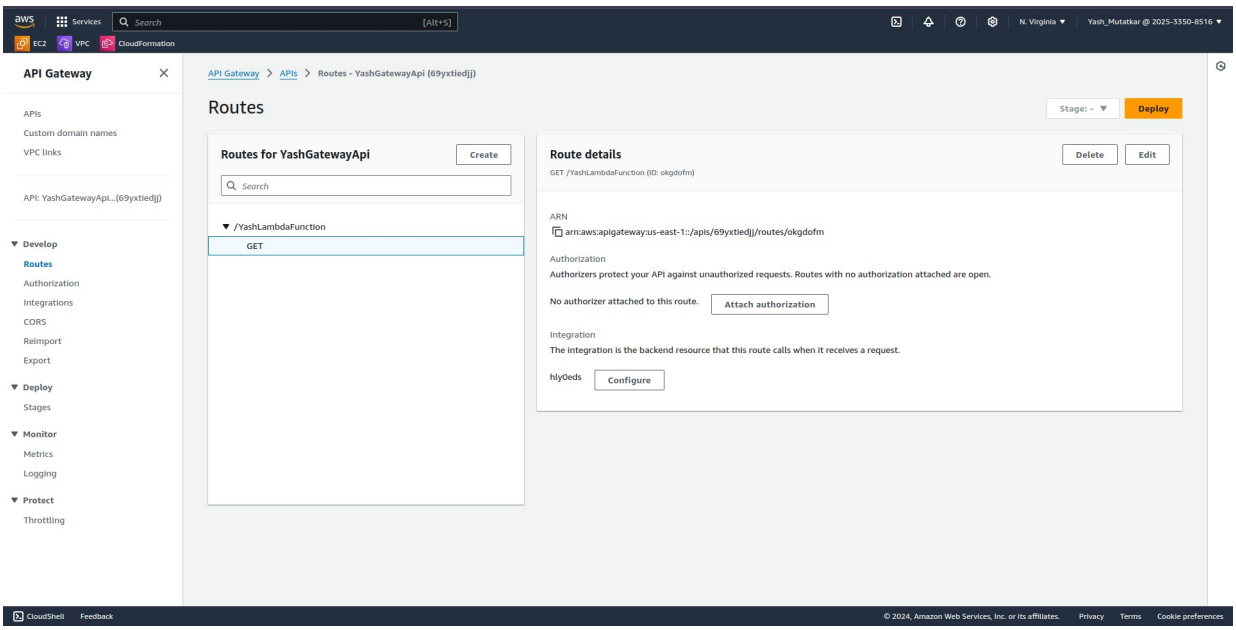
Step 3: Setting Up API Gateway

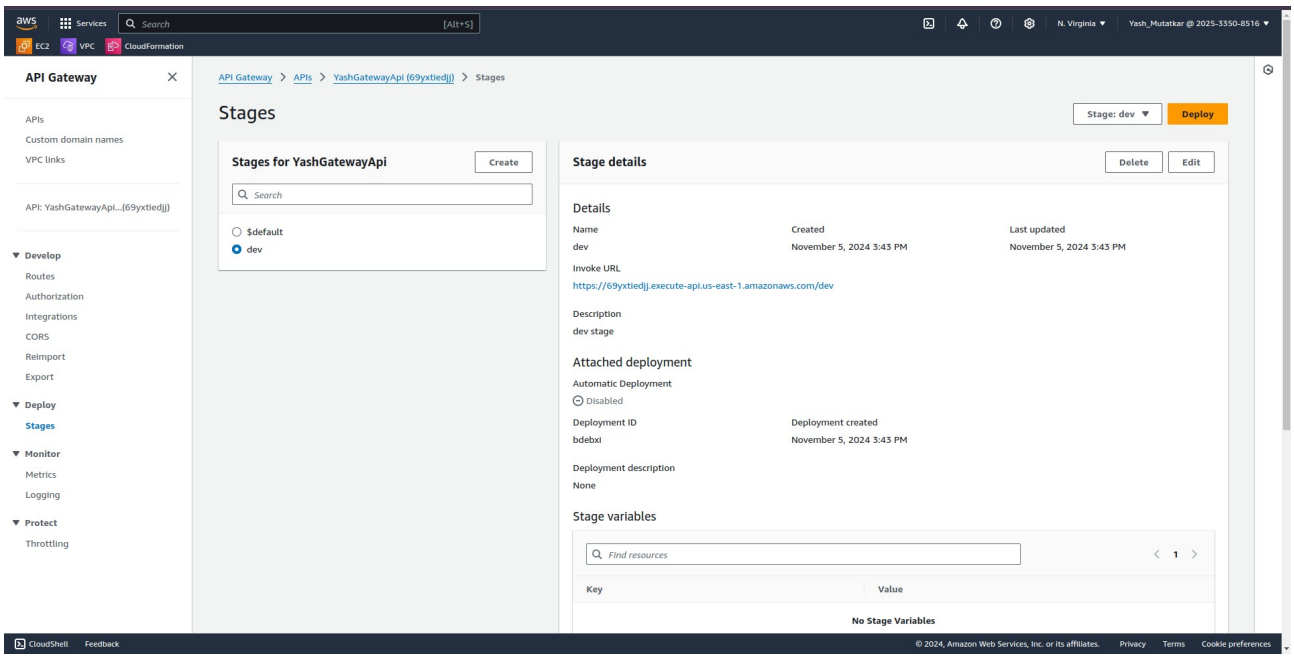
Next, I needed a way to trigger my Lambda function via an HTTP request, so I set up API Gateway:

- **Navigating to API Gateway**: I searched for “API Gateway” in the console and clicked on it.
- **Creating a New API**: I clicked “Create API” and selected “HTTP API” for a simple setup. After clicking “Build,” I was ready to configure my API.
- **Configuring the API**: In the configuration section, I added an integration for my Lambda function. I chose “Lambda” and selected my **`YashLambdaFunction`**. Once done, I clicked “Create” to finalize the integration.

- **Defining a Route**: I set up a route for my API by specifying:
    - **Method**: GET
    - **Resource path**: `/hello`

  After that, I clicked “Create” to add the route.
- **Reviewing and Creating the API**: I clicked through the options until I reached the final step, where I named my API `YashGatewayApi` and clicked “Create.”
- **Deploying the API**: Finally, I clicked on “Deployments” in the left menu, and there it was—the endpoint URL I needed to test my Lambda function!





Step 4: Testing the Endpoint

To ensure everything was working, I decided to test my new API endpoint:

1. **Using Postman**: I opened Postman, created a new GET request, and entered the endpoint URL I copied from API Gateway https://69yxtiedjj.execute-api.us-east-1.amazonaws.com/dev/YashLambdaFunction, After clicking “Send,” I was thrilled to see the response with my message!
2. **Using a Web Browser**: I also tried pasting the endpoint URL directly into my web browser. When I hit Enter, the same friendly greeting appeared on my screen.