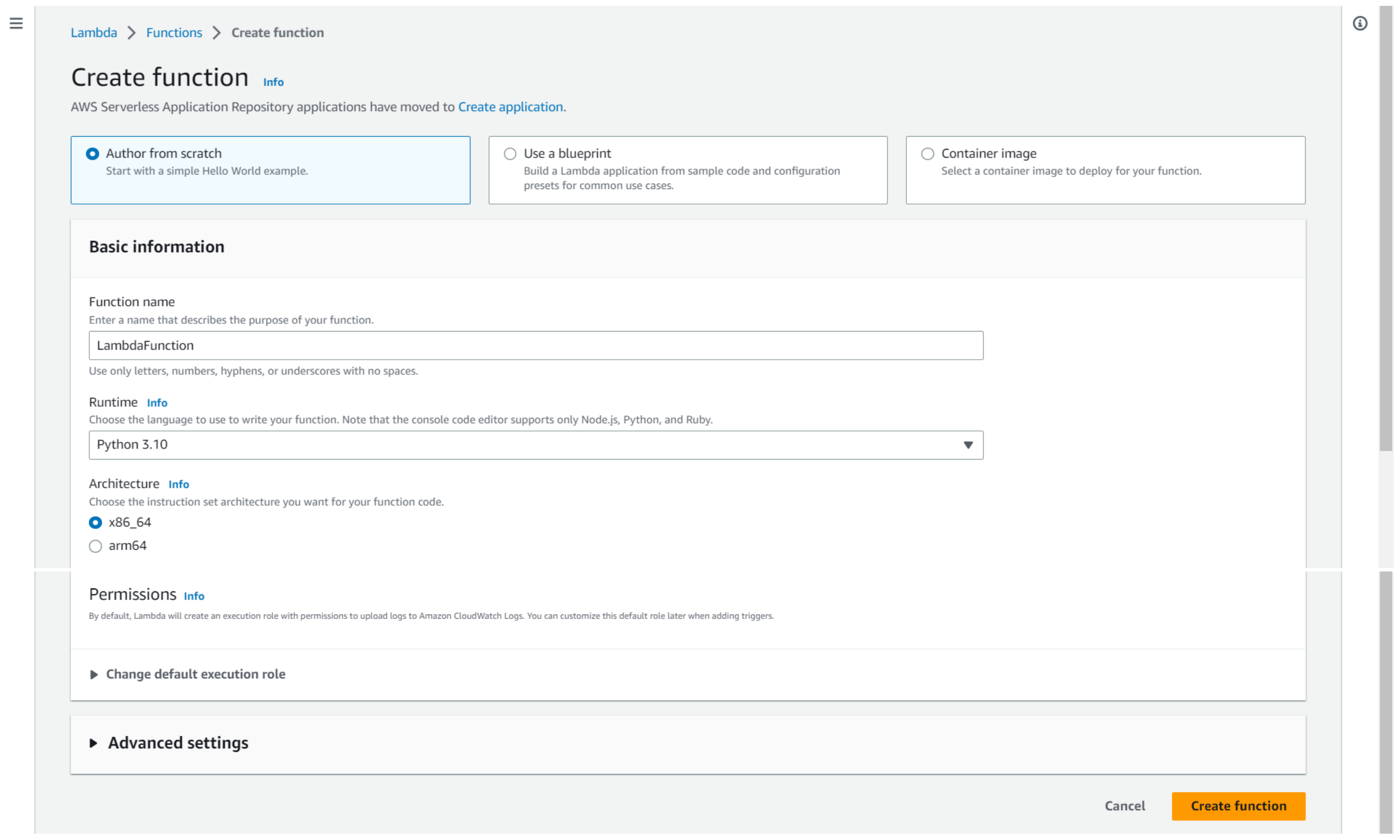
Serverless Application on AWS

Steps1: Create backend using Lambda which will accept API Requests with some parameters.

1. Create Lambda function =>



Lambda > Functions > Create function

Create function Info

AWS Serverless Application Repository applications have moved to Create application.

(●) Author from scratch — Start with a simple Hello World example.
( ) Use a blueprint — Build a Lambda application from sample code and configuration presets for common use cases.
( ) Container image — Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.
LambdaFunction
Use only letters, numbers, hyphens, or underscores with no spaces.

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 3.10

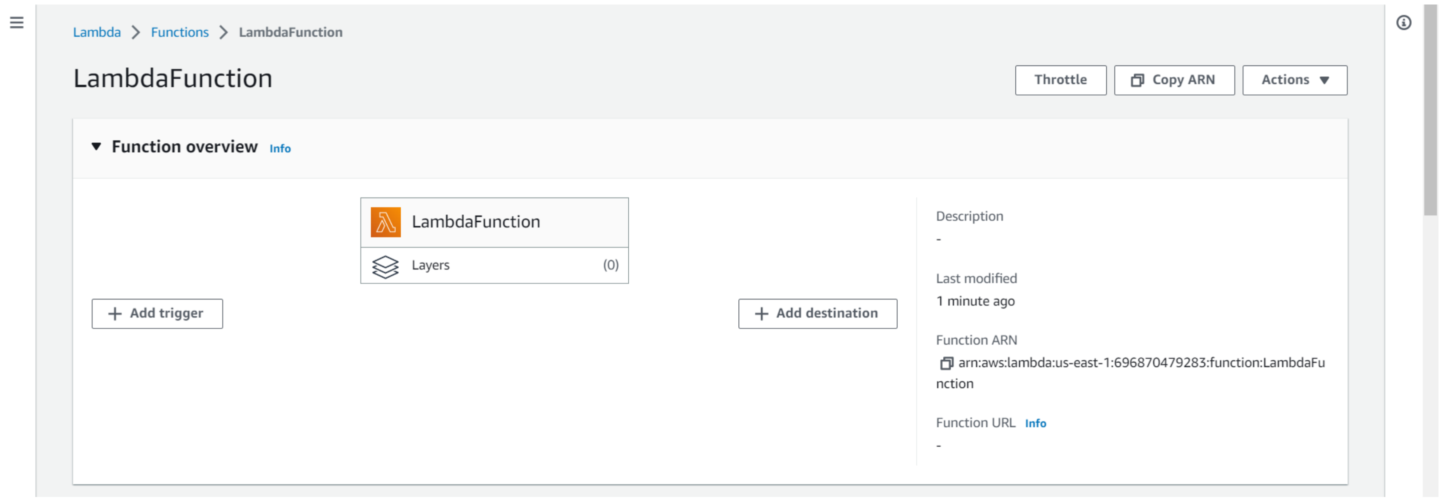
Architecture Info
Choose the instruction set architecture you want for your function code.
(●) x86_64
( ) arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▶ Change default execution role

▶ Advanced settings

Cancel | Create function



Lambda > Functions > LambdaFunction

LambdaFunction — Throttle | Copy ARN | Actions

▼ Function overview Info

LambdaFunction
Layers (0)
+ Add trigger
+ Add destination

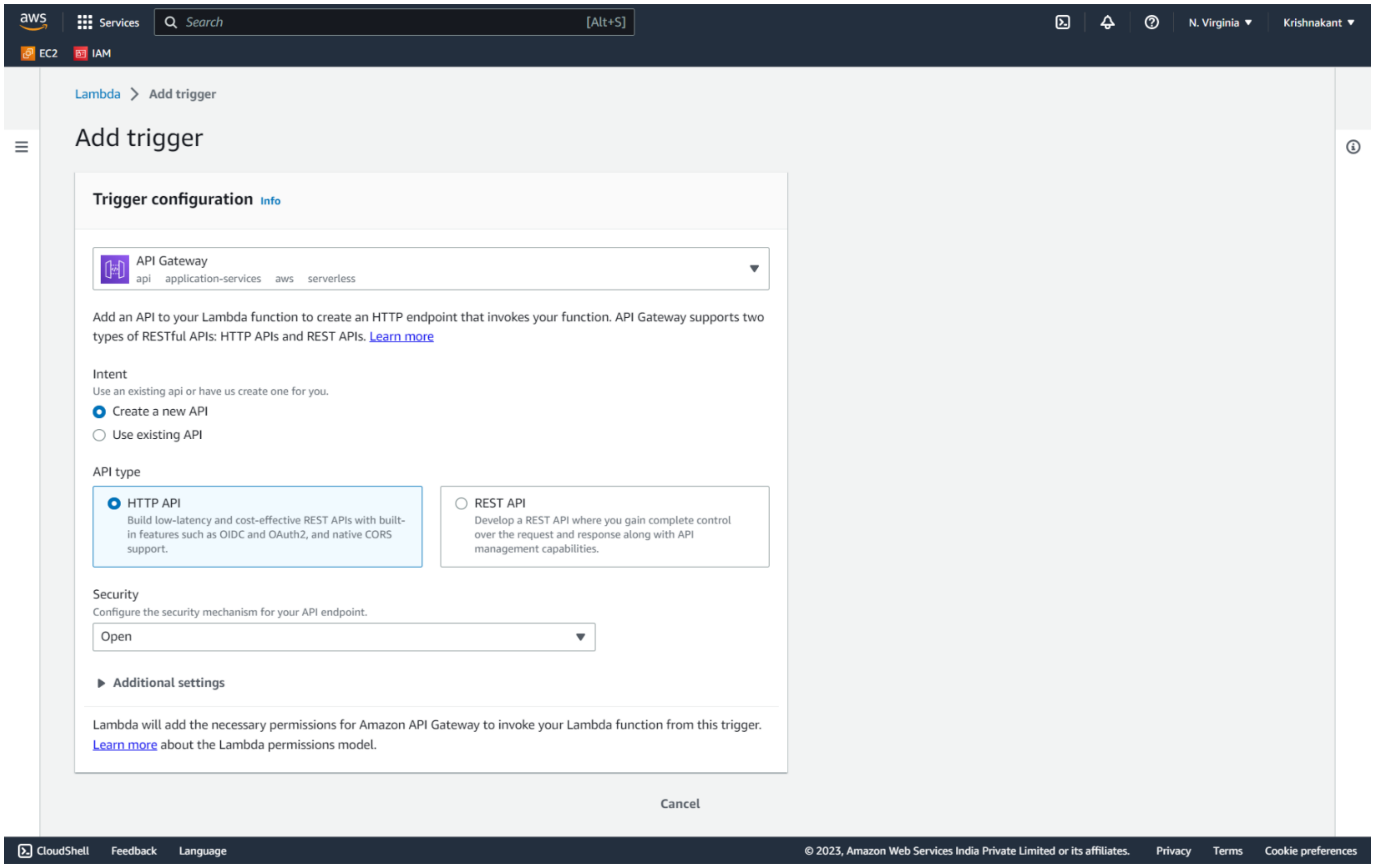
Description
-

Last modified
1 minute ago

Function ARN
arn:aws:lambda:us-east-1:696870479283:function:LambdaFunction

Function URL Info
-

2. Now add API Gateway as a Trigger to handle API Request.



Add trigger

Trigger configuration Info

API Gateway
api application-services aws serverless

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. Learn more

Intent
Use an existing api or have us create one for you.
[✓ Create a new API]
Use existing API

API type
[✓ HTTP API] Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.
REST API Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Security
Configure the security mechanism for your API endpoint.
Open

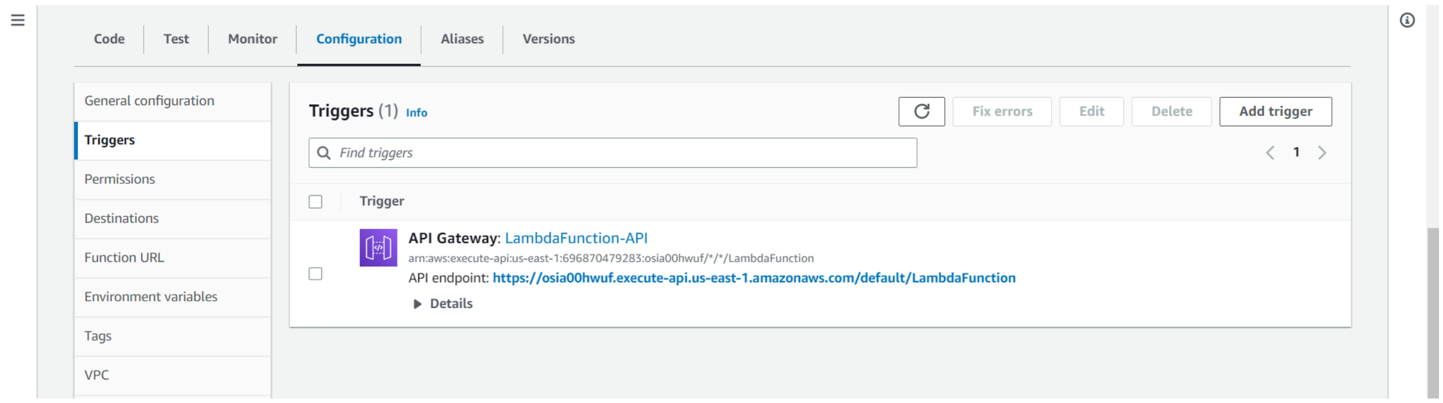
Additional settings

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
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

You'll see the API Endpoint in the configuration tab (Mouse cursor).

That's where we'll send API request with data.

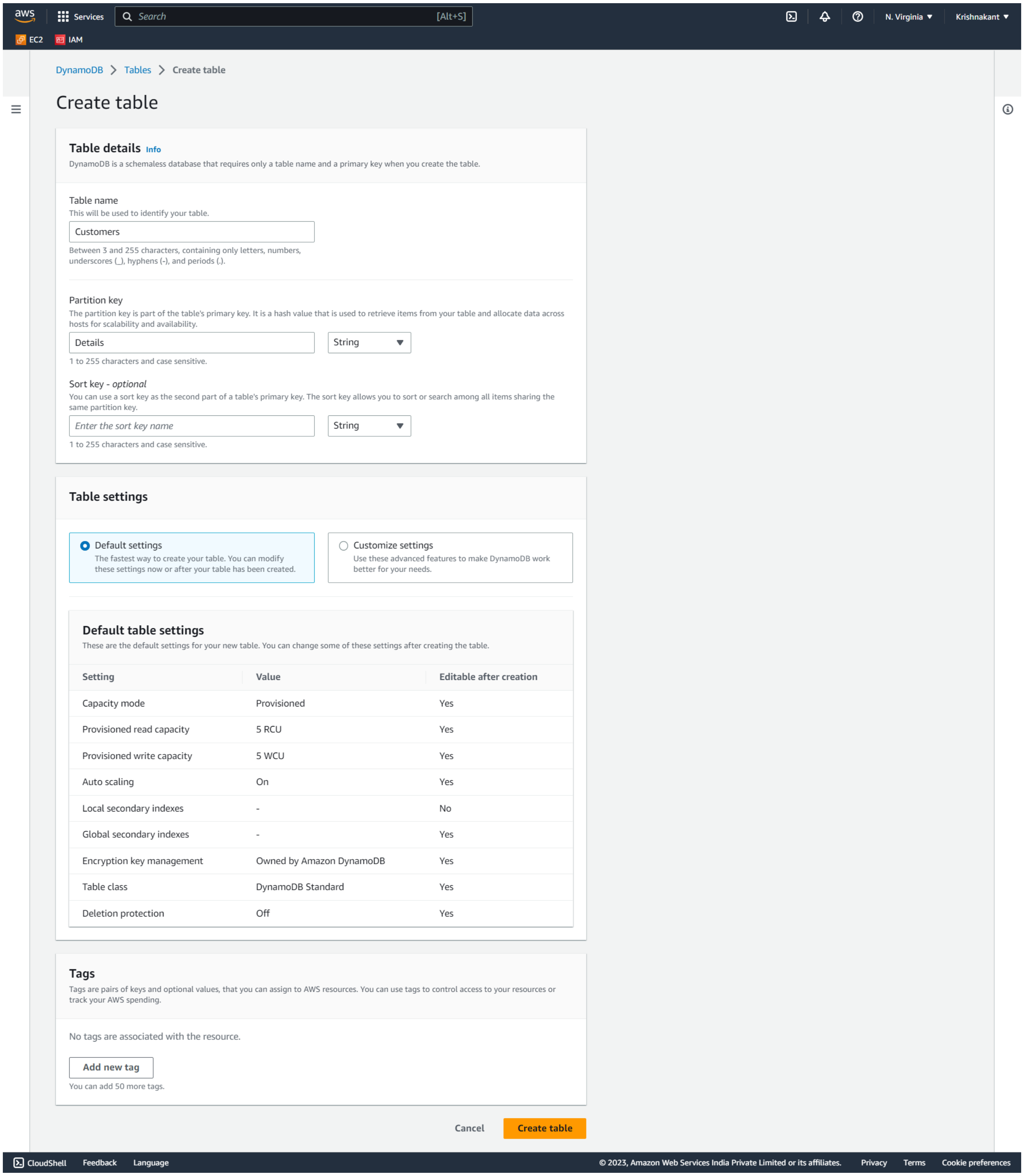


Triggers (1) Info

API Gateway: LambdaFunction-API
arn:aws:execute-api:us-east-1:696870479283:osia00hwuf/*/*/LambdaFunction
API endpoint: https://osia00hwuf.execute-api.us-east-1.amazonaws.com/default/LambdaFunction

Step2: Now before writing code. We'll create Dynamo DB.

1. 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.
Customers
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.
Details — 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.

Table settings

[✓ Default settings] The fastest way to create your table. You can modify these settings now or after your table has been created.

Customize settings — Use these advanced features to make DynamoDB work better for your needs.

Default table settings

These are the default settings for your new table. You can change some of these settings after creating the table.

Setting	Value	Editable after creation
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Table class	DynamoDB Standard	Yes
Deletion protection	Off	Yes

Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

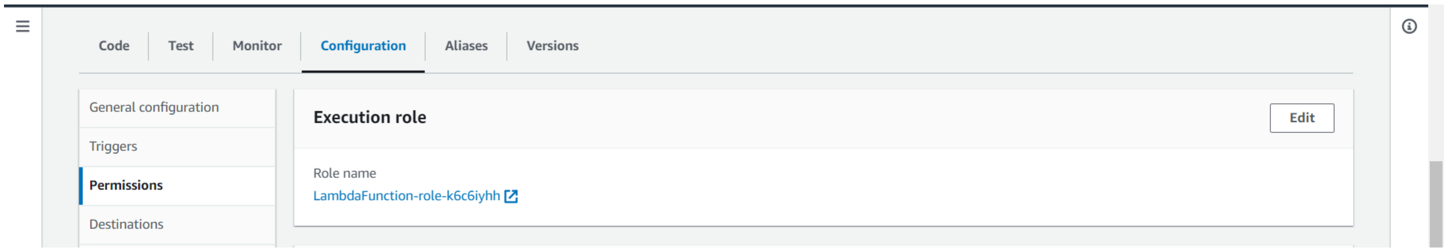
You can add 50 more tags.

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Now database is created. Note the **Partition key** we'll need it after words

Step3: Now give DynamoDb permission to Lambda using Roles.

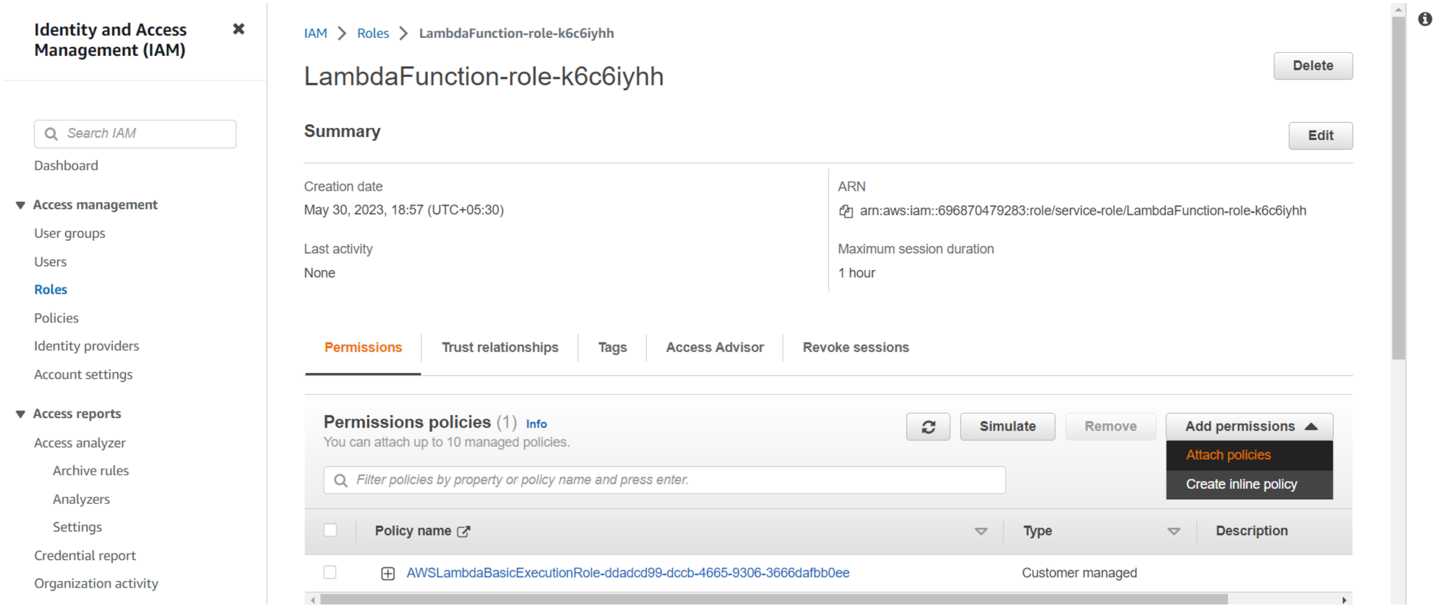
1. Head to Permission tab in Configuration. Select the role that's created by default.



Execution role

Role name
LambdaFunction-role-k6c6iyhh

2. Now in 'add permission' > Attach policy.



LambdaFunction-role-k6c6iyhh

Summary

Creation date
May 30, 2023, 18:57 (UTC+05:30)

ARN
arn:aws:iam::696870479283:role/service-role/LambdaFunction-role-k6c6iyhh

Last activity
None

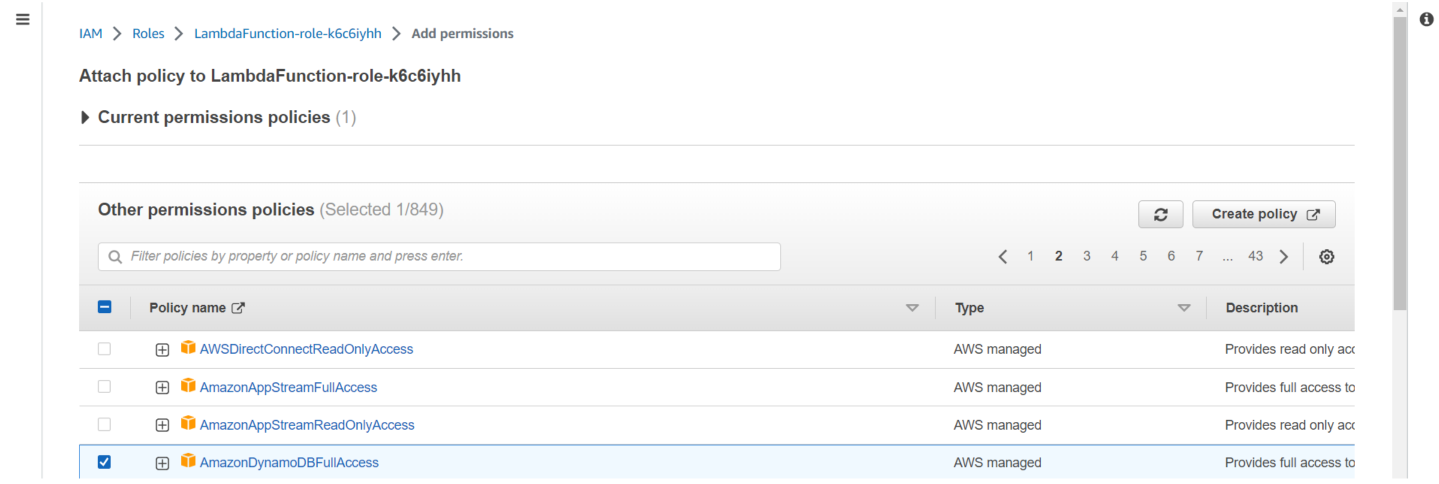
Maximum session duration
1 hour

Permissions policies (1) Info
You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter.

Policy name	Type	Description
AWSLambdaBasicExecutionRole-ddadcd99-dccb-4665-9306-3666dafbb0ee	Customer managed	

3. Give 'DynamoDBFullAccess'


IAM > Roles > LambdaFunction-role-k6c6iyhh > Add permissions
Create policy
1 2 3 4 5 6 7 ... 43

Attach policy to LambdaFunction-role-k6c6iyhh

Current permissions policies (1)

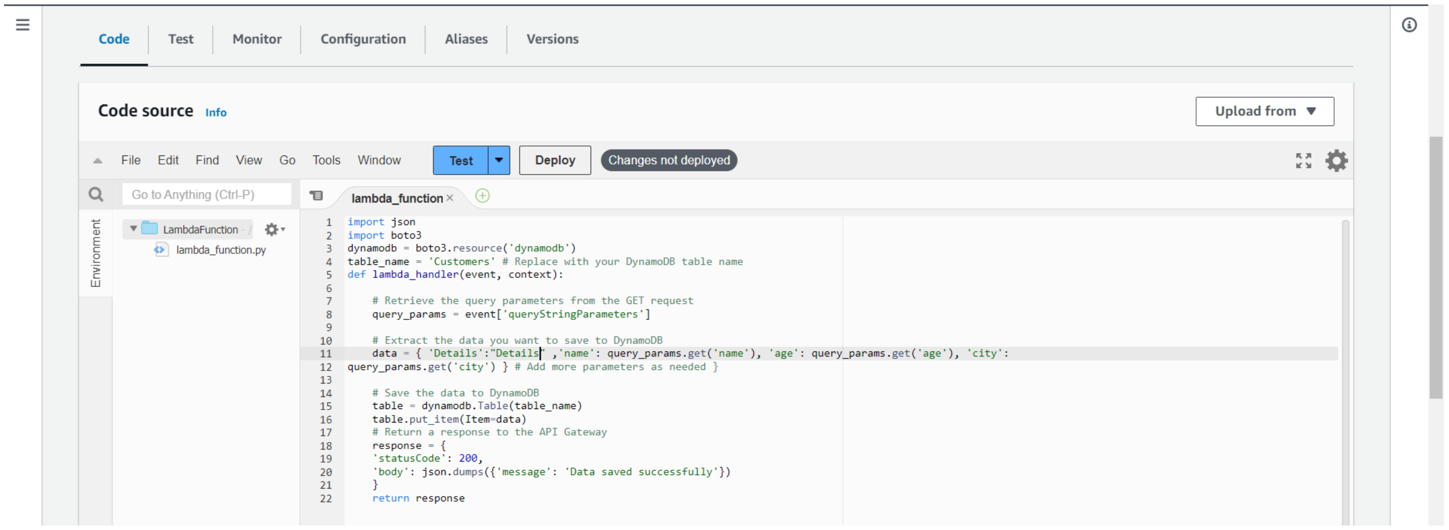
Other permissions policies (Selected 1/849)

Filter policies by property or policy name and press enter.

	Policy name	Type	Description
☐	AWSDirectConnectReadOnlyAccess	AWS managed	Provides read only acc
☐	AmazonAppStreamFullAccess	AWS managed	Provides full access to
☐	AmazonAppStreamReadOnlyAccess	AWS managed	Provides read only acc
✓	AmazonDynamoDBFullAccess	AWS managed	Provides full access to

Step4: Now Write code which takes HTTP parameters and save that into the Dynamo Database.

1. Write Json code in lambda function

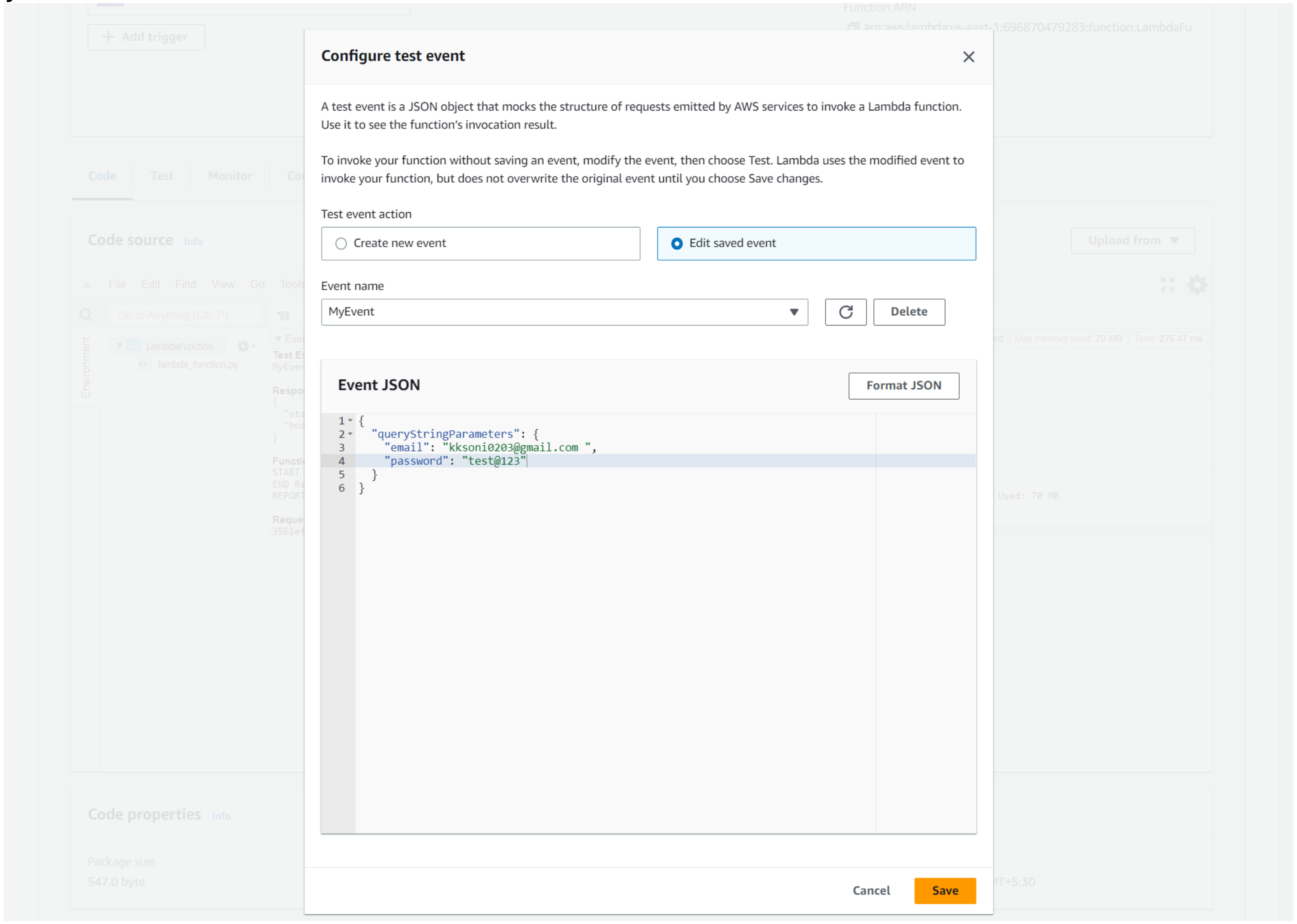


```
import json
import boto3
dynamodb = boto3.resource('dynamodb')
table_name = 'Customers' # Replace with your DynamoDB table name
def lambda_handler(event, context):

    # Retrieve the query parameters from the GET request
    query_params = event['queryStringParameters']

    # Extract the data you want to save to DynamoDB
    data = { 'Details':"Details" ,'name': query_params.get('name'), 'age': query_params.get('age'), 'city':
query_params.get('city') } # Add more parameters as needed }

    # Save the data to DynamoDB
    table = dynamodb.Table(table_name)
    table.put_item(Item=data)
    # Return a response to the API Gateway
    response = {
    'statusCode': 200,
    'body': json.dumps({'message': 'Data saved successfully'})
    }
    return response
```

```
import json

import boto3

dynamodb = boto3.resource('dynamodb')

table_name = 'Customers' # Replace with your DynamoDB table name

def lambda_handler(event, context):

    # Retrieve the query parameters from the GET request

    query_params = event['queryStringParameters']

    # Extract the data you want to save to DynamoDB

    data = { 'userid':"1" ,'name': query_params.get('name'), 'age':
query_params.get('age'), 'city':

query_params.get('city') } # Add more parameters as needed

    # Save the data to DynamoDB

    table = dynamodb.Table(table_name)

    table.put_item(Item=data)

    # Return a response to the API Gateway

    response = {

    'statusCode': 200,

    'body': json.dumps({'message': 'Data saved successfully'})

    }

    return response
```

2. Create Test environment to debug any errors.

```
{
  "queryStringParameters": {
    "email": "kksoni0203@gmail.com ",
    "password": "test@123"
  }
}
```



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, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

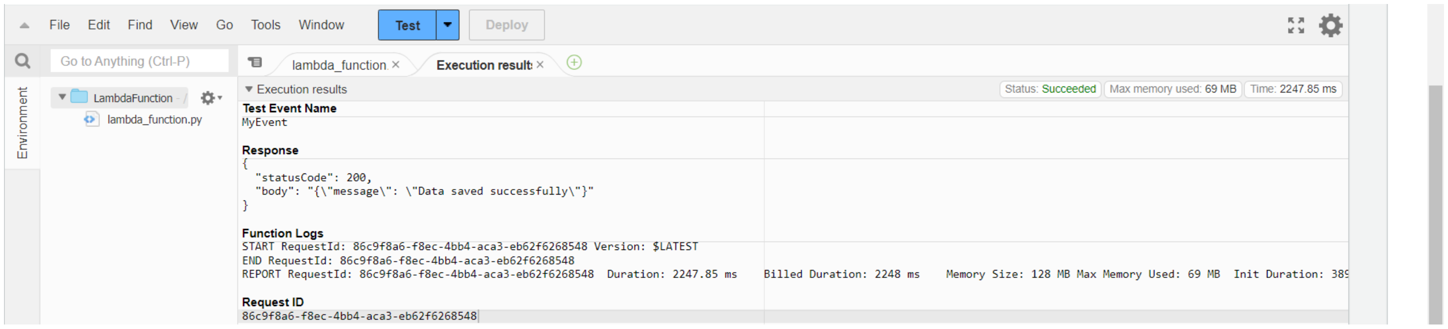
Test event action: ( ) Create new event [✓ Edit saved event]

Event name: MyEvent

Event JSON

```
{
  "queryStringParameters": {
    "email": "kksoni0203@gmail.com ",
    "password": "test@123"
  }
}
```

Now, after you make any changes. Deploy it then click the Test button.
You'll see the code running and Errors and output



Execution results — Status: Succeeded | Max memory used: 69 MB | Time: 2247.85 ms

Test Event Name

```
MyEvent
```

Response

```
{
  "statusCode": 200,
  "body": "{\"message\": \"Data saved successfully\"}"
}
```

Function Logs

```
START RequestId: 86c9f8a6-f8ec-4bb4-aca3-eb62f6268548 Version: $LATEST
END RequestId: 86c9f8a6-f8ec-4bb4-aca3-eb62f6268548
REPORT RequestId: 86c9f8a6-f8ec-4bb4-aca3-eb62f6268548  Duration: 2247.85 ms    Billed Duration: 2248 ms    Memory Size: 128 MB Max Memory Used: 69 MB  Init Duration: 38
```

Request ID

```
86c9f8a6-f8ec-4bb4-aca3-eb62f6268548
```

Step5: Create index.html file in text editor

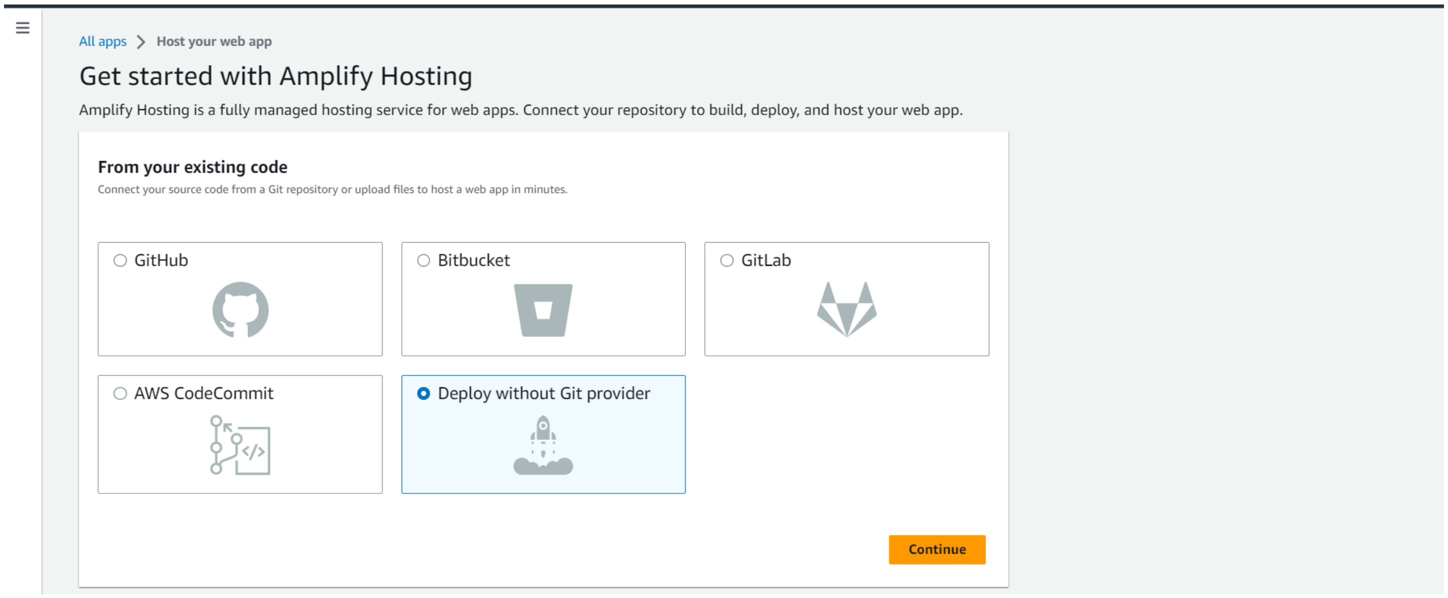
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Register</title>
  </head>
  <body>
    <form
      action="https://osia00hwuf.execute-api.us-east-1.amazonaws.com/default/LambdaFunction">
      Enter your Email: <input name="name"> <br>
      Enter your Password: <input type="number" name="age"> <br>
      <input type="submit">
    </form>
  </body>
</html>
```

Note: Replace the Action value with your API Gateway Endpoint.

Add this index.html in a folder and Zip/Archive it.

Step6: Create a AWS Amplify Web Project.

1. Choose "Deploy without Git Provider"



All apps > Host your web app

Get started with Amplify Hosting

Amplify Hosting is a fully managed hosting service for web apps. Connect your repository to build, deploy, and host your web app.

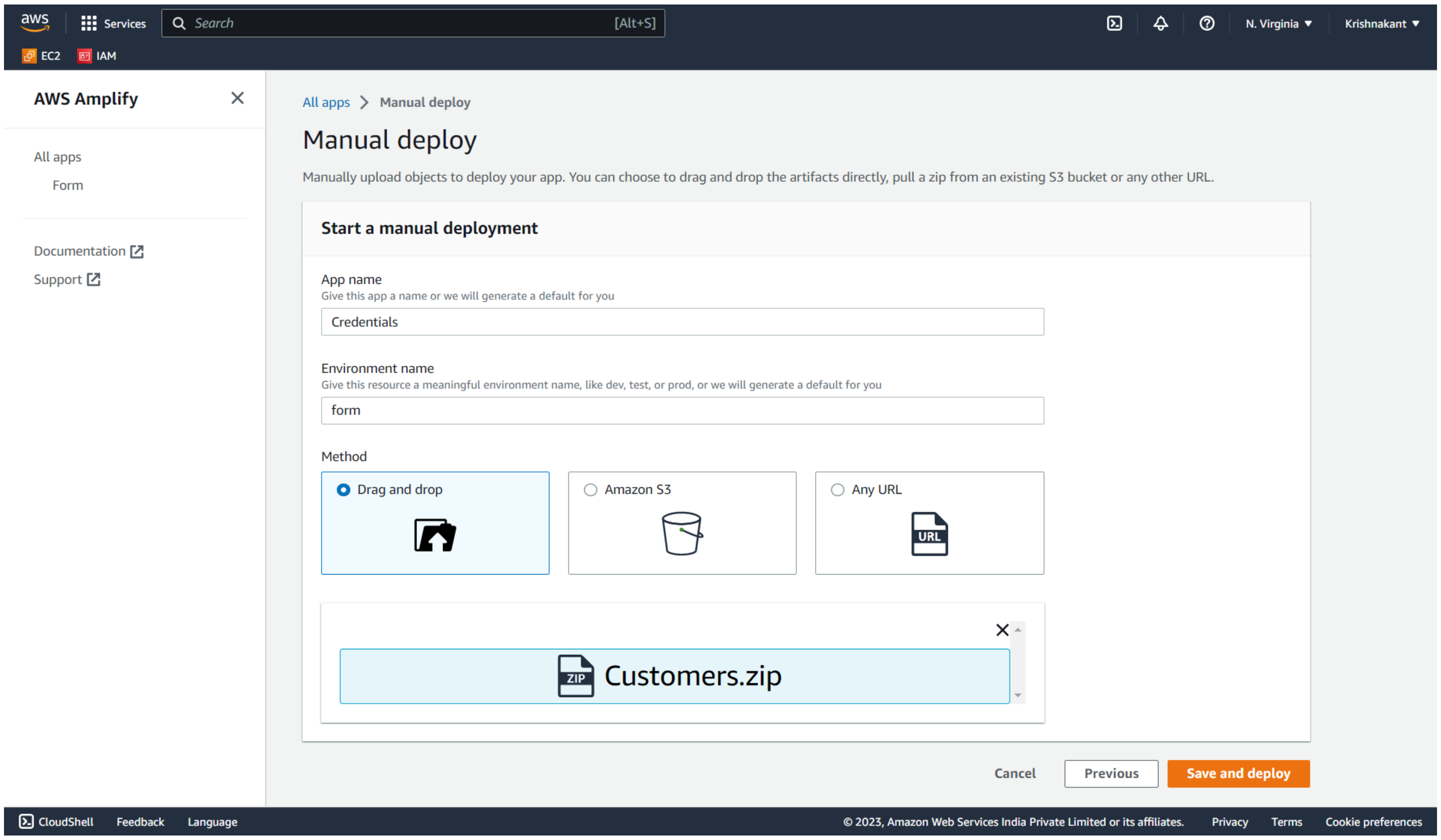
From your existing code

Connect your source code from a Git repository or upload files to host a web app in minutes.

GitHub | Bitbucket | GitLab | AWS CodeCommit | [✓ Deploy without Git provider]

Continue

2. Upload the Zip file on Amplify



All apps > Manual deploy

Manual deploy

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.

Start a manual deployment

App name
Give this app a name or we will generate a default for you
Credentials

Environment name
Give this resource a meaningful environment name, like dev, test, or prod, or we will generate a default for you
form

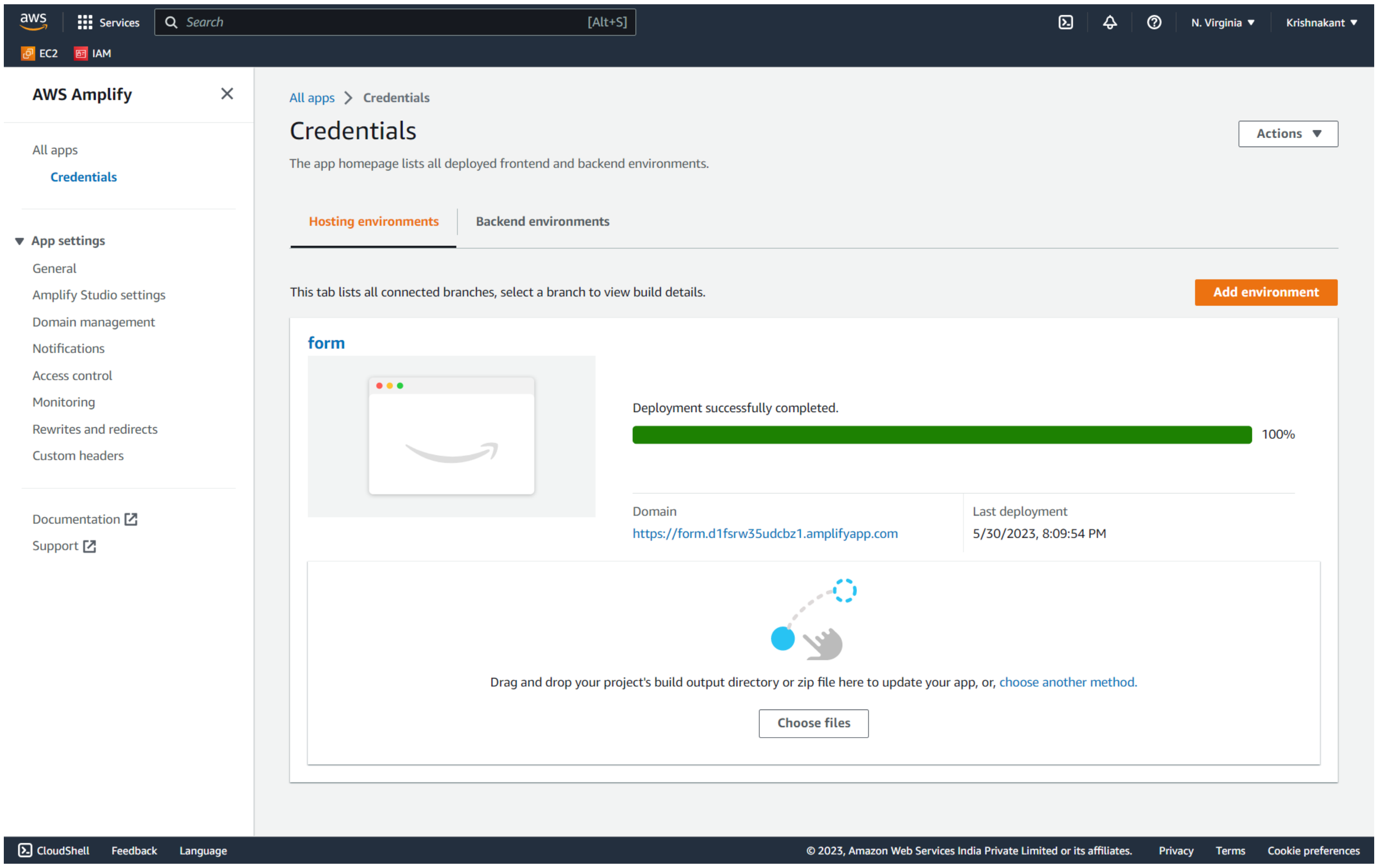
Method
[✓ Drag and drop] | Amazon S3 | Any URL

Customers.zip

Cancel | Previous | Save and deploy

© 2023, Amazon Web Services India Private Limited or its affiliates.

3. Click on Domain Link


aws | Services | Search [Alt+S] | N. Virginia | Krishnakant
EC2 | IAM

AWS Amplify
- All apps
  - Credentials
- App settings
  - General
  - Amplify Studio settings
  - Domain management
  - Notifications
  - Access control
  - Monitoring
  - Rewrites and redirects
  - Custom headers
- Documentation
- Support

All apps > Credentials

Credentials

The app homepage lists all deployed frontend and backend environments.

Actions

Hosting environments | Backend environments

This tab lists all connected branches, select a branch to view build details.

Add environment

form

Deployment successfully completed.

100%

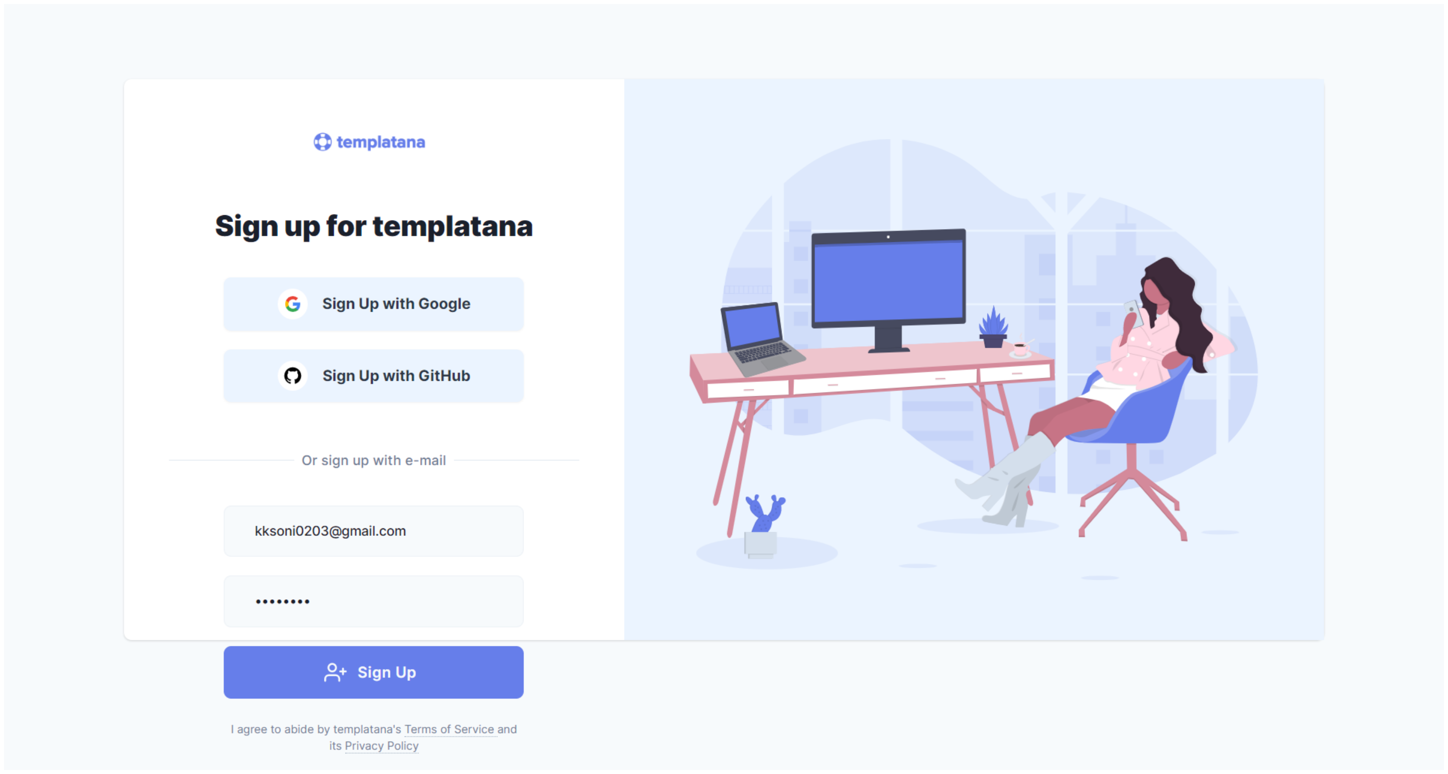
Domain	Last deployment
https://form.d1fsrw35udcbz1.amplifyapp.com	5/30/2023, 8:09:54 PM

Drag and drop your project's build output directory or zip file here to update your app, or, choose another method.

Choose files

CloudShell | Feedback | Language | © 2023, Amazon Web Services India Private Limited or its affiliates. | Privacy | Terms | Cookie preferences

4. Login the form


templatana

Sign up for templatana

Sign Up with Google

Sign Up with GitHub

Or sign up with e-mail

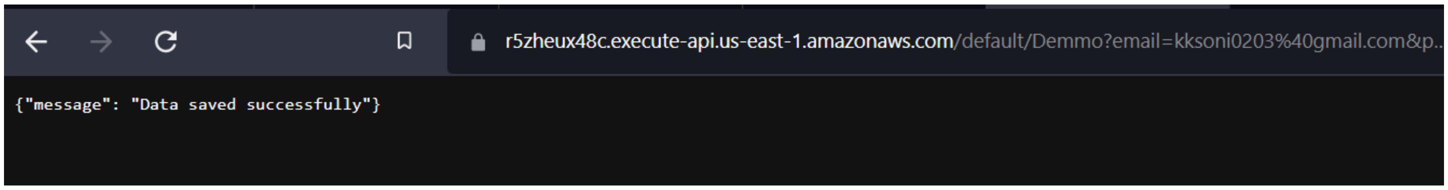
kksoni0203@gmail.com

••••••••

Sign Up

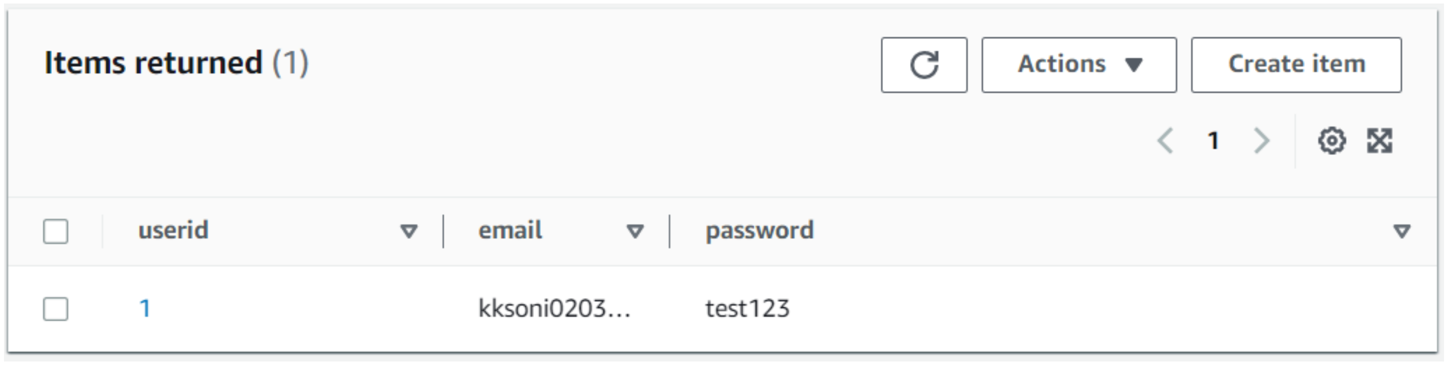
I agree to abide by templatana's Terms of Service and its Privacy Policy

5. The data store in the RDS database



r5zheux48c.execute-api.us-east-1.amazonaws.com/default/Demmo?email=kksoni0203%40gmail.com&p...

```
{"message": "Data saved successfully"}
```



Items returned (1)

userid	email	password
1	kksoni0203...	test123