

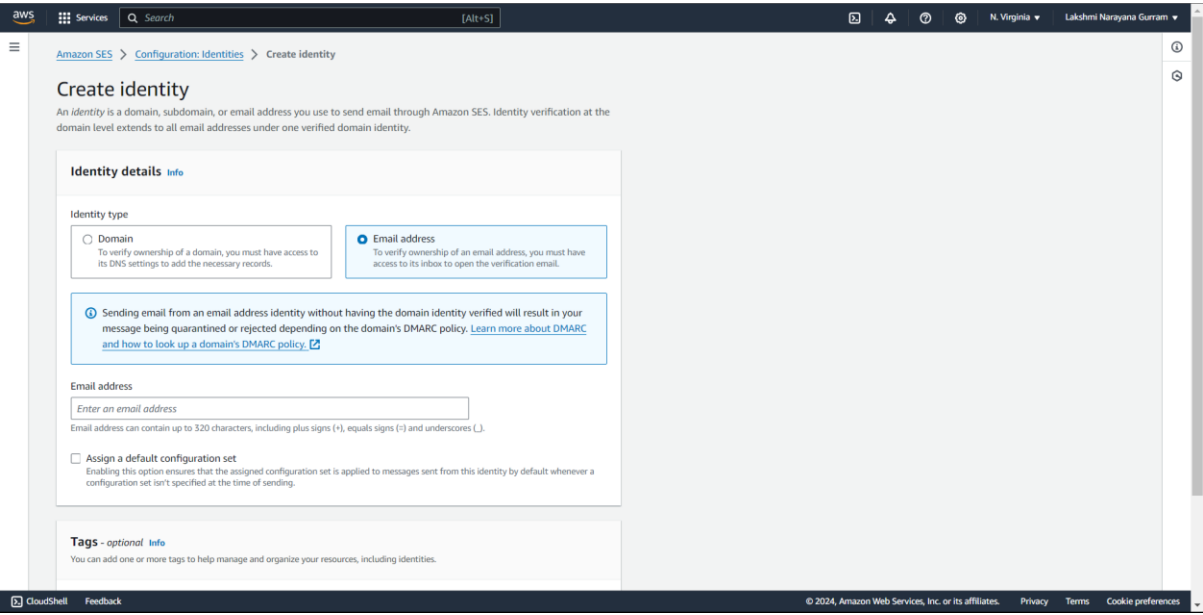
Remainder Application using AWS Serverless

2100031943

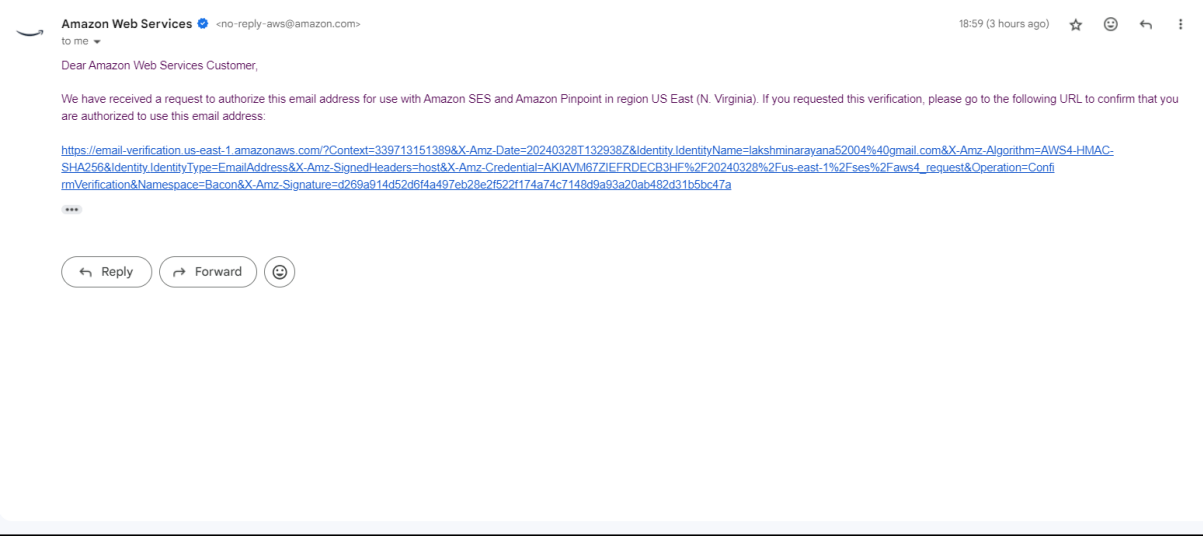
G Lakshmi Narayana

Open SES service and create a identity

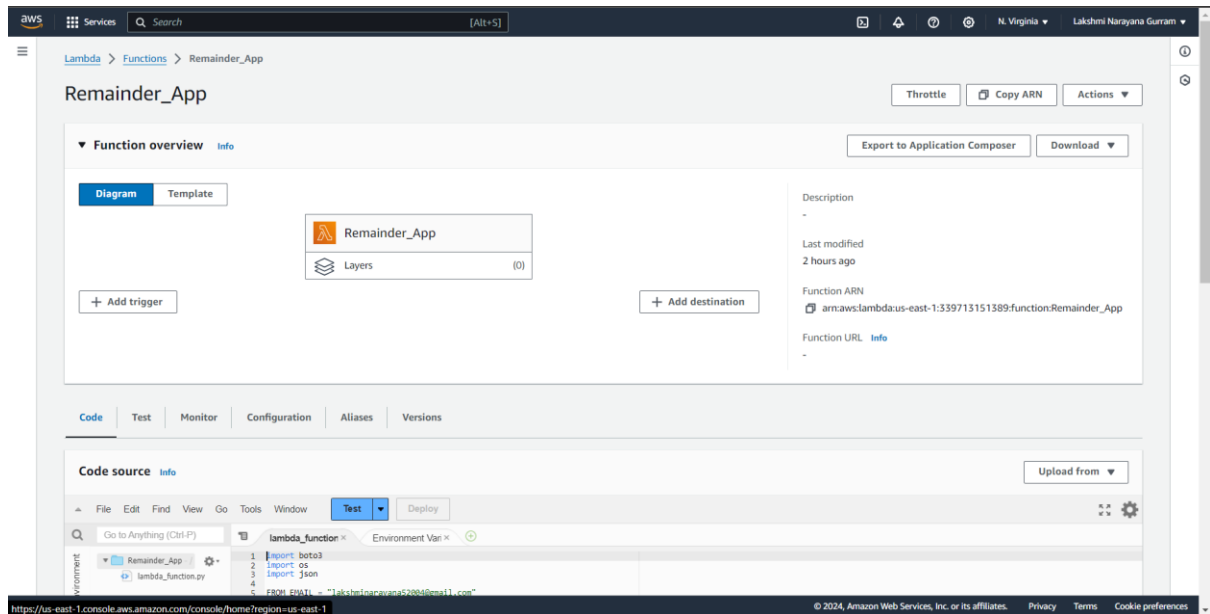
And select email



and verify the mail



Create a Lambda function with run time python



Code:

```
import boto3
```

```
import os
```

```
import json
```

```
FROM_EMAIL = "set your mail"
```

```
ses = boto3.client('ses')
```

```
def lambda_handler(event, context):
```

```
    print("Received Mail:", json.dumps(event))
```

```
    ses.send_email(
```

```
        Source=FROM_EMAIL,
```

```
        Destination={'ToAddresses': [event['Input']['email']]},
```

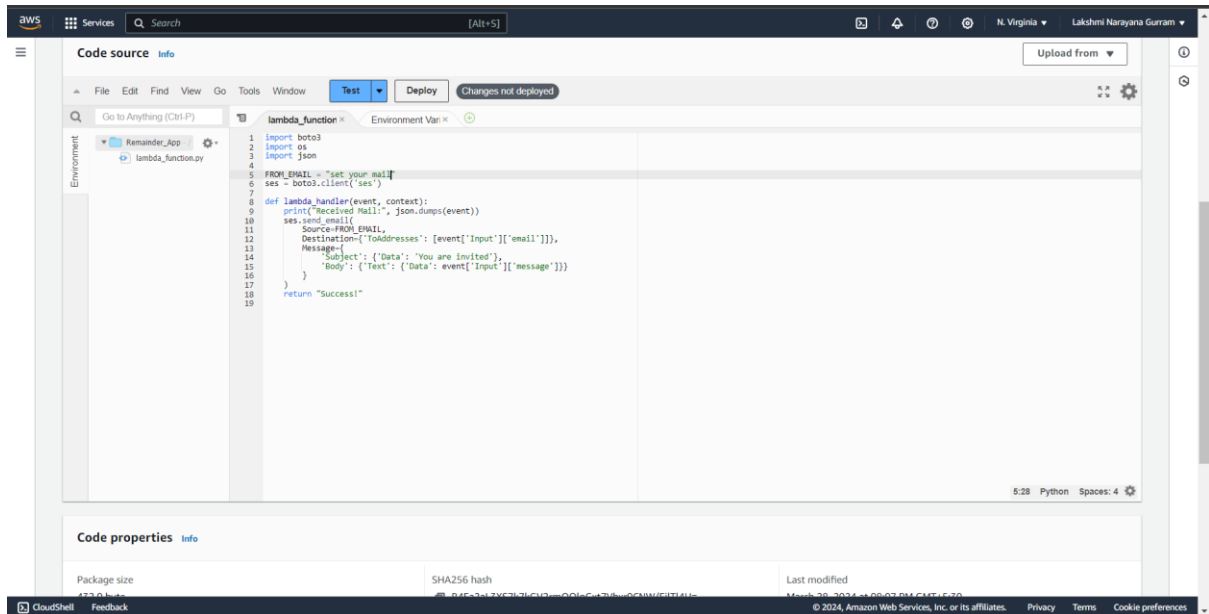
```
        Message={
```

```
            'Subject': {'Data': 'You are invited'},
```

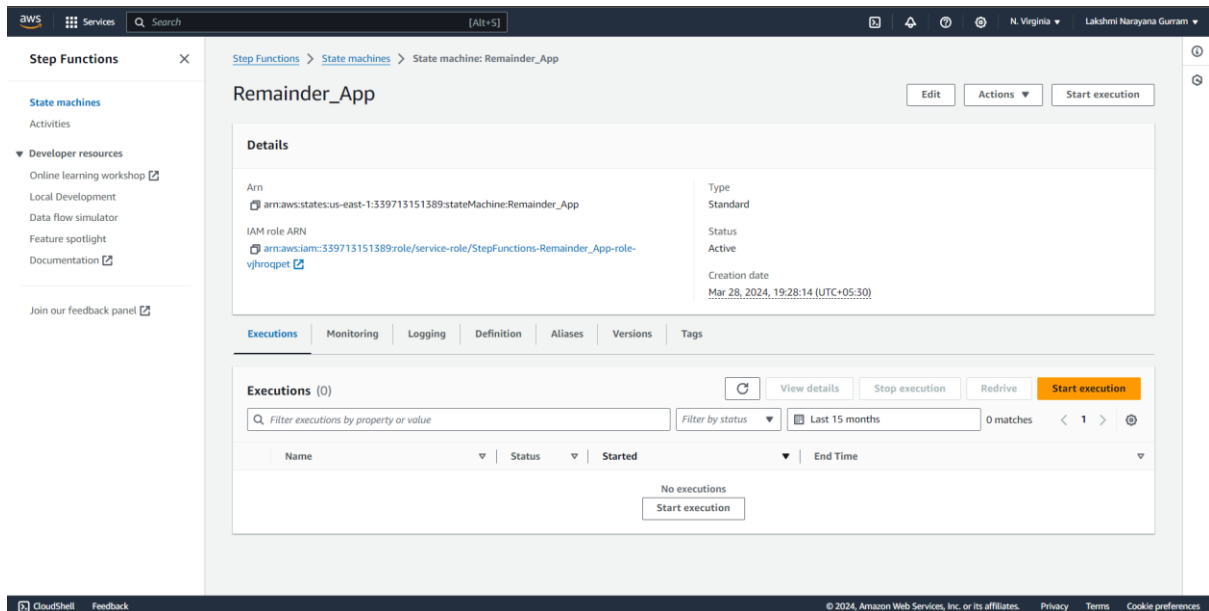
```
            'Body': {'Text': {'Data': event['Input']['message']}})
```

```
}
)

return "Success!"
```



Create a step function



Code:

```
{
  "Comment": "A description of my state machine",
  "StartAt": "Timer",
  "States": {
    "Timer": {
      "Type": "Wait",
```

```

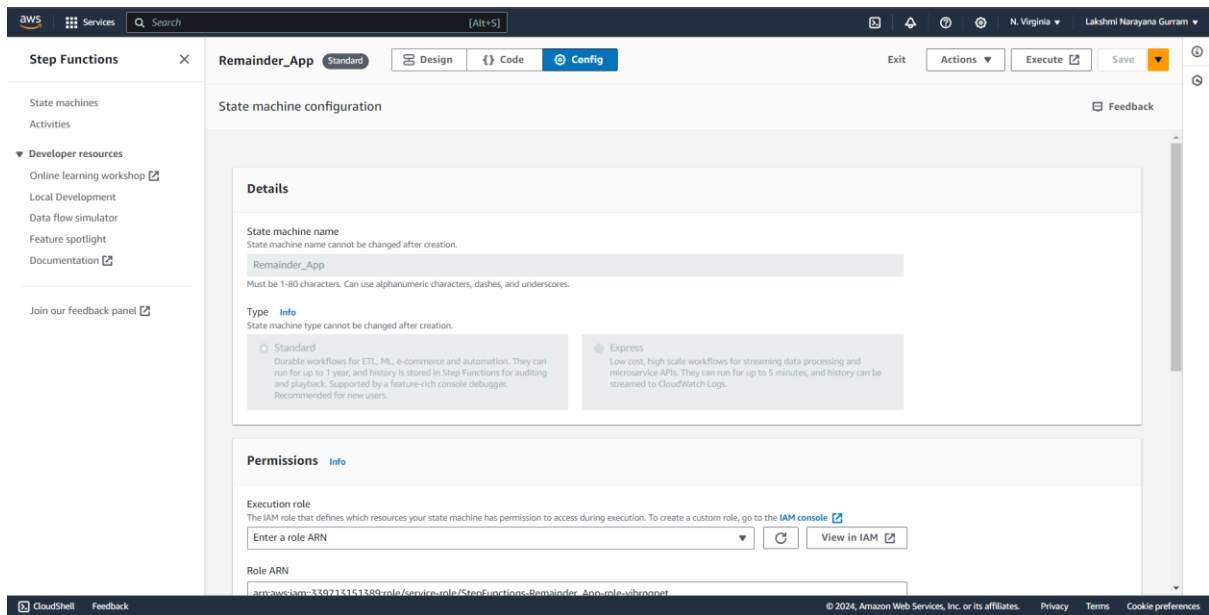
    "SecondsPath": "$.waitSeconds",
    "Next": "Email"
  },
  "Email": {
    "Type": "Task",
    "Resource": "lambda function arn",
    "Parameters": {
      "FunctionName": "EMAIL_LAMBDA",
      "Payload": {
        "Input.$": "$"
      }
    }
  },
  "Next": "NextState"
},
"NextState": {
  "Type": "Pass",
  "End": true
}
}
}

```

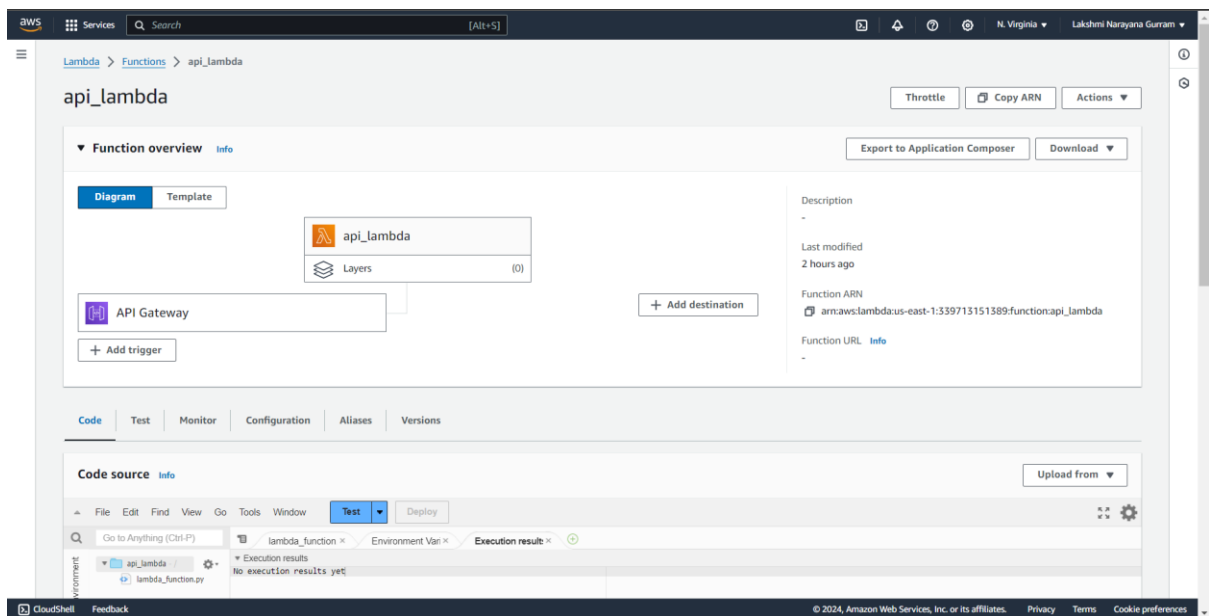
The screenshot displays the AWS Step Functions console interface. On the left, a sidebar shows navigation options like 'State machines', 'Activities', and 'Developer resources'. The main area is titled 'Remainder_App' and includes tabs for 'Design', 'Code', and 'Config'. The 'Code' tab is active, showing a JSON configuration for a state machine. The configuration defines a 'Timer' state that transitions to an 'Email' task, which then transitions to a 'NextState' pass state before reaching the 'End' state.

The visual flow diagram on the right illustrates the state machine's execution path: Start → Wait state Timer → Lambda: Invoke Email → Pass state NextState → End.

At the bottom of the console, there is a footer with the text: © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.



create another lambda function
run time python



Code: import boto3

import json

SM_ARN = "stepfunction_Arn"

sm = boto3.client('stepfunctions')

def lambda_handler(event, context):

```
print("Received event:", json.dumps(event))
```

```
# Check if 'body' attribute is present in the event
```

```
if 'body' not in event:
```

```
    return {
```

```
        "statusCode": 400,
```

```
        "headers": {"Access-Control-Allow-Origin": "*"},
```

```
        "body": json.dumps({"Status": "Failure", "Reason": "'body' attribute missing in the event"})
```

```
    }
```

```
try:
```

```
    data = json.loads(event['body'])
```

```
    data['waitSeconds'] = int(data['waitSeconds'])
```

```
    checks = [
```

```
        'waitSeconds' in data,
```

```
        isinstance(data['waitSeconds'], int),
```

```
        'message' in data
```

```
    ]
```

```
if False in checks:
```

```
    response = {
```

```
        "statusCode": 400,
```

```
        "headers": {"Access-Control-Allow-Origin": "*"},
```

```
        "body": json.dumps({"Status": "Failure", "Reason": "Input failed validation"})
```

```
    }
```

```
else:
```

```
    sm.start_execution(stateMachineArn=SM_ARN, input=json.dumps(data))
```

```
    response = {
```

```
        "statusCode": 200,
```

```
        "headers": {"Access-Control-Allow-Origin": "*"},
```

```
"body": json.dumps({"Status": "Success"})
```

```
}
```

except Exception as e:

```
response = {
```

```
    "statusCode": 500,
```

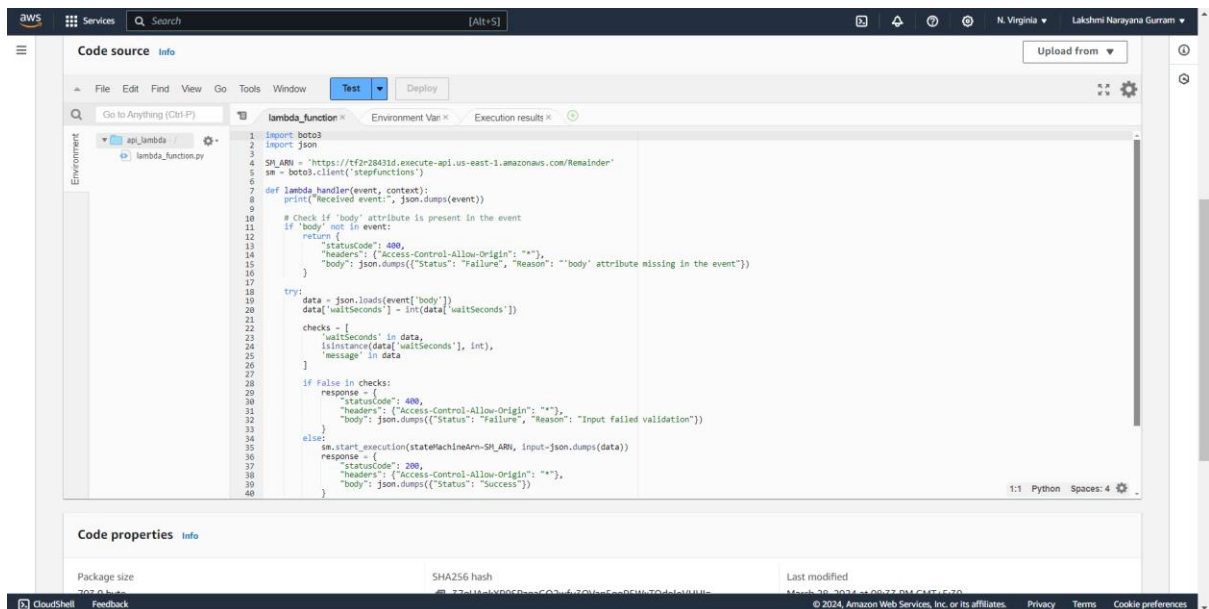
```
    "headers": {"Access-Control-Allow-Origin": "*"},
```

```
    "body": json.dumps({"Status": "Failure", "Reason": str(e)})
```

```
}
```

return response

paste and deploy it



Create an S3 bucket

Amazon S3 > Buckets > Create bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region
US East (N. Virginia) us-east-1

Bucket type [Info](#)

☒ **General purpose**
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ **Directory - New**
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)
remainderApp
Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

Format: s3://bucket/prefix

Object Ownership [Info](#)

Give it public access

Applications that work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☐ **Block all public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ **Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.

☐ **Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ **Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☒ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions

Add bucket policy

Code: {

"Version": "2012-10-17",

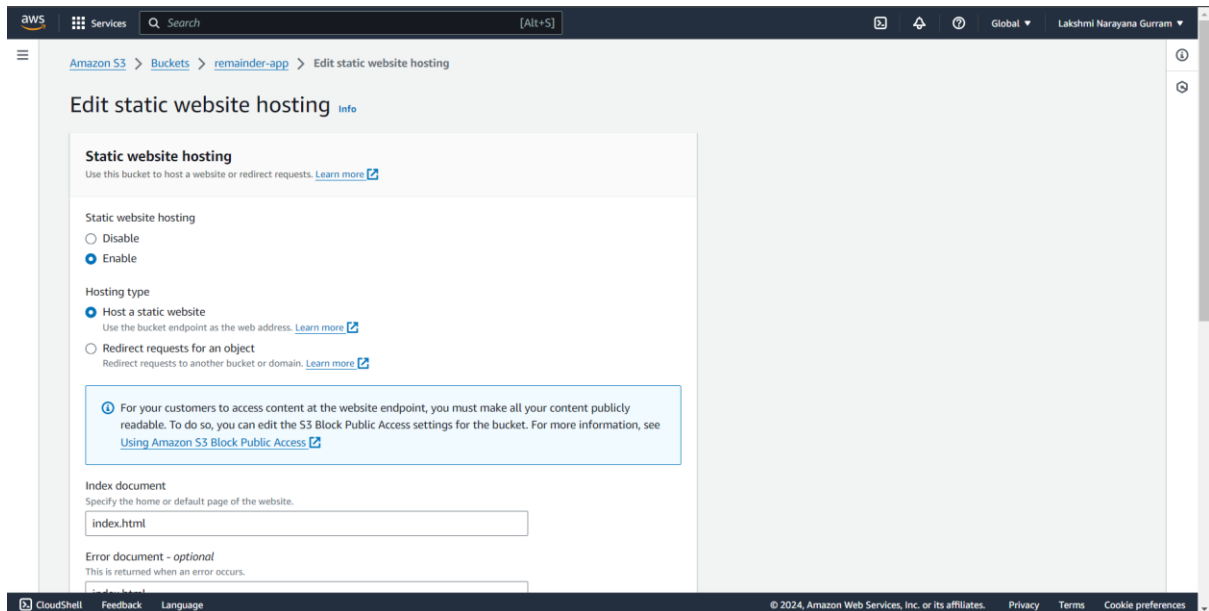
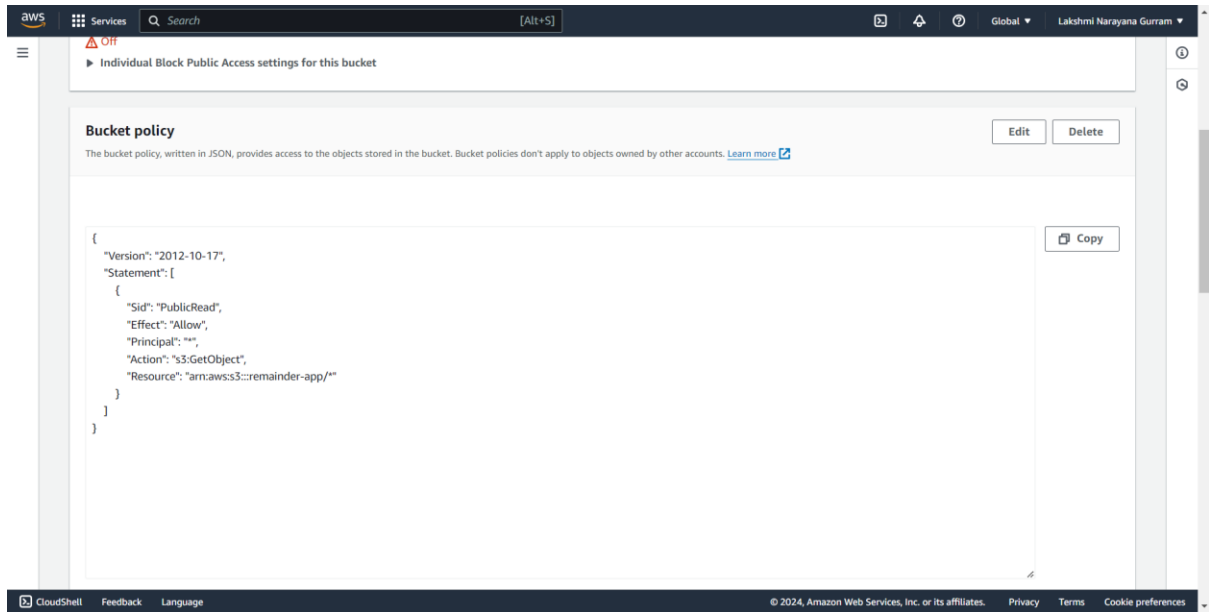
"Statement": [

{

"Sid": "PublicRead",

"Effect": "Allow",


```
"Principal": "*",  
  
"Action": "s3:GetObject",  
  
"Resource": "arn:aws:s3:::bucketname/*"  
}  
]  
}
```



Create an Api gateway

The screenshot shows the 'Create REST API' page in the AWS Management Console. The breadcrumb trail is 'API Gateway > APIs > Create API > Create REST API'. The page title is 'Create REST API'. Under the 'API details' section, there are four radio button options: 'New API' (selected), 'Clone existing API', 'Import API', and 'Example API'. Below these, the 'API name' field contains 'Reminder App'. The 'Description - optional' field is empty. The 'API endpoint type' dropdown is set to 'Regional'. At the bottom right of the form are 'Cancel' and 'Create API' buttons.

API details

☒ New API
Create a new REST API.

☐ Clone existing API
Create a copy of an API in this AWS account.

☐ Import API
Import an API from an OpenAPI definition.

☐ Example API
Learn about API Gateway with an example API.

API name
Reminder App

Description - optional

API endpoint type
Regional

Cancel Create API

Create a resource in rest api

The screenshot shows the 'Create resource' page in the AWS Management Console. The breadcrumb trail is 'API Gateway > APIs > Resources - Reminder_App [tf2z28431d] > Create resource'. The page title is 'Create resource'. Under the 'Resource details' section, the 'Proxy resource' radio button is selected. Below this, the 'Resource path' dropdown is set to '/' and the 'Resource name' field contains 'remainder'. The 'CORS (Cross Origin Resource Sharing)' checkbox is checked. At the bottom right of the form are 'Cancel' and 'Create resource' buttons.

Create resource

Resource details

☒ Proxy resource
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path
/

Resource name
remainder

☒ CORS (Cross Origin Resource Sharing)
Create an OPTIONS method that allows all origins, all methods, and several common headers.

Cancel Create resource

Create a method post and select lambda
and below select the lambda function

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

Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

☒ Default timeout
The default timeout is 29 seconds.

► Method request settings

Create a index.html page

```
<> index.html > ...
  Click here to ask Blackbox to help you code faster
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Send Reminder</title>
7  </head>
8  <body>
9    <h1>Send Reminder</h1>
10   <label for="email">Email:</label>
11   <input type="email" id="email" name="email"><br><br>
12   <label for="message">Message:</label>
13   <input type="text" id="message" name="message"><br><br>
14   <label for="waitSeconds">Wait Seconds:</label>
15   <input type="number" id="waitSeconds" name="waitSeconds"><br><br>
16   <button onclick="sendReminder()">Send Reminder</button>
17
18
19   <script src="scripts.js"></script>
20 </body>
21 </html>
22
```

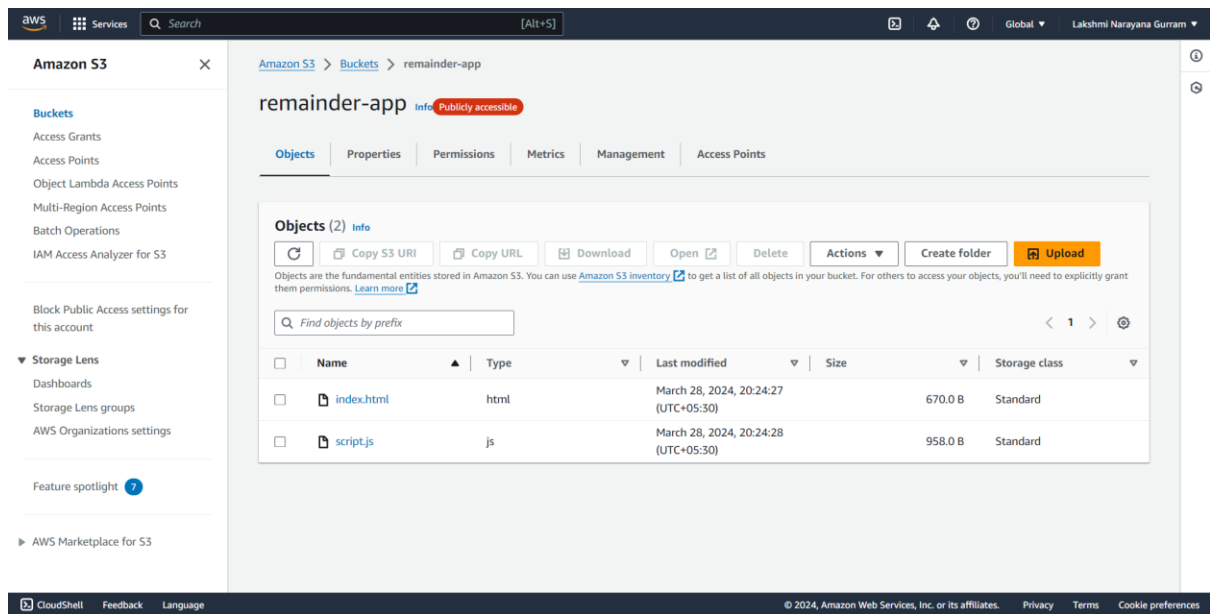
And scripts.js code

```
function sendReminder() {
  const email = document.getElementById('email').value;
  const message = document.getElementById('message').value;
  const waitSeconds = document.getElementById('waitSeconds').value;

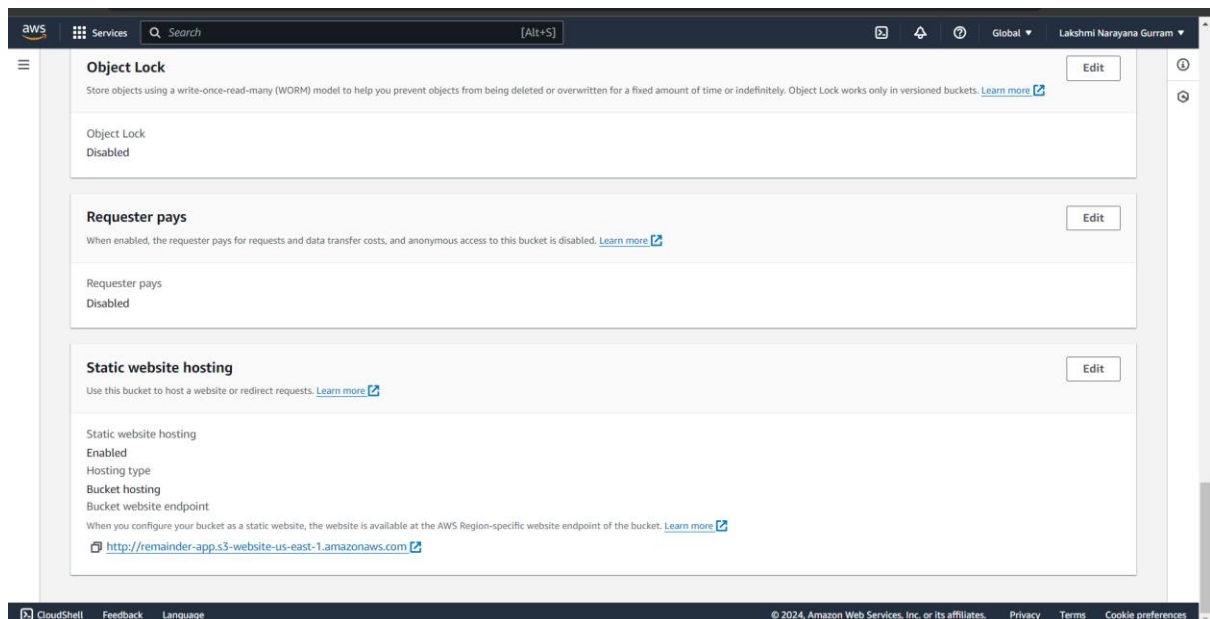
  const data = {
    email: email,
    message: message,
    waitSeconds: waitSeconds
  };

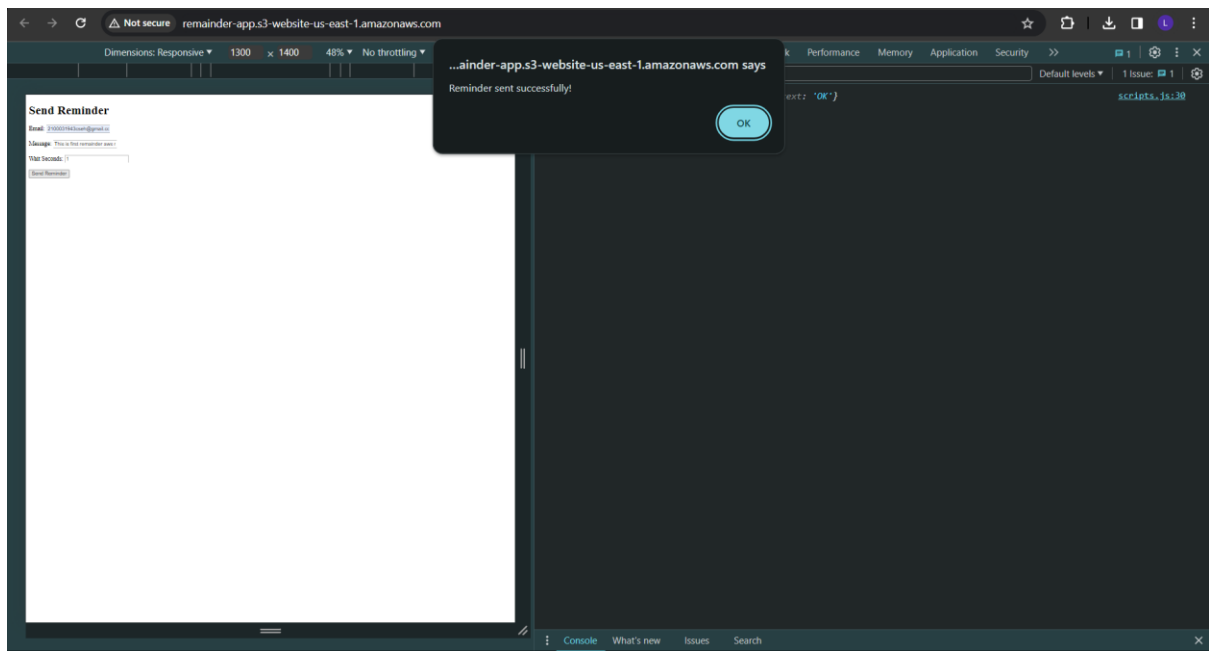
  fetch('/Reminder', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(data)
  })
  .then(response => {
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    return response.json();
  })
  .then(data => {
    console.log('Success:', data);
    // Handle success response here
  })
  .catch(error => {
    console.error('Error:', error);
    // Handle error here
  });
}
```

Once create upload them in the previous created s3 bucket



After successfully uploading go to properties ther u will find and static web hosting link click on it





This is the result

lakshminarayana52004@gmail.com

22:35 (0 minutes ago)

to me

from: lakshminarayana52004@gmail.com

reply-to: 2100031943cseh@gmail.com

to: 2100031943cseh@gmail.com

date: 28 Mar 2024, 22:35

subject: this is a message

mailed-by: gmail.com

Signed by: gmail.com

security: Standard encryption (TLS) [Learn more](#)

Important according to Google magic.

This is first remainder aws mail sent