

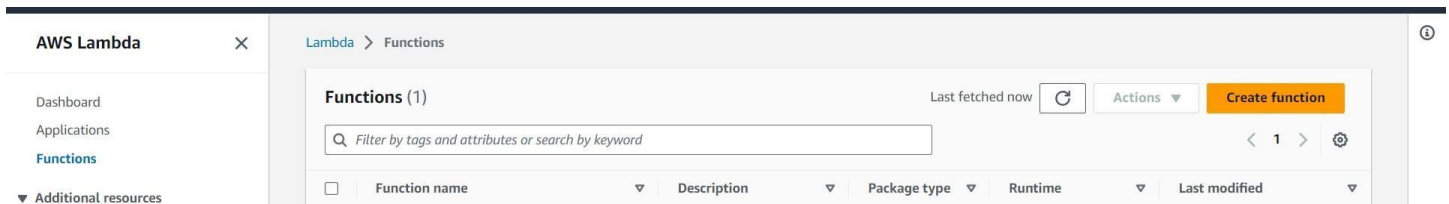
## Task-1

### Serverless Function on AWS

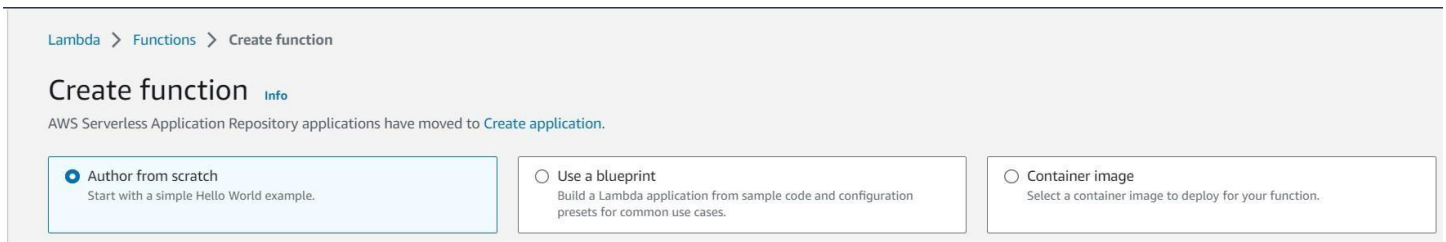
**Aim:** Create a serverless function using AWS Lambda, which allows you to run code without provisioning or managing servers. Develop a simple function (e.g., a function that generates random numbers) and trigger it through API Gateway. This project introduces you to serverless computing and event-driven architectures.

#### Steps 1: Create an AWS Lambda Function

- a) Navigate to the Lambda service.
- b) Click the "Create function" button.



- c) Choose "Author from scratch" and fill in the following details:



- d) Function name: "RendomNoGen".

#### Basic information

##### Function name

Enter a name that describes the purpose of your function.

RendomNoGen

Use only letters, numbers, hyphens, or underscores with no spaces.

- e) Runtime: Choose the runtime you prefer (e.g., Python 3.9)

#### Runtime [Info](#)

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.9



#### Architecture [Info](#)

Choose the instruction set architecture you want for your function code.

☒ x86\_64

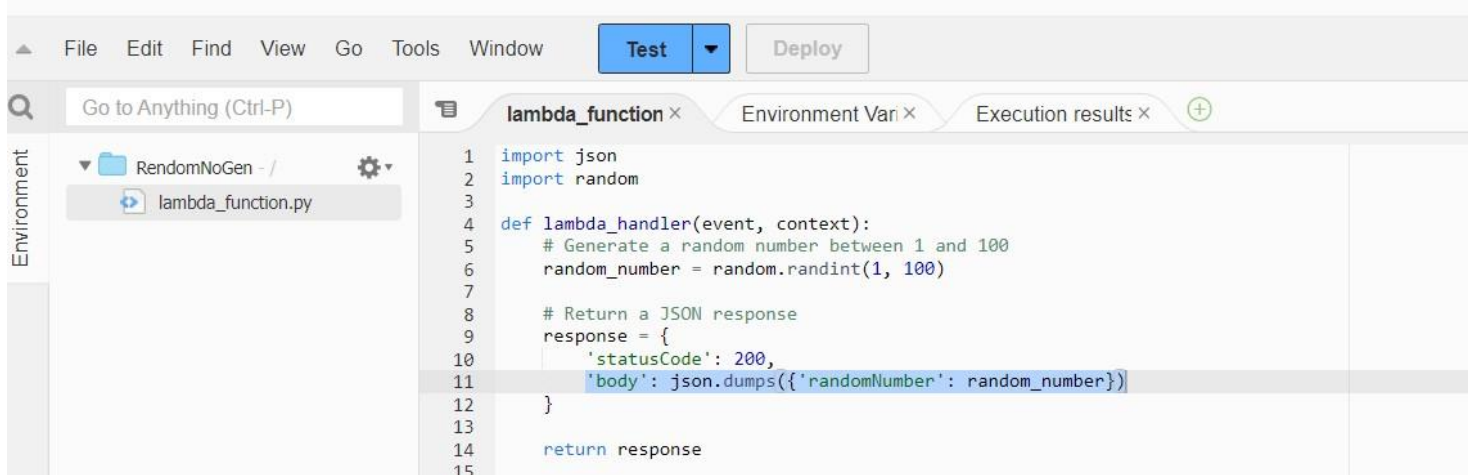
☐ arm64

f) Click the "Create function" button.

g) In the "Function code" section, you can paste the following Python code that generates a random number:

```
import json import random def
lambda_handler(event, context):
    # Generate a random number between 1 and 100
    random_number = random.randint(1, 100)

    # Return a JSON response
    response = { 'statusCode': 200,
                  'body': json.dumps({'randomNumber': random_number})
    }
    return response
```



h) Update Function Configuration (Optional):

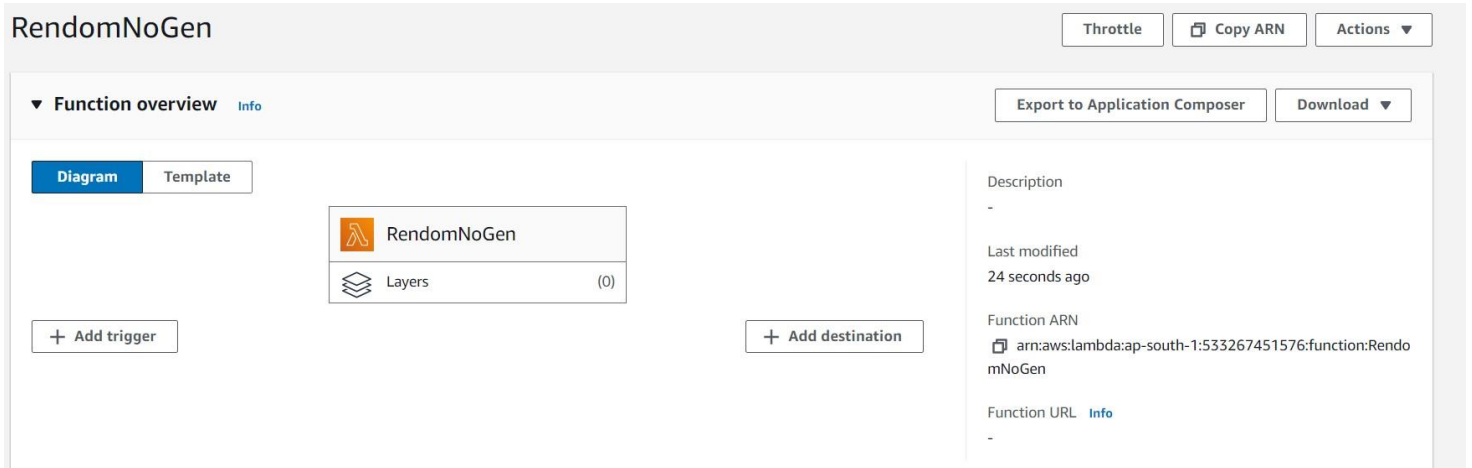
- (Optional) You can adjust memory and timeout settings based on your needs under "Configuration". i)

Deploy the Function:

- Click on "Actions" and select "Deploy".

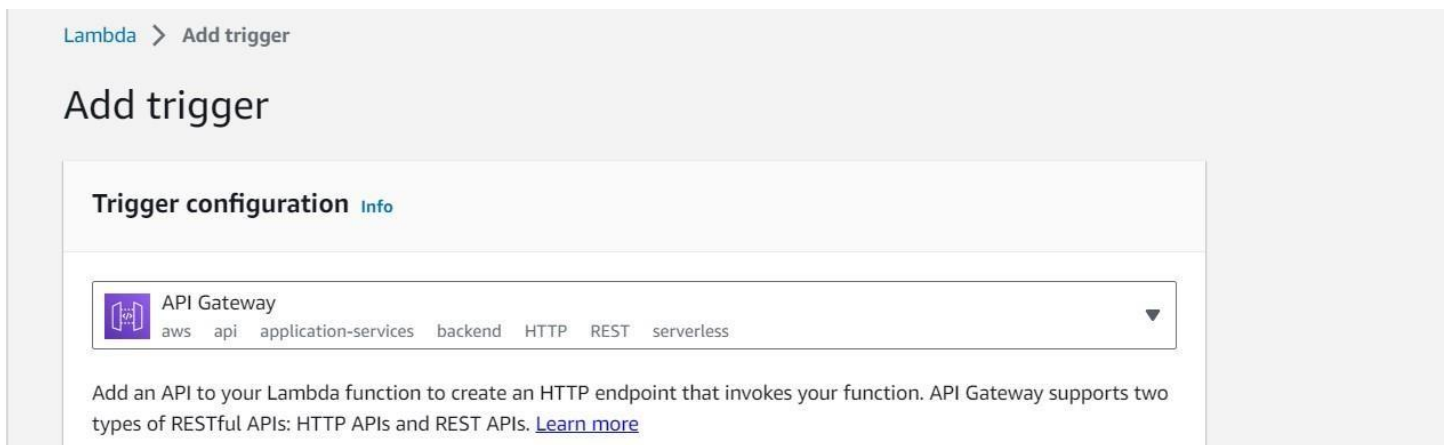
## Step 2: Set Up API Gateway

a) Once the Lambda function is created, click on the "Add trigger" button.



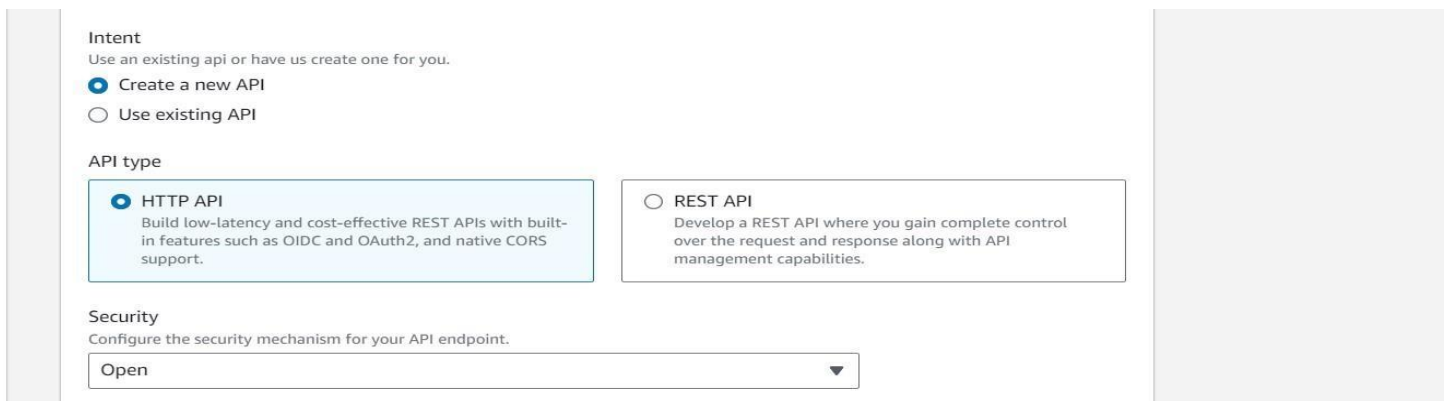
The screenshot shows the 'RandomNoGen' function overview in the AWS Lambda console. At the top, there are buttons for 'Throttle', 'Copy ARN', and 'Actions'. Below this, the 'Function overview' tab is selected, showing a 'Diagram' and 'Template' view. The 'Diagram' view displays the function name 'RandomNoGen' and a list of layers (0). There are buttons for '+ Add trigger' and '+ Add destination'. On the right, the 'Description' section shows the function's last modified time (24 seconds ago), its ARN (arn:aws:lambda:ap-south-1:533267451576:function:RandomNoGen), and its URL.

b) Choose "API Gateway" as the trigger type.



The screenshot shows the 'Add trigger' configuration page in the AWS Lambda console. The 'Trigger configuration' section is active, showing a dropdown menu with 'API Gateway' selected. Below the dropdown, there is a description of API Gateway and a link to 'Learn more'.

c) In the "Configure triggers" section, choose "Create an API" and fill in the following details:



The screenshot shows the 'Configure triggers' section in the AWS Lambda console. It includes a section for 'Intent' with radio buttons for 'Create a new API' (selected) and 'Use existing API'. Below this, the 'API type' section has two options: 'HTTP API' (selected) and 'REST API'. The 'Security' section at the bottom has a dropdown menu with 'Open' selected.

d) API name: RandomNoGen-API

### ▼ Additional settings

#### API name

Choose a name for your API. API names don't need to be unique.

RandomNoGen-API

e) Click the "Add" button to create the API Gateway.

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger.  
[Learn more](#) about the Lambda permissions model.

Cancel

Add

### Step 3: Test and Deploy the code.

a) Select Test event action as "Create new event" and Event name: "TestEvent" and Click on Save Button.

#### Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

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

#### Test event action

☒ Create new event

☐ Edit saved event

#### Event name

TestEvent

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

b) Then test the code and deploy it.

#### Code source

Info

Environment

Environment Variables: Environment Variables, lambda\_function, Execution results

Status: Succeeded

Test Event Name (unsaved) test event

Response

```
{
  "statusCode": 200,
  "body": "{\"random_number\": 7}"
}
```

Function Logs

START RequestId: e29531f3-1e96-4ede-8983-515c946e29e8 Version: \$LATEST

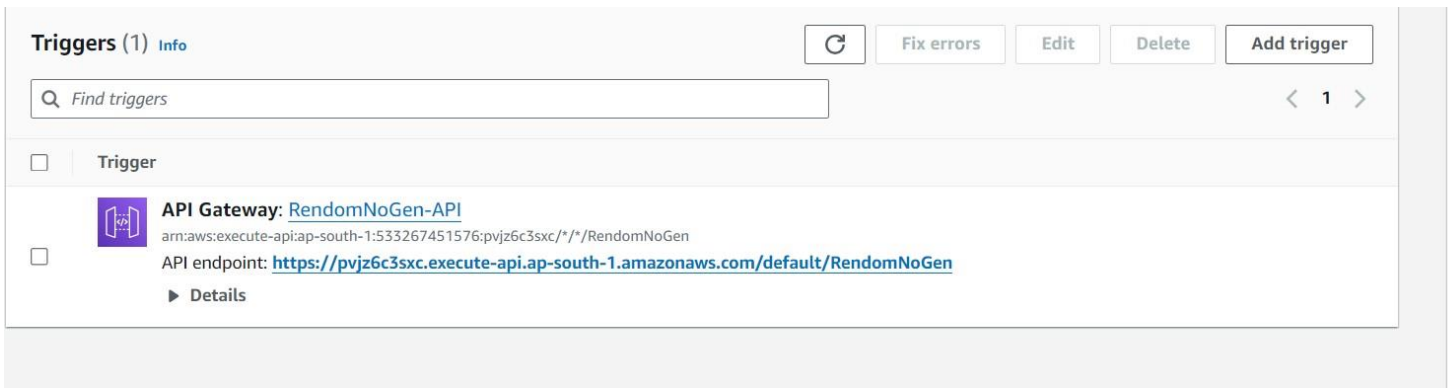
END RequestId: e29531f3-1e96-4ede-8983-515c946e29e8

REPORT RequestId: e29531f3-1e96-4ede-8983-515c946e29e8 Duration: 2.11 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memory Used: 31 MB

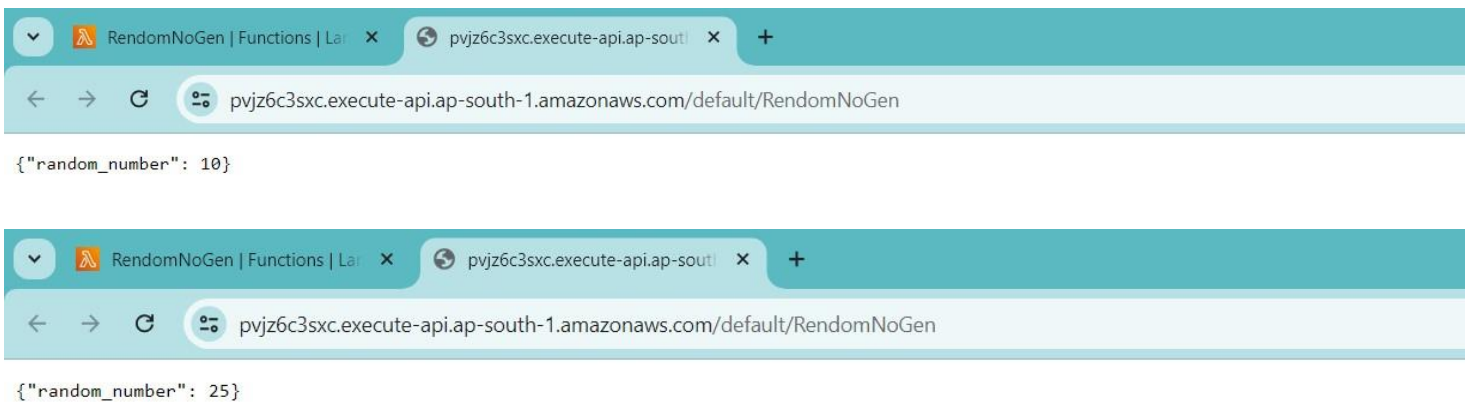
Request ID

e29531f3-1e96-4ede-8983-515c946e29e8

c) Go to Configuration and in trigger click on API endpoint ""



d) The result



**Conclusion:** We have successfully created a serverless function using AWS Lambda and triggered it through API Gateway. This simple example demonstrates the power of serverless computing and event-driven architectures. We can further enhance this project by adding more complex logic to your Lambda function and implementing different types of triggers and event sources.