# DevOps - Project 03

github: - github.com/manish-g0u74m Linkedin : linkedin.com/in/manish-g0u74m

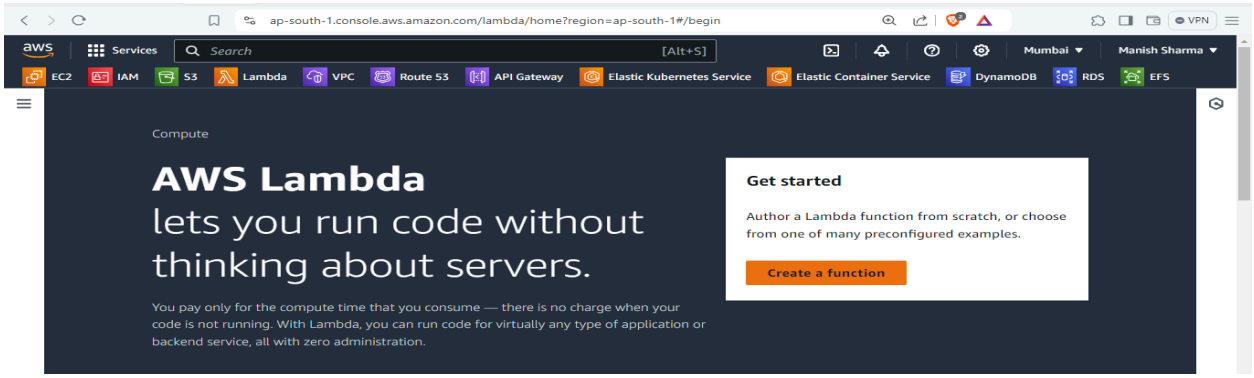
---

## Setting Up an AWS Lambda Function with URL Trigger Using API Gateway (REST API)

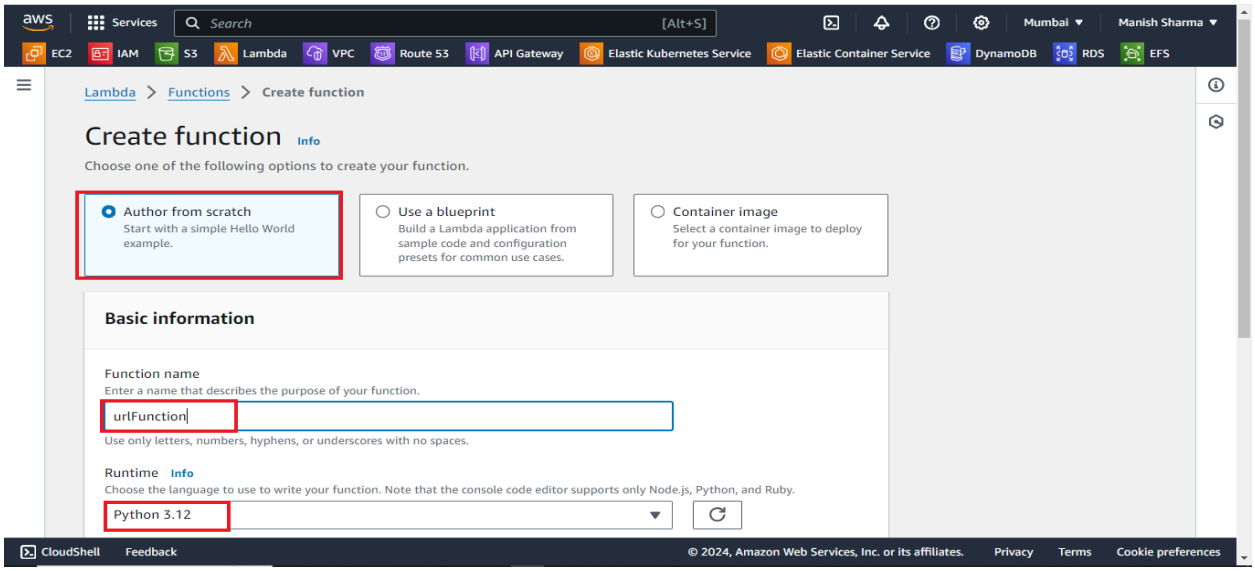
---

### Step 1. Define a Function

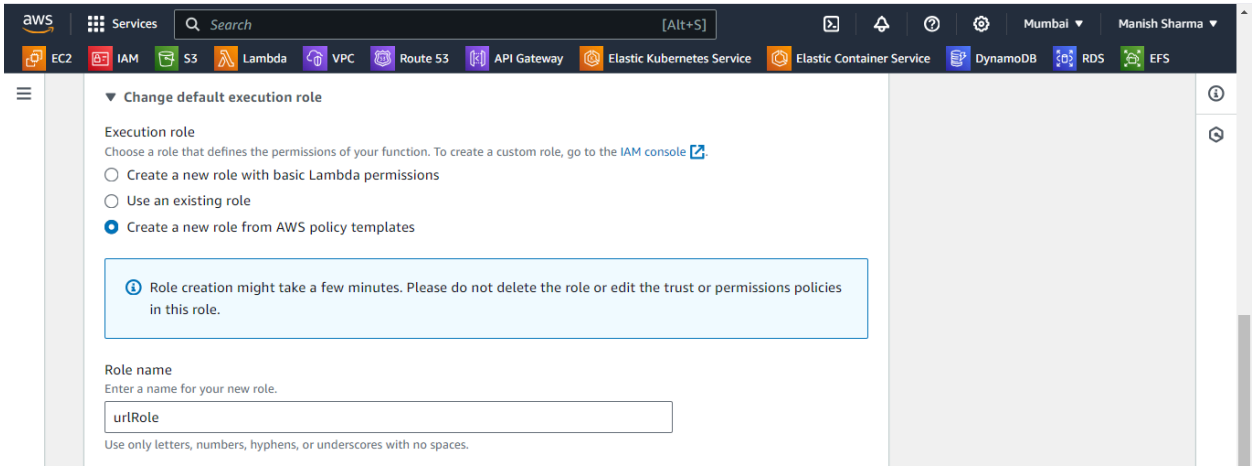
- Go to the Lambda Console and click on Create Function.



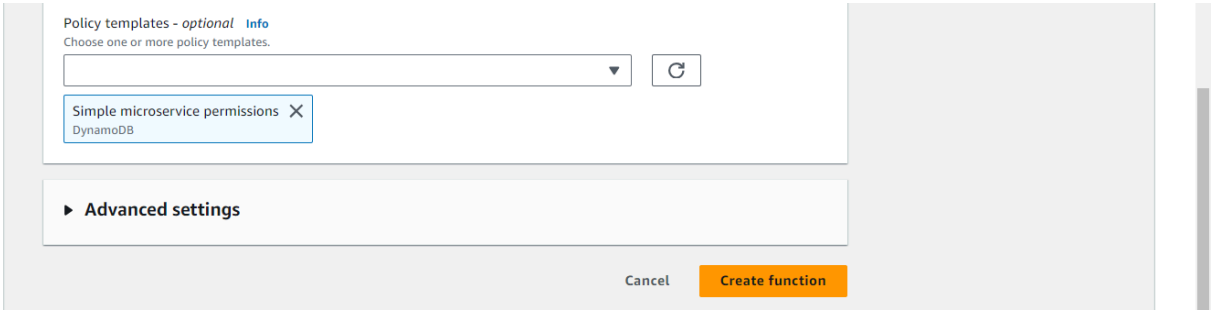

- Give the function a name, such as **urlFunction**.
- Select Python3 as the runtime.



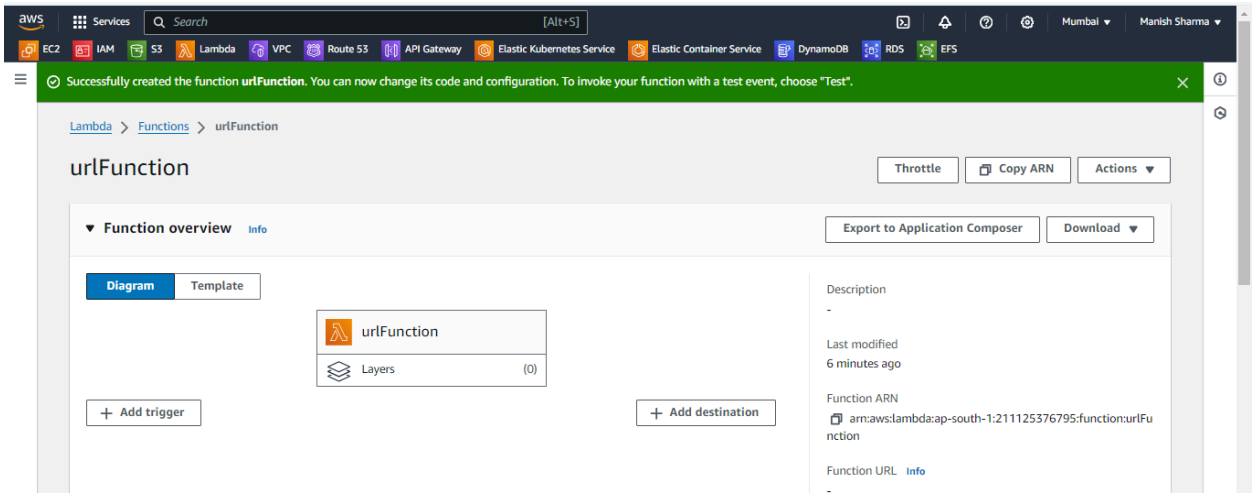
- Attach an IAM role by selecting "Create a new role from AWS policy templates".




- In Policy Templates, select Simple Microservice Permissions and Click on Create Function.

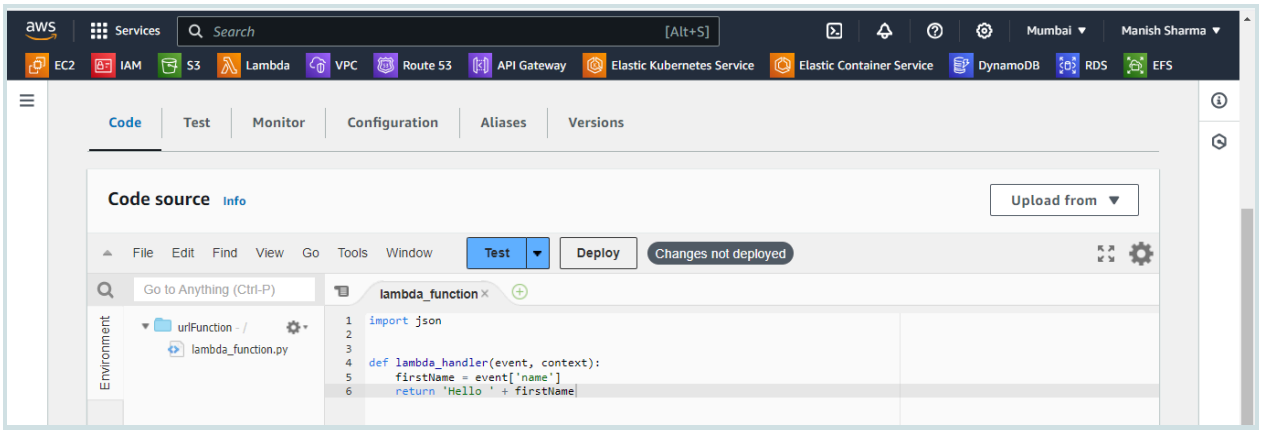



- Next, You see the urlFunction Overview Dashboard



- Next, you should see a text editor where you will insert a code similar to the following
  **Function's code**

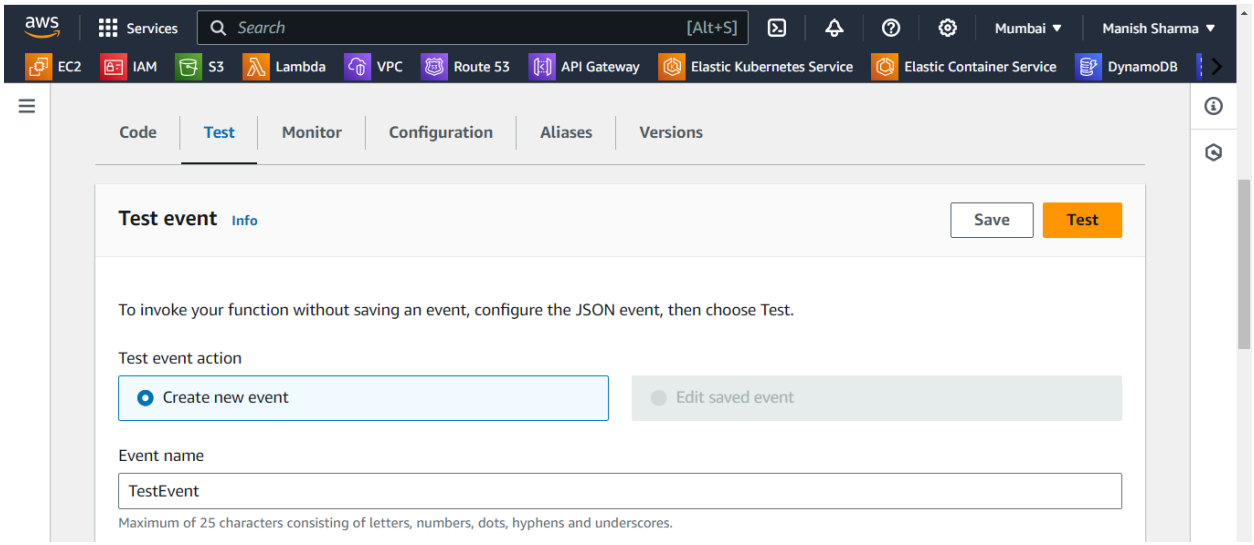
```
import json

def lambda_handler(event, context):
    firstName = event['name']
    return 'Hello ' + firstName
```




---

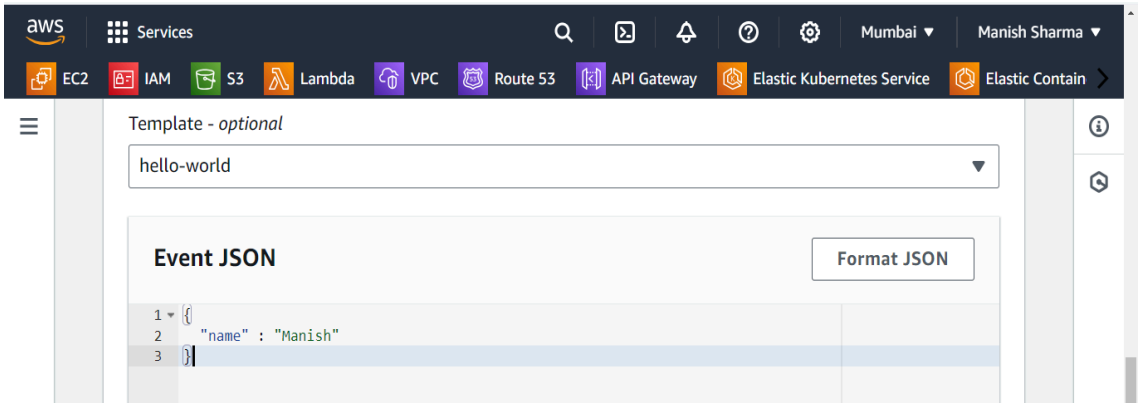
## Step 2 Define a test

1. Now let's test the function. Click on "Test".
2. Select "Create new test event"
3. Set the "Event name" to whatever you'd like. For example "**TestEvent**"

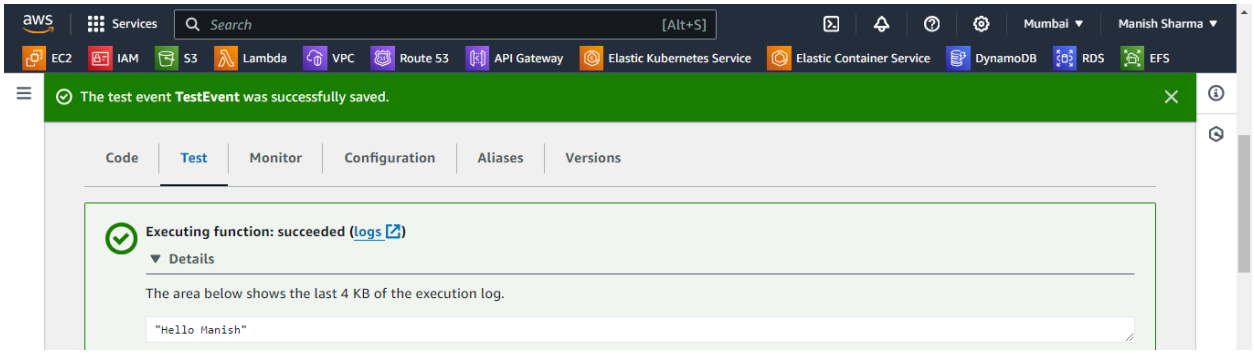



4. Provide keys to test

```
{
  "name" : "Manish"
}
```



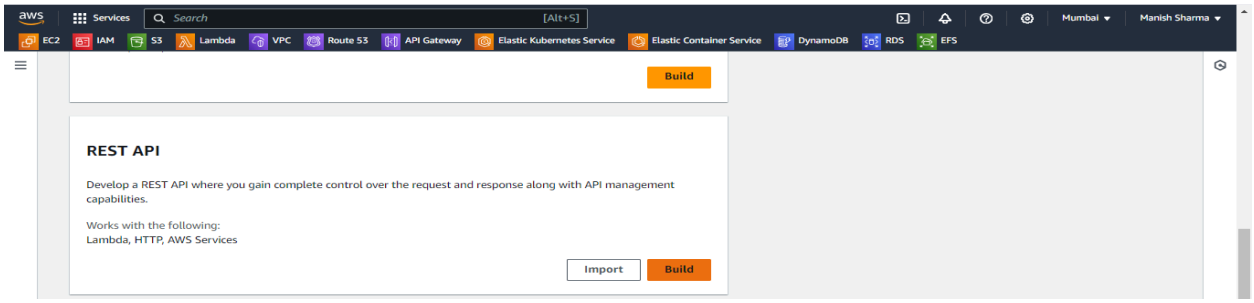

5. Click on "**Save**" and then the "`Test`" button.
6. You should see something similar to `Execution result: succeeded`



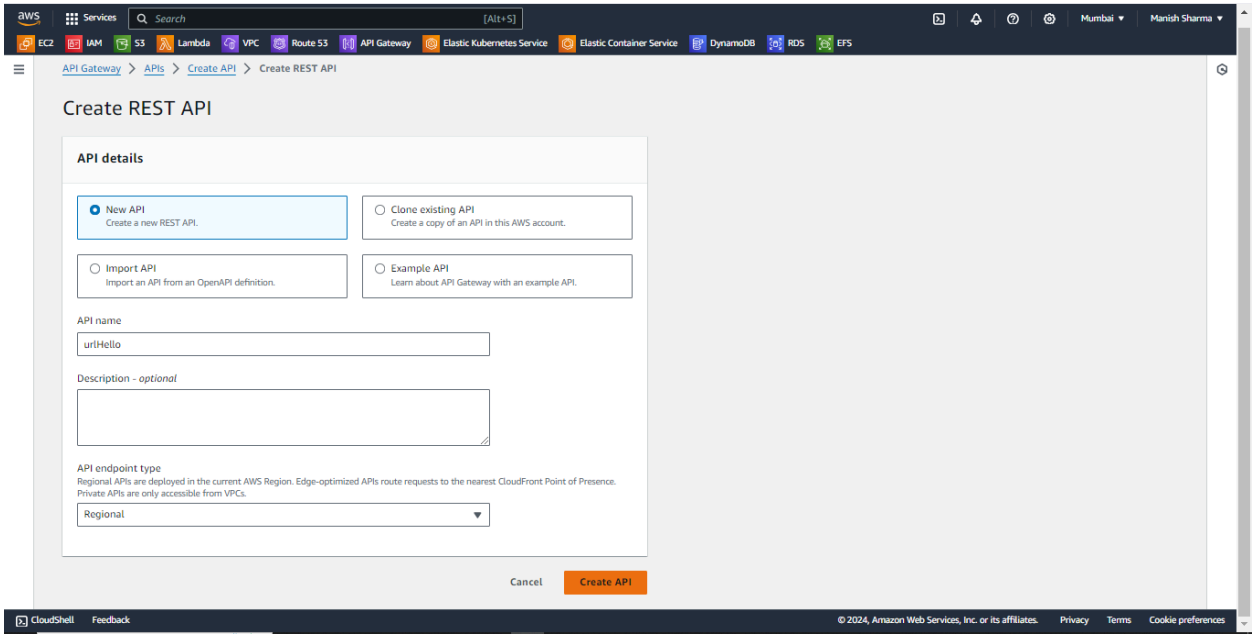

---

## Step 3 Define a Trigger

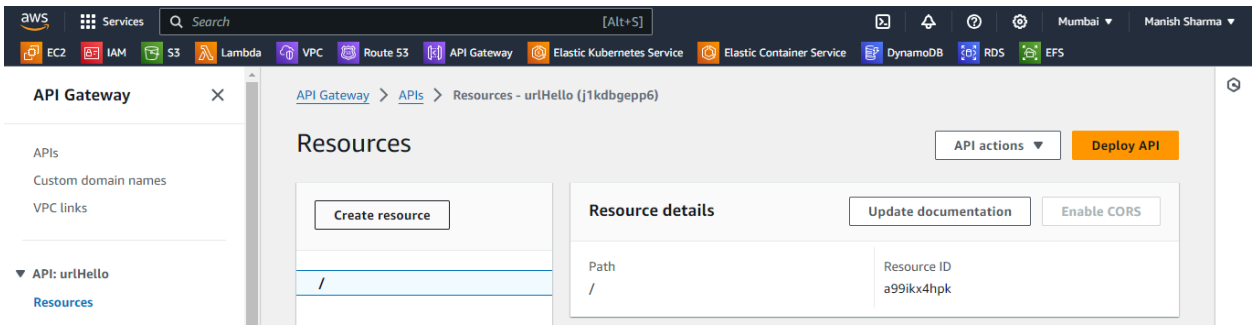
- Go to the **API Gateway Console**.
- Click on **Create API** and choose **REST API** and then click on **Build**.



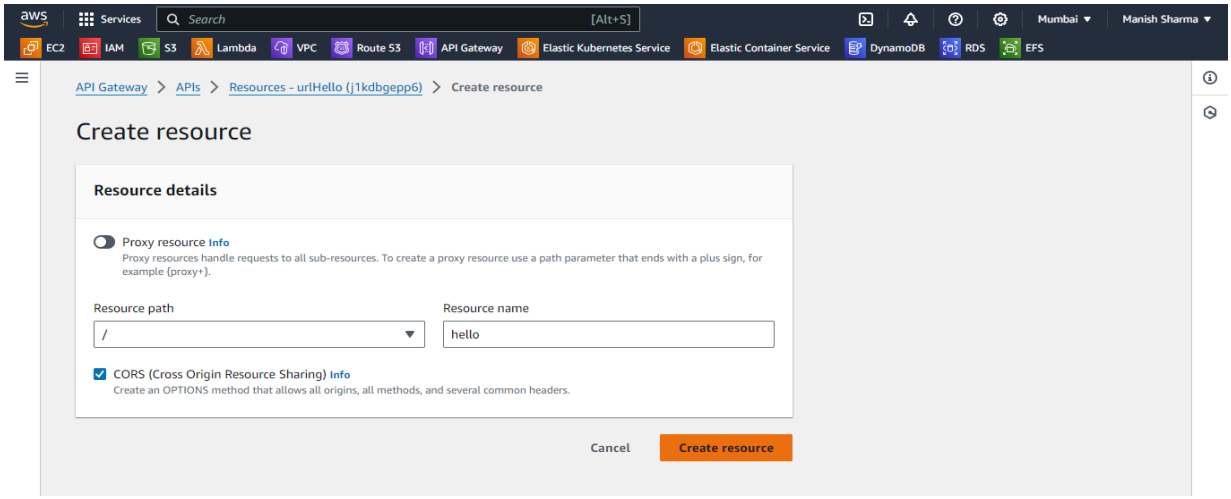

- Provide the API details and name whatever you want for the API like **“urlHello”** and a description if needed.
- Choose **Regional** as the endpoint type, then click **Create API**.



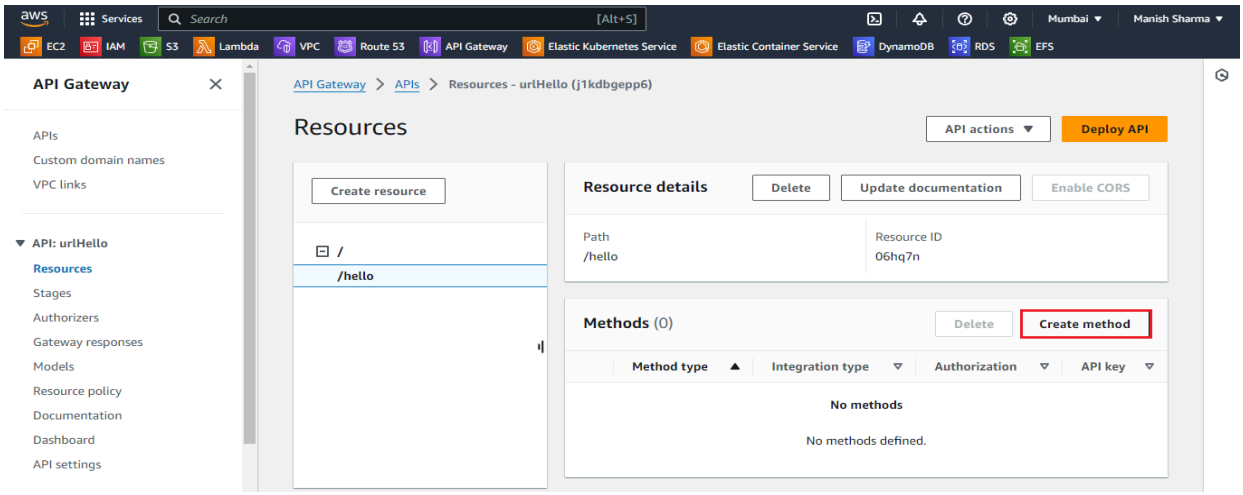

- Nest you will see Resources Dashboard and select **Create Resource**.



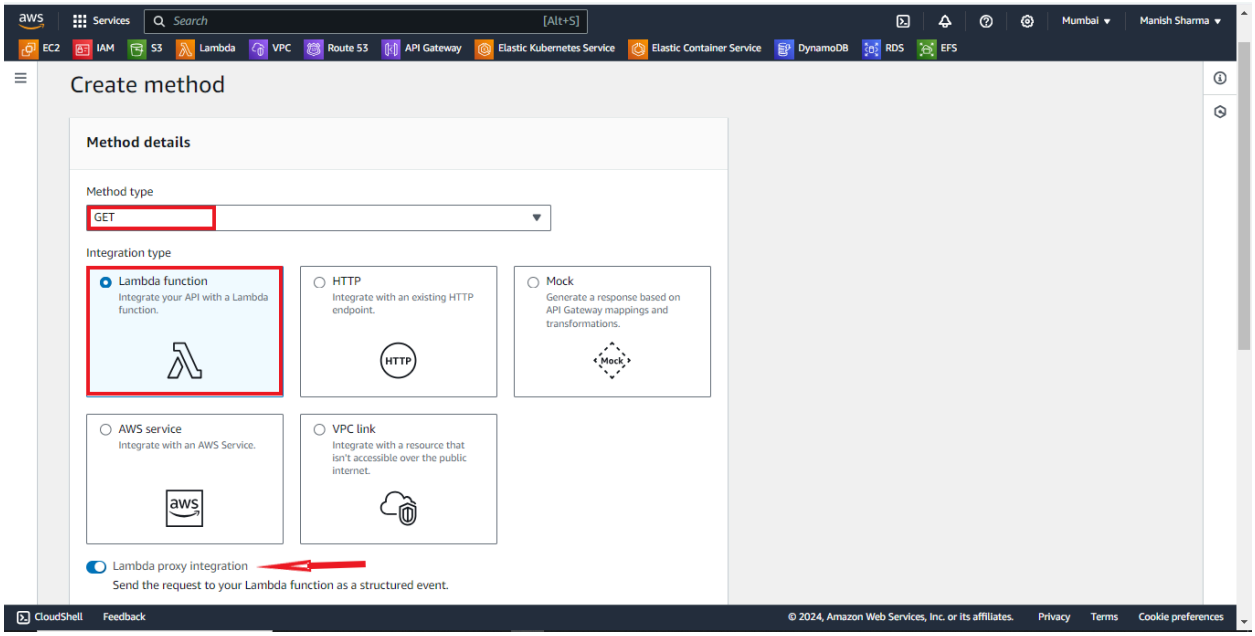

- Provide a **Resource Name** (e.g., **`hello`**) and check mark on CORS.
- Click on **Create Resource**.



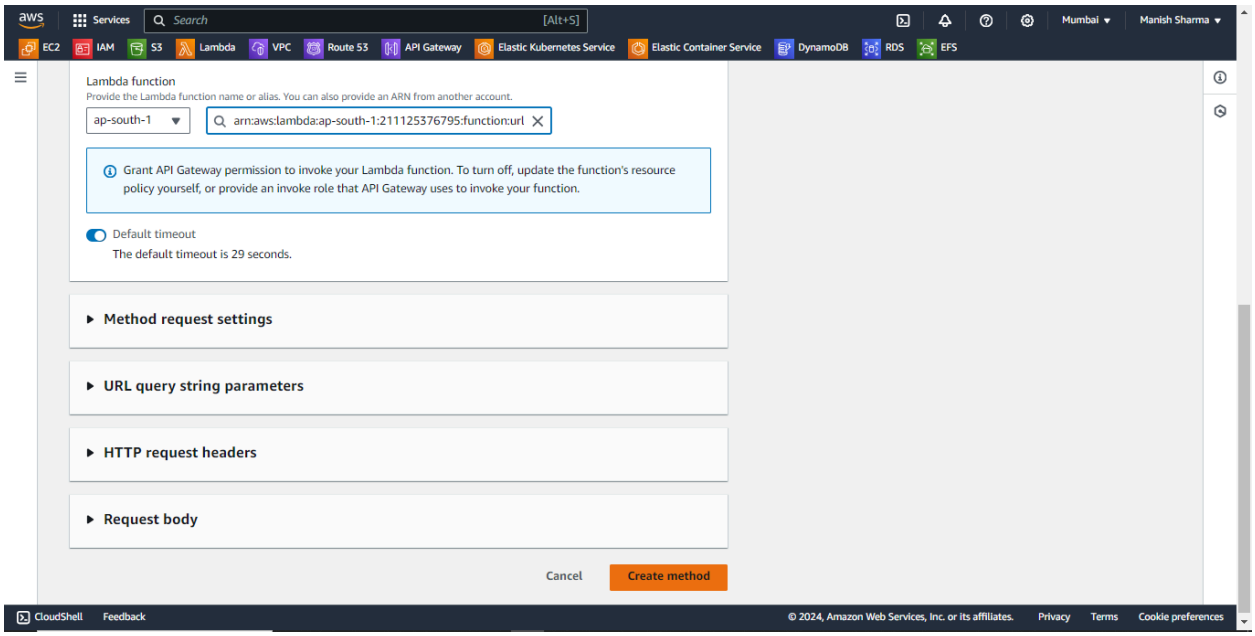

- Now select the new created resource “hello” and choose **Create Method**.



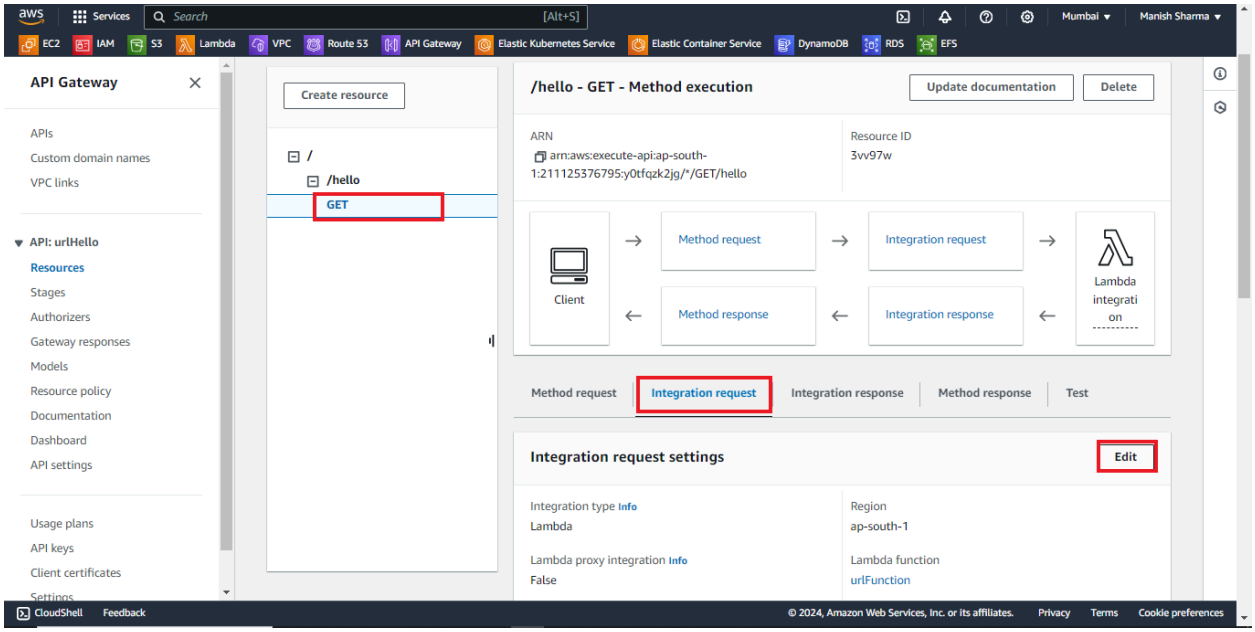

- Now Select **GET** from the dropdown and click on the checkmark to confirm.
- In the **Integration type** select **Lambda Function**.
- Ensure the **Use Lambda Proxy Integration** box is checked.




- In the **Lambda Function** field, type the name of your Lambda function (e.g., `urlFunction`), and click on **Create Method**.

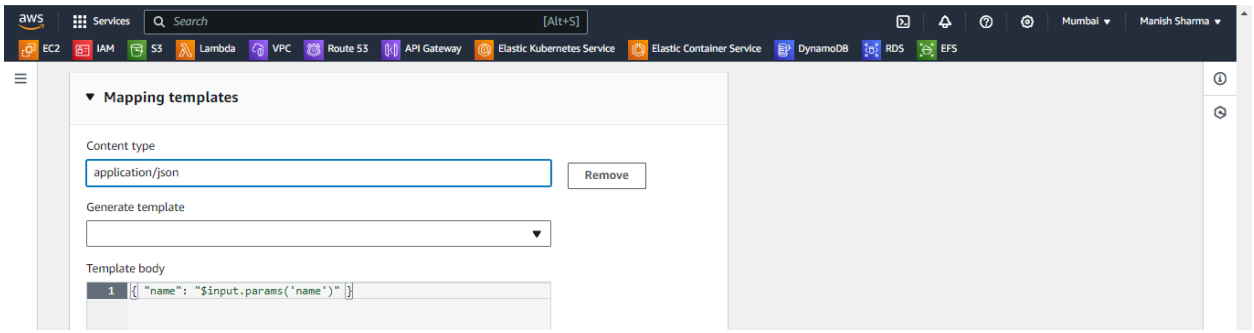



- Now click on created method (GET method) and modify "Integration Request":




- In the last section you will see a “Mapping templates” option so add the below template and click on Save.

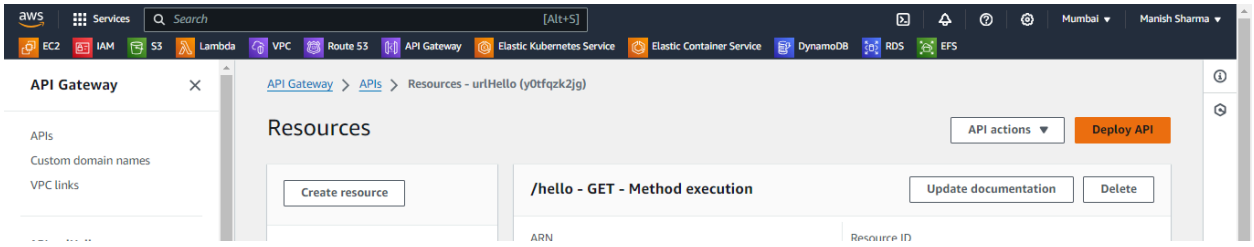
  **{ "name": "$input.params('name')" }**



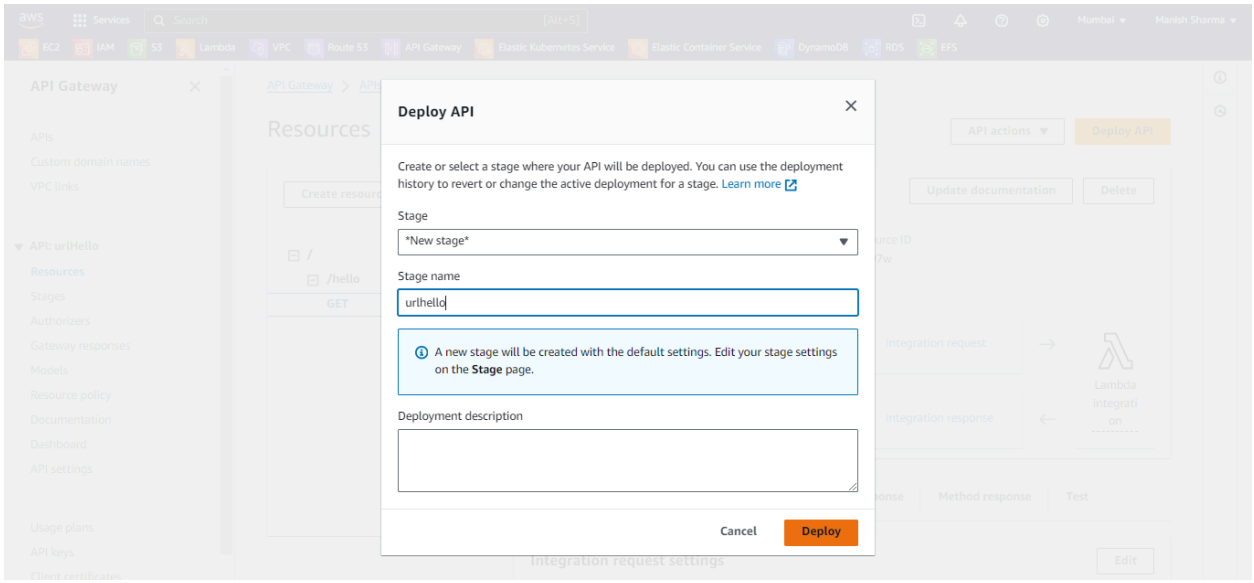

---

## Step 4 Deploy the API

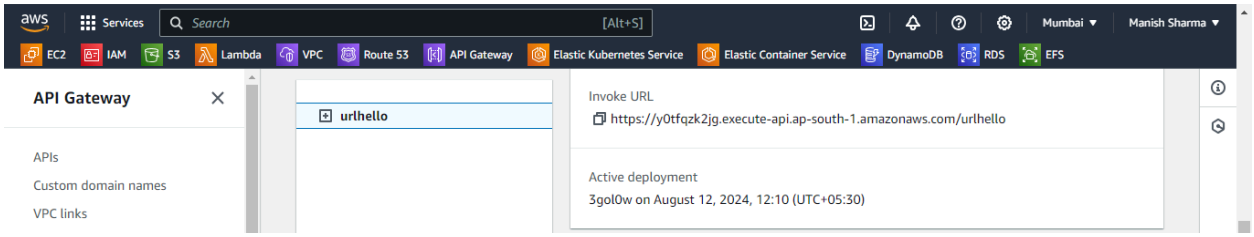
- Now, click on **Deploy API**.




- Create a new **Deployment Stage** (e.g., `urlhello`) or choose an existing one and provide a description whatever you want.
- Click **Deploy**.



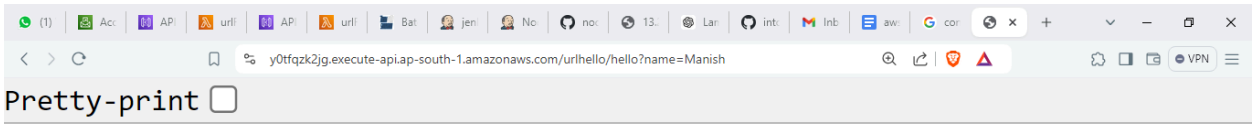

- After deployment, You'll see an **Invoke URL** for the API. This URL will look something like `https://<api-id>.execute-api.<region>.amazonaws.com/urlhello`.




---

## Step 5 Final step Test the Function

- To test the function, you can click on. You might have to modify it to include the input so it looks similar to this: .../hello?name=Manish
- `Open this URL in your browser, and you should see Hello Manish.`




```
"Hello Manish"
```

**Connect with me on LinkedIn: [linkedin.com/in/manish-g0u74m](linkedin.com/in/manish-g0u74m)**