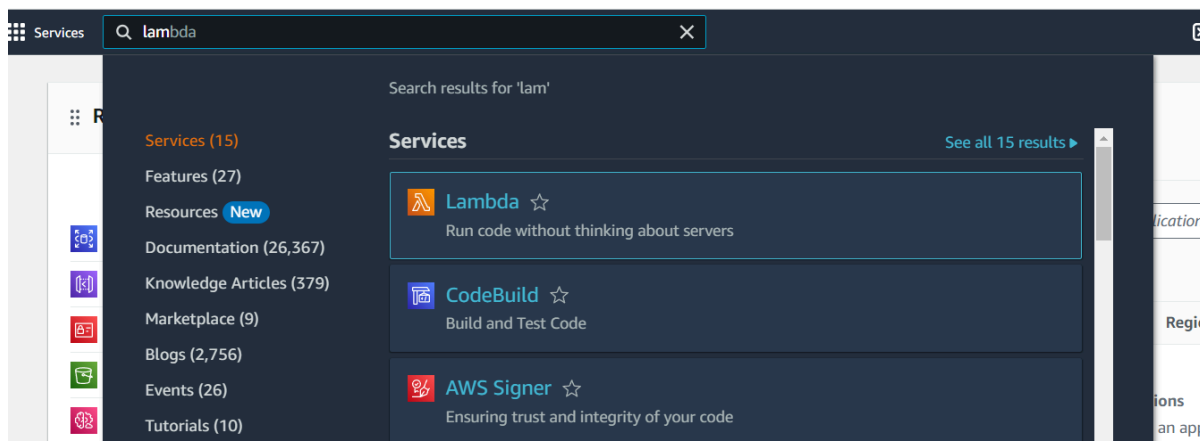


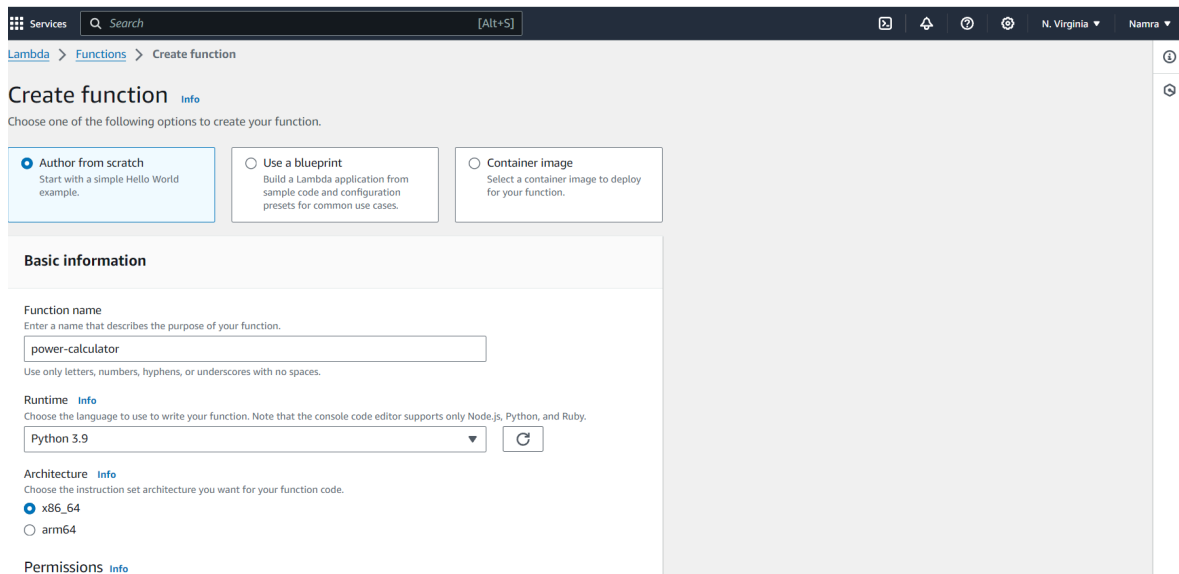
## Task-2 Serverless Function on AWS

**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 triggers it through API Gateway.

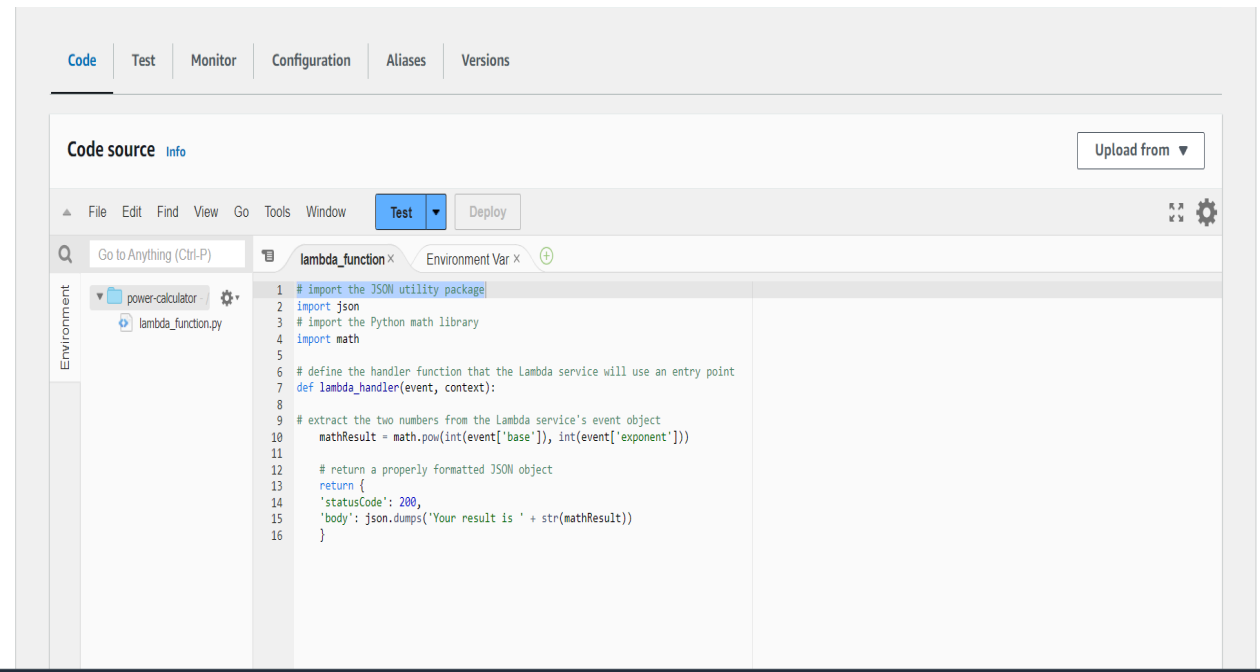
### Step 1: Navigate to the AWS console and in their search AWS Lambda



### Step 2 : Click on create function,give name and choose runtime enviroment



### Step 3: Add code to lambda function



### Step 4: configure test case for testing code

Test event action

☒ Create new event ☐ Edit saved event

Event name

testing

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](#)

Template - optional

hello-world

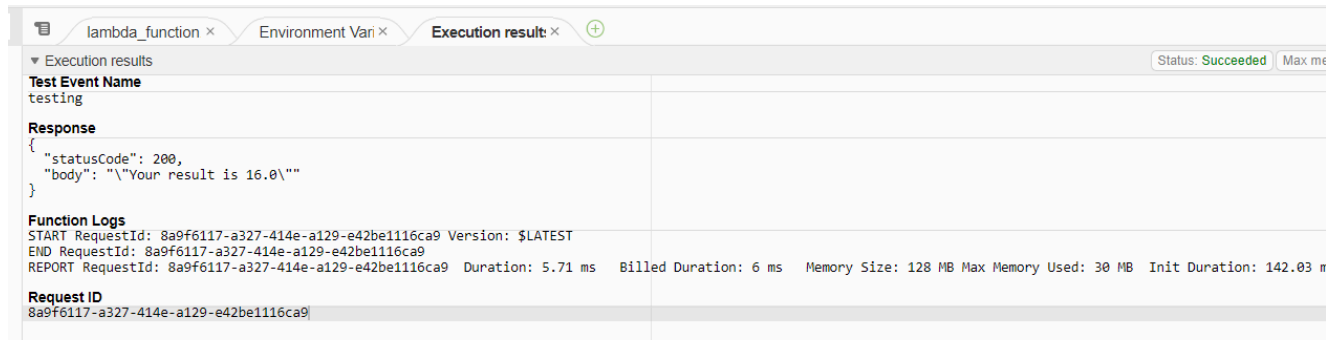
Event JSON

Format JSON

```
1 {
2   "base": 2,
3   "exponent": 4
4 }
5
```

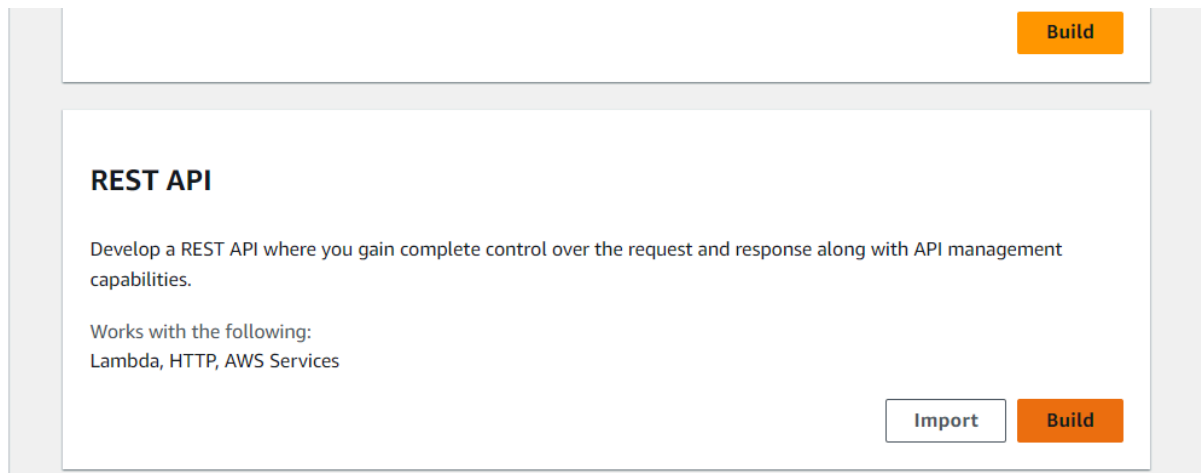
Cancel Invoke Save

## Step 5: test successful code working properly



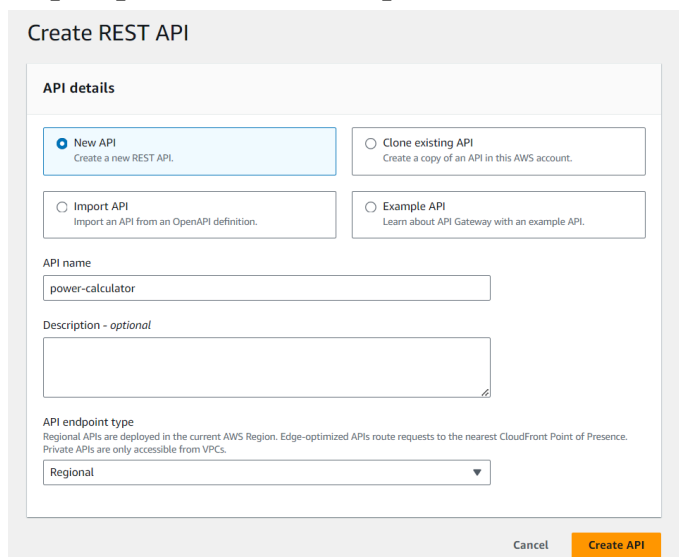
The screenshot shows the AWS Lambda console with the 'Execution results' tab selected. The status is 'Succeeded'. The 'Test Event Name' is 'testing'. The 'Response' is a JSON object: `{ "statusCode": 200, "body": "\"Your result is 16.0\"" }`. The 'Function Logs' show the execution details: `START RequestId: 8a9f6117-a327-414e-a129-e42be1116ca9 Version: $LATEST`, `END RequestId: 8a9f6117-a327-414e-a129-e42be1116ca9`, and `REPORT RequestId: 8a9f6117-a327-414e-a129-e42be1116ca9 Duration: 5.71 ms Billed Duration: 6 ms Memory Size: 128 MB Max Memory Used: 30 MB Init Duration: 142.03 ms`. The 'Request ID' is `8a9f6117-a327-414e-a129-e42be1116ca9`.

## Step 6: Create Api gateway for triggering Lambda function,select rest api



The screenshot shows the AWS API Gateway console. The 'REST API' section is highlighted. It describes a REST API where you gain complete control over the request and response along with API management capabilities. It mentions that it works with the following: Lambda, HTTP, AWS Services. There are 'Import' and 'Build' buttons at the bottom right.

## Step 7: provide name of api



The screenshot shows the 'Create REST API' form. The 'API details' section is active. The 'New API' option is selected, which creates a new REST API. The 'API name' field is filled with 'power-calculator'. The 'Description - optional' field is empty. The 'API endpoint type' is set to 'Regional'. The 'Create API' button is highlighted.

**Step 8: after api created click on create method for this we need post method for providing input to function and select service to integrate with**

## Create method

### Method details

Method type

POST ▼

Integration type

☒ Lambda function

Integrate your API with a Lambda function.



☐ HTTP

Integrate with an existing HTTP endpoint.



☐ Mock

Generate a response based on API Gateway mappings and transformations.



☐ AWS service

Integrate with an AWS Service.



☐ VPC link

Integrate with a resource that isn't accessible over the public internet.



☐ Lambda proxy integration

Send the request to your Lambda function as a structured event.

Lambda function

Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1 ▼

🔍 arn:aws:lambda:us-east-1:992382839514:function:pow X

**Step 9: after that click on deploy api and your api will be created ,copy that invoke url it will be needed**

API Gateway > APIs > power-calculator (u4pfh4n8t4) > Stages

## Stages

Stage actions ▼ Create stage

dev

### Stage details [Info](#)

Edit

Stage name	dev	Rate <a href="#">Info</a>	-	Web ACL	-
Cache cluster <a href="#">Info</a>	Inactive	Burst <a href="#">Info</a>	-	Client certificate	-
Default method-level caching	Inactive				

Invoke URL

<https://u4pfh4n8t4.execute-api.us-east-1.amazonaws.com/dev>

Active deployment

w5eqka on June 24, 2024, 11:56 (UTC+05:30)

### Logs and tracing [Info](#)

Edit

CloudWatch logs	Detailed metrics	X-Ray tracing
Inactive	Inactive	Inactive

**Step 10: go to dynamo db and create table as we want to store our result in table**

DynamoDB > Tables > Create table

## Create table

### Table details [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

**Table name**  
This will be used to identify your table.

power-calculator

Between 3 and 255 characters, containing only letters, numbers, underscores (\_), hyphens (-), and periods (.).

**Partition key**  
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

ID

String ▼

1 to 255 characters and case sensitive.

**Sort key - optional**  
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Enter the sort key name

String ▼

1 to 255 characters and case sensitive.

**Step 11 : go to created table and inside general information their is table arn copy that it will require for allowing lambda function to make changes to table**

The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with a search bar and a list of tables: 'power-calculator' (selected) and 'powerofmath'. The main panel displays the 'General information' tab for the 'power-calculator' table. At the top, there's a notification banner about protecting the table from accidental writes and deletes, with an 'Edit PITR' button. Below this, the 'General information' section is divided into two parts. The top part shows: Partition key ID (String), Sort key (-), Capacity mode (Provisioned), Table status (Active), Alarms (No active alarms), Point-in-time recovery (PITR) (Off), and Resource-based policy (Not active). The bottom part, under 'Additional info', shows: Table class (DynamoDB Standard), Indexes (0 globals, 0 locals), DynamoDB stream (Off), Time to Live (TTL) (Off), Replication Regions (0 Regions), Encryption (Owned by Amazon), Date created (June 24, 2024, 12:02:54 (UTC+05:30)), and Deletion protection (Off). A green 'ARN copied' notification is visible. At the bottom, the Amazon Resource Name (ARN) is displayed: 'arn:aws:dynamodb:us-east-1:992382839514:table/power-calculator'.

**Step 12 : go to lambda function and inside that go to configuration click on role it will navigate you to IAM in that along with lambda function attach inline policy for allowing lambda function to access dynamo db table**

The screenshot shows the AWS IAM console 'Create policy' wizard. The breadcrumb navigation is 'IAM > Roles > Power-calculator > Create policy'. The current step is 'Step 1: Specify permissions'. The main area is titled 'Specify permissions' with a sub-header 'Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.' Below this is the 'Policy editor' section, which contains a JSON editor with the following content:

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": [
8         "dynamodb:PutItem",
9         "dynamodb>DeleteItem",
10        "dynamodb:GetItem",
11        "dynamodb:Scan",
12        "dynamodb:Query",
13        "dynamodb:UpdateItem"
14      ],
15       "Resource": "arn:aws:dynamodb:us-east-1:992382839514:table/power-calculator"
16     }
17   ]
18 }
```

**Step 13 : go to lambda function again test the code and then go check dynamo db table values will be added to it**

Completed. Read capacity units consumed: 0.5

Items returned (1)

Actions

Create item

ID (String)	LatestGreetingTime
16.0	Mon, 24 Jun 2024 06:45:54 +0000

**Step 14 : go to AWS amplify for hosting your browser in it**

All apps / Create new app

Support Docs

Choose create method

Start a manual deployment

Start a manual deployment

Manually upload objects to deploy your app. You can choose to drag and drop the artifacts directly, pull a zip from an existing S3 bucket or any other URL.

App name

power-calculator

Branch name

staging

Zip the contents of your build output, not the top level folder

Make sure you zip the contents of your build output and not the top level folder. For example, if your build output generates a folder named "build" or "public", first navigate into that folder, select all of the contents, and zip it from there.

Read more

Method

Drag and drop

Amazon S3

Any URL

index.zip

Uploaded

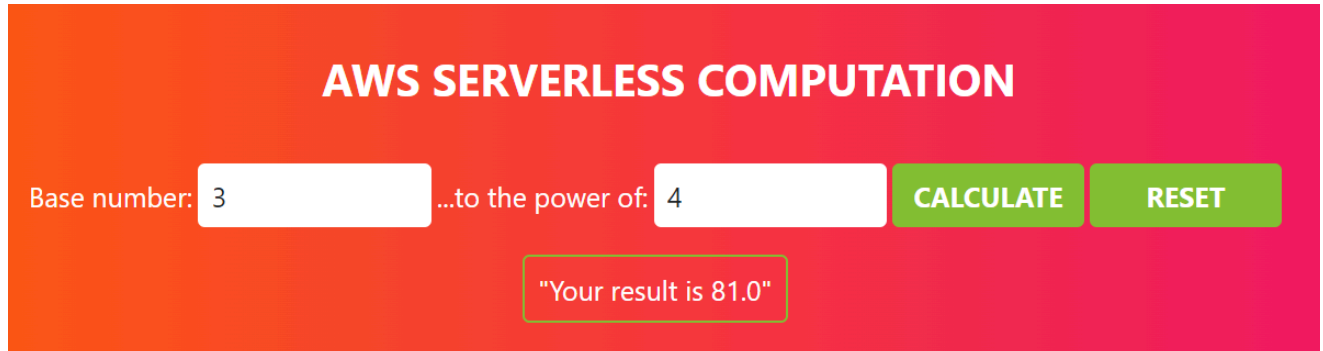
Remove

Cancel

Previous

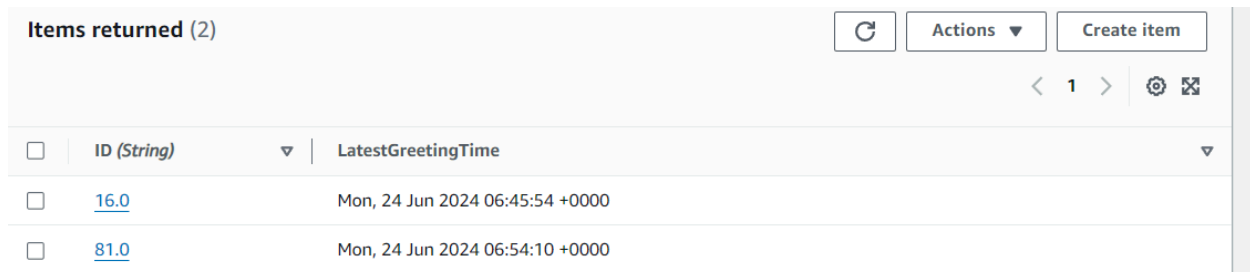
Save and deploy

**Step 15 : once deployed click on url provided, your website will be live and you can test it**



The screenshot shows a web application titled "AWS SERVERLESS COMPUTATION" on a red-to-orange gradient background. It features two input fields: "Base number:" with the value "3" and "...to the power of:" with the value "4". To the right of these fields are two green buttons labeled "CALCULATE" and "RESET". Below the input fields, a yellow box displays the result: "Your result is 81.0".

**Item saved in database also**



The screenshot shows a database table with the header "Items returned (2)". The table has two columns: "ID (String)" and "LatestGreetingTime". There are two rows of data, each with a checkbox in the first column. The first row has the ID "16.0" and the time "Mon, 24 Jun 2024 06:45:54 +0000". The second row has the ID "81.0" and the time "Mon, 24 Jun 2024 06:54:10 +0000".

<input type="checkbox"/>	ID (String)	LatestGreetingTime
<input type="checkbox"/>	<a href="#">16.0</a>	Mon, 24 Jun 2024 06:45:54 +0000
<input type="checkbox"/>	<a href="#">81.0</a>	Mon, 24 Jun 2024 06:54:10 +0000

## Summary

In this I have built a serverless power-calculator application in which I have used AWS lambda service ,dynamo-db,api gateway,aws amplify .When user open application they provide input in browser which hosted on AWS amplify when user click on calculate request(POST) from browser go to AWS lambda using api gateway which will trigger AWS lambda .Lambda will perform computation and after that it will store result in database .so we have make serverless application for which we don't need to run server when we needed then only function get triggered