

AWS Cloud Resume Challenge:

Note: The HTML, CSS, and JavaScript code can be found in the GitHub repository linked below. Please replace the placeholder images with your own images.

For instructions on hosting your portfolio website using S3 and CloudFront, and configuring it with your custom domain, check out my video linked below.

Github Repository: <https://github.com/suzal777/PortfolioWebsite.git>

Video Link: https://youtu.be/ddrTULjH_bg

The below tutorials are for adding Views Count feature and Contact Me feature in the Portfolio Website.

All the best!

Views Count feature

Creating DynamoDB Table:

Create a DynamoDB table named "PortfolioViewCounter", with "id" defined as the partition key.

The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. The table name is 'PortfolioViewCounter'. The partition key is 'id' of type String. The sort key is optional and left blank. Under table settings, 'Default settings' is selected. The page includes standard AWS navigation and status bars.

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.
 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.
 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.
 String
1 to 255 characters and case sensitive.

Table settings

Default settings
The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.

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

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Click on the "PortfolioViewCounter" table.

The screenshot shows the Amazon DynamoDB console interface. On the left, there's a navigation sidebar with 'DynamoDB' selected under 'Tables'. The main area displays a table titled 'Tables (2)'. The table has columns for Name, Status, Partition key, Sort key, Indexes, Replication Regions, Deletion protection, Favorite, and Recycle bin. Two rows are visible: 'PortfolioViewCounter' (Status: Active, Partition key: id (\$), Sort key: -) and 'ViewCounter' (Status: Active, Partition key: id (\$), Sort key: -). A blue banner at the top of the main area says 'Creating the PortfolioViewCounter table. It will be available for use shortly.' A red box highlights the 'PortfolioViewCounter' row. The bottom of the screen shows standard AWS footer links: CloudShell, Feedback, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Recycle bin
PortfolioViewCounter	Active	id (\$)	-	0	0	Off		
ViewCounter	Active	id (\$)	-	0	0	Off		

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Navigate to Explore Items and click on Create Item to add a new entry.

The screenshot shows the Amazon DynamoDB Management Console interface. On the left, the navigation menu for 'DynamoDB' is visible, with 'Explore items' highlighted and a red box drawn around it. The main workspace displays the 'PortfolioViewCounter' table configuration. At the top right, there are tabs for 'Scan' (selected) and 'Query'. Below these are dropdown menus for 'Select a table or index' (set to 'Table - PortfolioViewCounter') and 'Select attribute projection' (set to 'All attributes'). A section for 'Filters - optional' is present with 'Run' and 'Reset' buttons. A green success message at the bottom left states: 'Completed · Items returned: 0 · Items scanned: 0 · Efficiency: 100% · RCUs consumed: 2'. The central area is titled 'Table: PortfolioViewCounter - Items returned (0)' and shows a message: 'Scan started on May 14, 2025, 15:15:32'. It indicates 'No items' and 'No items to display.' A prominent blue 'Create item' button is located at the bottom center of this area, also enclosed in a red box. The bottom of the screen includes standard AWS footer links: CloudShell, Feedback, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

Enter "page" as the value for the partition key "id", then click on Add New Attribute. Set the attribute type to Number, name it "count", and leave the value as 0. Finally, click on Create Item to save the changes.

The screenshot shows the 'Create item' page in the AWS DynamoDB console. The URL is https://us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#edit-item?itemMode=1&route=ROUTE_ITEM_EXPLORER&table=PortfolioViewCounter. The page title is 'Create item' under 'DynamoDB > Explore items: PortfolioViewCounter > Create item'. There are two tabs at the top right: 'Form' (selected) and 'JSON view'.

The main area is titled 'Attributes'. It shows a table with three columns: 'Attribute name', 'Value', and 'Type'.

- The first row has 'id - Partition key' in 'Attribute name', 'page' in 'Value', and 'String' in 'Type'. The 'Value' field is highlighted with a red box.
- The second row has 'count' in 'Attribute name', '0' in 'Value', and 'Number' in 'Type'. The 'Attribute name' field is highlighted with a red box.

At the bottom right of the attributes table are three buttons: 'Cancel', 'Create item' (highlighted with a yellow box), and 'Remove'.

At the very bottom of the page, there is a footer bar with links: CloudShell, Feedback, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

Creating Lambda
Function:

Create a Lambda function named "UpdatePortfolioViewCounter" with the runtime set to Python 3.13, then click on Create Function to finalize the creation.

The screenshot shows the AWS Lambda 'Create function' wizard. The 'Basic information' step is active. The 'Function name' field contains 'UpdatePortfolioViewCounter', which is highlighted with a red box. The 'Runtime' dropdown is set to 'Python 3.13', also highlighted with a red box. Other settings shown include 'x86_64' for Architecture and 'Change default execution role' for Permissions. To the right, a sidebar titled 'Tutorials' is open, showing a 'Create a simple web app' tutorial with two bullet points: 'Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage' and 'Invoke your function through its function URL'. A 'Start tutorial' button is also visible in the sidebar.

Create function [Info](#)

Choose one of the following options to create your function.

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.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

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

[CloudShell](#) [Feedback](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

The Lambda function is created.

The screenshot shows the AWS Lambda console interface. At the top, a green success message states: "Successfully created the function UpdatePortfolioViewCounter. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the function name "UpdatePortfolioViewCounter" is displayed, along with a "Function overview" section. This section includes tabs for "Diagram" (selected), "Template", and "Code". It shows the function ARN: arn:aws:lambda:us-east-1:545009862004:function:UpdatePortfolioViewCounter. On the right side of the overview, there are sections for "Description", "Last modified" (21 seconds ago), and "Function URL". A "Tutorials" sidebar is visible on the right, featuring a "Create a simple web app" tutorial. The bottom of the screen shows the "Code source" tab selected, with a code editor area containing the function's code. The AWS navigation bar at the top includes links for "Lambda", "Functions", and "UpdatePortfolioViewCounter".

Next, navigate to the Code section and replace the existing code with the code provided below. Note: The code below has already replaced the table name as "PortfolioViewCounter".

The screenshot shows the AWS Lambda function editor for the 'UpdatePortfolioViewCounter' function. The code editor displays the 'lambda_function.py' file. A red box highlights the line 'table = dynamodb.Table('ViewCounter')'. A red arrow points from this line to the text 'Replace this with table name'.

```
import json
import boto3
from botocore.exceptions import ClientError

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('ViewCounter') # DynamoDB table name
ITEM_ID = 'page' # Partition key value

def lambda_handler(event, context):
    try:
        response = table.update_item(
            Key={'id': ITEM_ID},
            UpdateExpression="SET #c = if_not_exists(#c, :zero) + :inc",
            ExpressionAttributeNames={"#c": "count"},
            ExpressionAttributeValues={":inc": 1, ":zero": 0},
            ReturnValues="UPDATED_NEW"
        )

        return {
            'statusCode': 200,
            'headers': {
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type, Authorization'
            },
            'body': json.dumps({
                'count': int(response['Attributes']['count'])
            })
        }
    except ClientError as e:
        print(e.response['Error']['Message'])


```

The right sidebar features a 'Tutorials' tab with a 'Create a simple web app' section. It includes a brief description, a bulleted list of learning objectives, and links to 'Learn more' and 'Start tutorial'.

```
import json
import boto3
from botocore.exceptions import ClientError

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('PortfolioViewCounter') # DynamoDB table name
ITEM_ID = 'page' # Partition key value

def lambda_handler(event, context):
    try:
        response = table.update_item(
            Key={'id': ITEM_ID},
            UpdateExpression="SET #c = if_not_exists(#c, :zero) + :inc",
            ExpressionAttributeNames={"#c": "count"},
            ExpressionAttributeValues={":inc": 1, ":zero": 0},
            ReturnValues="UPDATED_NEW"
        )
        return {
            'statusCode': 200,
            'headers': {
                'Access-Control-Allow-Origin': '*',

```

```
'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
'Access-Control-Allow-Headers': 'Content-Type, Authorization'
},
'body': json.dumps({
    'count': int(response['Attributes']['count'])
})
}

except ClientError as e:
    print("Error:", e)
    return {
        'statusCode': 500,
        'headers': {
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
            'Access-Control-Allow-Headers': 'Content-Type, Authorization'
        },
        'body': json.dumps({
            'error': 'Internal Server Error',
            'details': str(e)
        })
}
}
```

After pasting the code, click on Deploy on the left to save and apply the changes.

The screenshot shows the AWS Lambda function editor interface. On the left, the EXPLORER panel displays a folder named 'UPDATEPORTFOLIOVIEWCOUNTER' containing a file 'lambda_function.py'. Below it, under 'DEPLOY [UNDEPLOYED CHANGES]', there is a message: 'You have undeployed changes.' followed by two buttons: 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)'. The 'Deploy' button is highlighted with a red rectangle. The main area shows the Python code for the 'lambda_handler' function:

```
def lambda_handler(event, context):
    try:
        return {
            'headers': {
            },
            'body': json.dumps({
                'count': int(response['Attributes']['count'])
            })
        }
    except ClientError as e:
        print("Error:", e)
        return {
            'statusCode': 500,
            'headers': {
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type, Authorization'
            },
            'body': json.dumps({
                'error': 'Internal Server Error',
                'details': str(e)
            })
        }
```

On the right, the 'Tutorials' tab is selected, showing a section titled 'Create a simple web app'. It includes a brief description and a bulleted list of learning objectives:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Below this is a 'Learn more' link and a 'Start tutorial' button.

Creating API

Gateway:

Now, create an API Gateway by selecting HTTP API, then click on Build to start the configuration.

The screenshot shows the AWS API Gateway 'Create API' interface. At the top, the browser title bar reads 'API Gateway - Create API'. The URL in the address bar is 'us-east-1.console.aws.amazon.com/apigateway/main/precreate?region=us-east-1'. The navigation bar includes 'aws', a search bar, and account information 'myadmin @ 5450-0986-2004'. Below the navigation, the breadcrumb path is 'API Gateway > APIs > Create API'. The main content area is titled 'Choose an API type' with a 'Info' link. It lists three options:

- HTTP API**: Described as building low-latency and cost-effective REST APIs with built-in features like OIDC and OAuth2, and native CORS support. It works with Lambda and HTTP backends. A blue 'Import' button and an orange 'Build' button are shown, with the 'Build' button being highlighted with a red box.
- WebSocket API**: Described as building a WebSocket API for real-time use cases like chat applications or dashboards. It works with Lambda, HTTP, and AWS Services. An orange 'Build' button is shown.
- REST API**: No description or additional buttons are visible for this option.

At the bottom of the page, there are links for 'CloudShell', 'Feedback', '© 2025, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

Enter "PortfolioViewCounterAPI" as the API name, then click on Add Integration to proceed.

The screenshot shows the 'Configure API' step of creating an HTTP API. The 'API details' section is active, showing the 'API name' field filled with 'PortfolioViewCounterAPI'. The 'IP address type' section shows 'IPv4' selected. The 'Integrations (0)' section has an 'Add integration' button highlighted with a red box. At the bottom, there are 'Cancel', 'Review and create', and 'Next' buttons.

Step 1
Configure API

Step 2 - optional
Configure routes

Step 3 - optional
Define stages

Step 4
Review and create

Configure API

API details

API name
An HTTP API must have a name. The name is a non-unique value you use to identify and organize your APIs. To programmatically refer to this API, use the API ID that API Gateway generates for you.
PortfolioViewCounterAPI

IP address type | Info
Select the type of IP addresses that can invoke the default endpoint for your API. You don't need to redeploy your API for the update to take effect.
 IPv4
Includes only IPv4 addresses.
 Dualstack
Includes IPv4 and IPv6 addresses.

Integrations (0) | Info
Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For an HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.
Add integration

Cancel Review and create Next

Then, select Lambda as the integration type.

The screenshot shows the AWS API Gateway 'Create HTTP API' configuration interface. The left sidebar lists steps: Step 1 (Configure API, currently selected), Step 2 - optional (Configure routes), Step 3 - optional (Define stages), and Step 4 (Review and create). The main area is titled 'Configure API' and shows 'API details'. Under 'API name', the value 'PortfolioViewCounterAPI' is entered, and under 'Integration type', 'Lambda' is selected and highlighted with a red box. Below these fields, there are sections for 'HTTP', 'Private resource', and 'AWS service'. At the bottom of the integration section is a 'Remove' button. A large 'Add integration' button is located at the bottom left of the main form. At the very bottom of the page are navigation buttons: 'Cancel', 'Review and create' (which is highlighted in blue), and 'Next'.

Select the "UpdatePortfolioViewCounter" Lambda function from the list and Click on Next.

Step 1
Configure API

Step 2 - optional
Configure routes

Step 3 - optional
Define stages

Step 4
Review and create

Configure API

API details

API name
An HTTP API must have a name. The name is a non-unique value you use to identify and organize your APIs. To programmatically refer to this API, use the API ID that API Gateway generates for you.

IP address type | [Info](#)
Select the type of IP addresses that can invoke the default endpoint for your API. You don't need to redeploy your API for the update to take effect.
 IPv4
Includes only IPv4 addresses.
 Dualstack
Includes IPv4 and IPv6 addresses.

Integrations (1) [Info](#)
Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For an HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Lambda	<input type="text" value="arn:aws:lambda:us-east-1:545009862004:function:UpdatePortfolioViewCounter"/>	Remove
AWS Region	<input type="text" value="us-east-1"/>	<input type="text" value="2.0"/>
<input type="button" value="Add integration"/>		

[Cancel](#) [Review and create](#) **Next**

Set the method to POST, enter /view as the resource path, choose "UpdatePortfolioViewCounter" as the integration target, and then click on Next to continue.

The screenshot shows the 'Configure routes - optional' step of the API creation wizard. On the left, a vertical navigation bar lists steps: Step 1 (Configure API), Step 2 - optional (Configure routes, currently selected), Step 3 - optional (Define stages), and Step 4 (Review and create). The main area contains a 'Configure routes' section with an 'Info' link. It explains that API Gateway uses routes to expose integrations to consumers. Below this, there are three input fields: 'Method' (set to POST), 'Resource path' (set to /view), and 'Integration target' (set to UpdatePortfolioViewCounter). A red box highlights the 'Method' field, another highlights the 'Resource path' field, and a third highlights the 'Integration target' field. At the bottom right, there are 'Cancel', 'Review and create', 'Previous', and 'Next' buttons, with the 'Next' button being highlighted by a red box.

Click on Add Stage, enter "dev" as the stage name, enable the Auto deploy toggle, and then click on Next to proceed.

The screenshot shows the 'Define stages - optional' step of the API creation wizard. On the left, a vertical navigation bar lists four steps: Step 1 (Configure API), Step 2 (Configure routes), Step 3 (Define stages, currently selected and highlighted in blue), and Step 4 (Review and create). The main area contains a 'Configure stages' section with an info link. It explains that stages are environments for deployment and that changes are auto-deployed by default to the \$default stage. Below this, there's a table for managing stages:

Stage name	Auto-deploy
\$default	<input checked="" type="checkbox"/>
dev	<input checked="" type="checkbox"/>

Buttons for 'Add stage' (disabled), 'Remove' (disabled), and 'Next' (highlighted with a red box) are at the bottom right. The browser address bar shows the URL: us-east-1.console.aws.amazon.com/apigateway/main/create?region=us-east-1.

Finally, click on Create to deploy and create the API.

The screenshot shows the 'Review and create' step of the AWS API Gateway 'Create HTTP API' wizard. The left sidebar lists steps: Step 1 (Configure API), Step 2 (optional, Configure routes), Step 3 (optional, Define stages), Step 4, and the current step, Step 5 (Review and create). The main area is divided into three sections: 'API name and integrations', 'Routes', and 'Stages'. In 'API name and integrations', the API name is 'PortfolioViewCounterAPI' and the IP address type is 'IPv4'. In 'Routes', there is one route: 'POST /view → UpdatePortfolioViewCounter (Lambda)'. In 'Stages', there are two stages: '\$default (Auto-deploy: enabled)' and 'dev (Auto-deploy: enabled)'. At the bottom right, there are 'Cancel', 'Previous', and 'Create' buttons, with 'Create' being highlighted by a red box.

API Gateway - Create HTTP API

us-east-1.console.aws.amazon.com/apigateway/main/create?region=us-east-1

aws Search [Alt+S]

API Gateway > APIs > Create API > Create HTTP API

Review and create

API name and integrations

API name
PortfolioViewCounterAPI

IP address type
IPv4

Integrations

- UpdatePortfolioViewCounter (Lambda)

Routes

Routes

- POST /view → UpdatePortfolioViewCounter (Lambda)

Stages

Stages

- \$default (Auto-deploy: enabled)
- dev (Auto-deploy: enabled)

Cancel Previous Create

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Now, navigate to the CORS settings and configure it as follows: a) Set Access-Control-Allow-Origin to * b) Select GET, POST, and OPTIONS under Access-Control-Allow-Methods c) Add Content-Type and Authorization under Access-Control-Allow-Headers. Then, click on Save to apply the changes.

The screenshot shows the AWS API Gateway CORS configuration interface. On the left, a sidebar navigation includes sections for API Gateway, APIs, Custom domain names, Domain name access associations, VPC links, API: PortfolioViewCo... (9xv5bq7iwe), Develop (Routes, Authorization, Integrations, CORS, Reimport, Export), Deploy (Stages), and Monitor (Metrics, Logging). The 'CORS' link in the 'Develop' section is highlighted with a red box. The main content area is titled 'Cross-Origin Resource Sharing' and contains several configuration fields:

- Access-Control-Allow-Origin:** A text input field containing '*' with an 'Add' button next to it. This field is also highlighted with a red box.
- Access-Control-Allow-Headers:** A text input field containing 'content-type' and 'authorization' separated by a space, each with an 'X' button. This field is also highlighted with a red box.
- Access-Control-Allow-Methods:** A dropdown menu showing 'Choose Allowed Methods' with three buttons below it: 'GET X', 'POST X', and 'OPTIONS X'. This field is highlighted with a red box.
- Access-Control-Expose-Headers:** A text input field with an 'Add' button next to it.
- Access-Control-Max-Age:** A text input field containing '0'.
- Access-Control-Allow-Credentials:** A toggle switch set to 'NO'.

At the bottom right, there are 'Cancel' and 'Save' buttons. The 'Save' button is highlighted with a red box.

Now, navigate to IAM Roles, and click on the role named "UpdatePortfolioViewCounter-role" associated with the Lambda function. This role will be updated to grant the necessary permissions to access DynamoDB.

The screenshot shows the AWS IAM Roles page. On the left, there's a navigation sidebar with 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Roles' is highlighted with a red box. The main area displays a table of roles:

Role name	Trusted entities	Last activity
AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Service)	-
AWSServiceRoleForSupport	AWS Service: support (Service-Linked)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-
UpdatePortfolioViewCounter-role-mckwehxy	AWS Service: lambda	-
updateViewCount-role-0yi7zef	AWS Service: lambda	27 minutes ago

The row for 'UpdatePortfolioViewCounter-role-mckwehxy' has a blue border around it, indicating it is selected. Below the table, there are three sections: 'Roles Anywhere', 'X.509 Standard', and 'Temporary credentials'.

Roles Anywhere (Info): Authenticate your non AWS workloads and securely provide access to AWS services.

X.509 Standard: Use your own existing PKI infrastructure or use [AWS Certificate Manager Private Certificate Authority](#) to authenticate identities.

Temporary credentials: Use temporary credentials with ease and benefit from the enhanced security they provide.

At the bottom, there are links for CloudShell, Feedback, and various AWS footer links like Privacy, Terms, and Cookie preferences.

Click on Add Permissions, then select Attach Policy to proceed with granting the required permissions.

The screenshot shows the AWS IAM Roles page for a specific role named "UpdatePortfolioViewCounter-role-mckwehx". The "Permissions" tab is selected. A red box highlights the "Add permissions" button, which has a dropdown menu showing "Attach policies" and "Create inline policy".

Identity and Access Management (IAM)

Creation date
May 14, 2025, 15:23 (UTC+05:45)

Last activity
-

ARN
arn:aws:iam::545009862004:role/service-role/UpdatePortfolioViewCounter-role-mckwehx

Maximum session duration
1 hour

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

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

Filter by Type
All types

Policy name	Type	Attached entities
AWSLambdaBasicExecutionRole-38617b7a...	Customer managed	1

Permissions boundary (not set)

Generate policy based on CloudTrail events
You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

<https://us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/UpdatePortfolioViewCounter-role-mckwehx/attach-policies>

Search for DynamoDB, select the AmazonDynamoDBFullAccess policy, and then click on Add Permissions to grant the necessary access.

The screenshot shows the AWS IAM 'Add permissions' interface. The search bar at the top contains the text 'dyna'. A list of policies is displayed, with 'AmazonDynamoDBFullAccess' highlighted and selected, indicated by a blue border and checked checkbox. At the bottom right of the list, there are 'Cancel' and 'Add permissions' buttons, with 'Add permissions' also having a blue border.

Policy name	Type	Description
<input checked="" type="checkbox"/> AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon DynamoDB.
<input type="checkbox"/> AmazonDynamoDBFullAccesswithDataPipeline	AWS managed	This policy is on a deprecation path. See ...
<input type="checkbox"/> AmazonDynamoDBReadOnlyAccess	AWS managed	Provides read only access to Amazon Dyn...
<input type="checkbox"/> AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and read access to Dynamo...
<input type="checkbox"/> AWSLambdaInvocation-DynamoDB	AWS managed	Provides read access to DynamoDB Strea...

Navigate to API Gateway, click on PortfolioViewCounterAPI on the left side, then copy the Invoke URL for the dev stage.

The screenshot shows the AWS API Gateway console with the following details:

- API ID:** 9xv5bq7iwe
- Protocol:** HTTP
- Created:** 2025-05-14
- Description:** No Description
- IP address type:** IPv4
- Default endpoint:** Enabled, https://9xv5bq7iwe.execute-api.us-east-1.amazonaws.com
- ARN:** arn:aws:apigateway:us-east-1::apis/9xv5bq7iwe

Stages for PortfolioViewCounterAPI (2)

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
\$default	https://9xv5bq7iwe.execute-api.us-east-1.amazonaws.com	pw5k3h	enabled	2025-05-14
dev	https://9xv5bq7iwe.execute-api.us-east-1.amazonaws.com/dev	pw5k3h	enabled	2025-05-14

Tags (0)

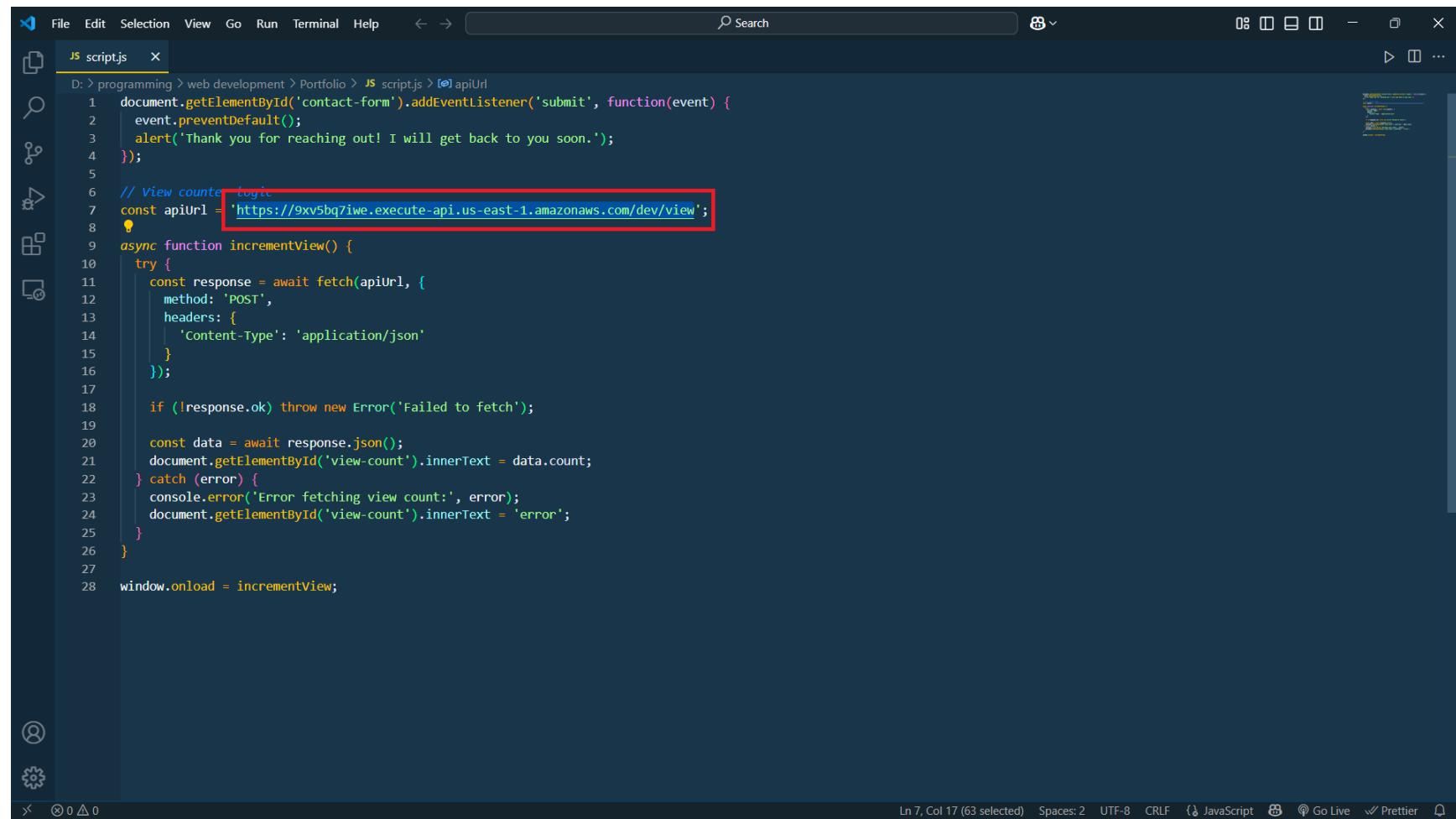
No tags

Navigation and Footer:

- Left sidebar: API Gateway, APIs, Custom domain names, Domain name access associations, VPC links, API: PortfolioViewCo... (9xv5bq7iwe) (highlighted), Develop, Deploy, Monitor.
- Bottom footer: CloudShell, Feedback, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, Cookie preferences.

Open the script.js file and replace the apiUrl with the copied URL, appending /view at the end.

Note: In the updated javascript code, you can directly replace apiUrl in fetch section.

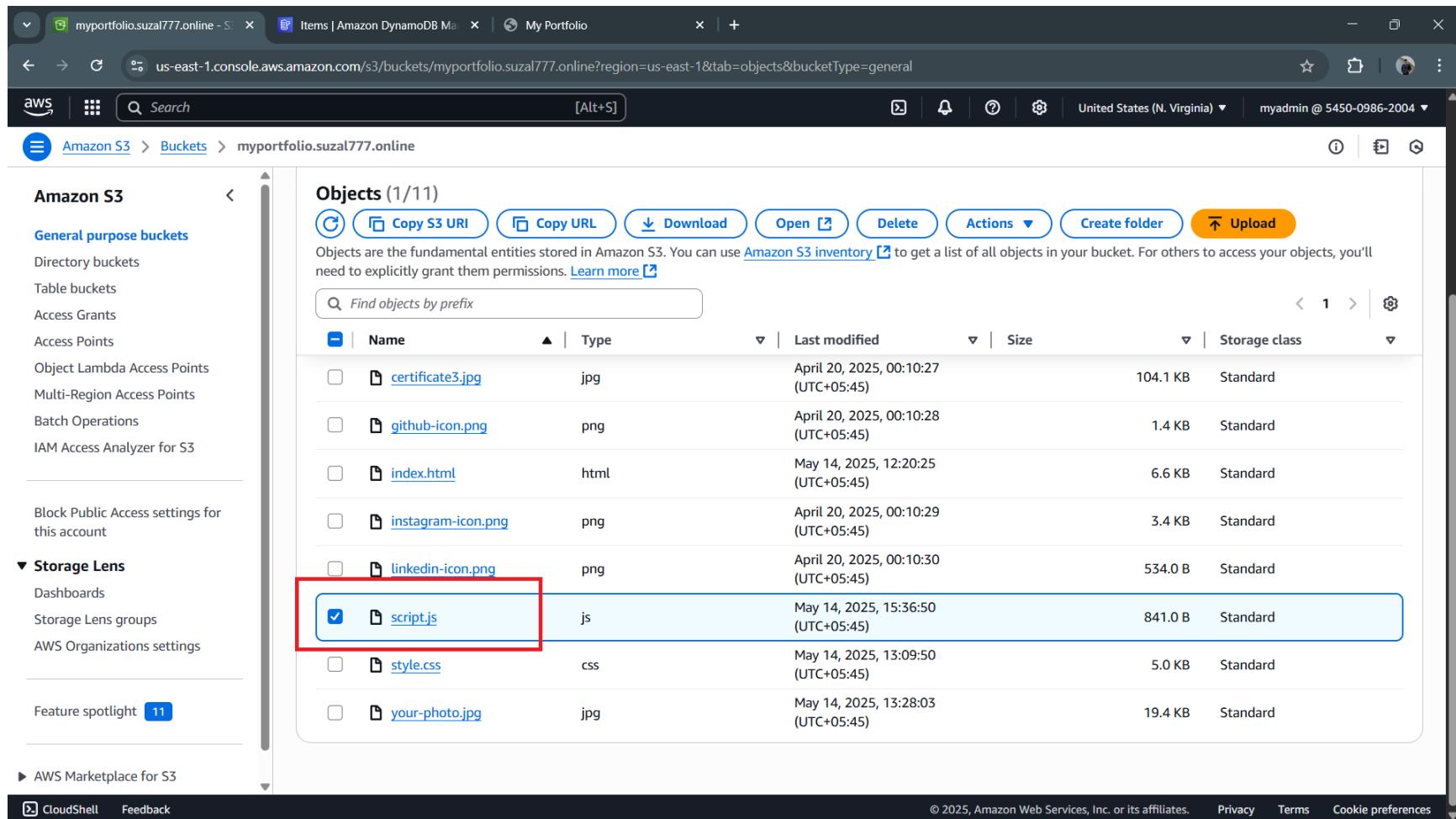


The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with a dark theme. The central area is a code editor displaying a JavaScript file named "script.js". The file contains the following code:

```
1 document.getElementById('contact-form').addEventListener('submit', function(event) {
2   event.preventDefault();
3   alert('Thank you for reaching out! I will get back to you soon.');
4 });
5
6 // View counter logic
7 const apiUrl = 'https://9xv5bq7iwe.execute-api.us-east-1.amazonaws.com/dev/view';
8
9 async function incrementView() {
10   try {
11     const response = await fetch(apiUrl, {
12       method: 'POST',
13       headers: {
14         'Content-Type': 'application/json'
15       }
16     });
17
18     if (!response.ok) throw new Error('Failed to fetch');
19
20     const data = await response.json();
21     document.getElementById('view-count').innerText = data.count;
22   } catch (error) {
23     console.error('Error fetching view count:', error);
24     document.getElementById('view-count').innerText = 'error';
25   }
26 }
27
28 window.onload = incrementView;
```

The line `const apiUrl = 'https://9xv5bq7iwe.execute-api.us-east-1.amazonaws.com/dev/view';` is highlighted with a red rectangular selection. The status bar at the bottom of the code editor shows the following information: Ln 7, Col 17 (63 selected), Spaces: 2, CRLF, { JavaScript, Go Live, Prettier, and a few other icons.

Upload the updated script.js file to your S3 bucket, replacing the previous version.



The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with navigation links like 'Amazon S3', 'General purpose buckets', 'Storage Lens', and 'AWS Marketplace for S3'. The main area is titled 'Objects (1/11)' and lists files in a table. The table columns are 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'. One file, 'script.js', is highlighted with a red box around its row, and a checkmark is visible in the checkbox column for that specific file.

Name	Type	Last modified	Size	Storage class
certificate3.jpg	jpg	April 20, 2025, 00:10:27 (UTC+05:45)	104.1 KB	Standard
github-icon.png	png	April 20, 2025, 00:10:28 (UTC+05:45)	1.4 KB	Standard
index.html	html	May 14, 2025, 12:20:25 (UTC+05:45)	6.6 KB	Standard
instagram-icon.png	png	April 20, 2025, 00:10:29 (UTC+05:45)	3.4 KB	Standard
linkedin-icon.png	png	April 20, 2025, 00:10:30 (UTC+05:45)	534.0 B	Standard
script.js	js	May 14, 2025, 15:36:50 (UTC+05:45)	841.0 B	Standard
style.css	css	May 14, 2025, 13:09:50 (UTC+05:45)	5.0 KB	Standard
your-photo.jpg	jpg	May 14, 2025, 13:28:03 (UTC+05:45)	19.4 KB	Standard

Navigate to your CloudFront distribution, go to the Invalidation section, and create a new invalidation. Add /script.js as the object path to clear the cached file.

The screenshot shows the AWS CloudFront console with the 'Copy invalidation' dialog open. The left sidebar lists various CloudFront management options like Distributions, Policies, Functions, etc. The main area shows the 'Copy invalidation' interface with a red box highlighting the 'Object paths' input field containing '/script.js'. Another red box highlights the 'Create invalidation' button at the bottom right of the dialog.

Distributions | CloudFront | Global Items | Amazon DynamoDB | My Portfolio

us-east-1.console.aws.amazon.com/cloudfront/v4/home?region=us-east-1#/distributions/E1ECBGZURCZ8U/invalidations/copy/I7C2PD0DS97Q7GXESC4ZAH0IC7

CloudFront > Distributions > E1ECBGZURCZ8U > Copy invalidation

CloudFront

Copy invalidation

Object paths Info

Add object paths

Add each object path to remove from the CloudFront cache. To use wildcards (*) in the invalidation, you must put the wildcard at the end of the path.

/script.js

To add object paths individually, use the [standard editor](#).

Cancel **Create invalidation**

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Your view counter is now operational on your portfolio website.

The screenshot shows a web browser window with three tabs at the top: 'Items | Amazon DynamoDB Ma', 'Console Home | Console Home', and 'My Portfolio'. The main content area displays the URL 'myportfolio.suzal777.online'. The page has a dark background with white text. The title 'Welcome to Suzal's Portfolio' is centered at the top. Below it is a navigation bar with links for 'About', 'Projects', 'Certificates', 'Contact', and 'Total Views: 6'. A circular profile picture of a young man is located on the left side of the page.

About Me



Hi, I'm Sujal Phaiju, a motivated engineering student with a strong foundation in cloud computing and hands-on experience with AWS services like EC2, S3, VPC, CloudFormation, CodePipeline, Lambda, and CloudFront. I am passionate about building scalable solutions and pursuing a career as a Cloud and DevOps Engineer. I am AWS Certified Solutions Architect – Associate, and currently seeking an internship to enhance my technical skills and contribute to impactful cloud projects.

Contact Me feature

Creating SNS Topic :

Go to SNS, navigate to the Topics section, and click on Create Topic. Select Standard as the topic type, enter "PortfolioTopic" as the name, and then click on Create Topic.

The screenshot shows the 'Create topic' wizard in the AWS SNS console. The 'Details' step is selected. In the 'Type' section, the 'Standard' option is highlighted with a red box and a blue selection bar, while the 'FIFO (first-in, first-out)' option is shown below it. The 'Name' field contains the value 'PortfolioTopic', which is also highlighted with a red box. The 'Display name - optional' field contains the value 'My Topic'. Below these fields, there are sections for 'Encryption - optional' and 'Access policy - optional', each with a disclosure arrow. At the bottom of the page, there are links for CloudShell, Feedback, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

Create topic

Details

Type | [Info](#)
Topic type cannot be modified after topic is created

FIFO (first-in, first-out)
• Strictly-preserved message ordering
• Exactly-once message delivery
• Subscription protocols: SQS

Standard
• Best-effort message ordering
• At-least once message delivery
• Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

Name

PortfolioTopic

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - optional | [Info](#)
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

My Topic

Maximum 100 characters.

► **Encryption - optional** [Info](#)
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

► **Access policy - optional** [Info](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Click on the "PortfolioTopic" you just created, then click on Create Subscription to add a subscriber.

The screenshot shows the AWS SNS console for the 'PortfolioTopic' in the 'Topics' section. The left sidebar includes links for Dashboard, Topics (selected), Subscriptions, and Mobile (Push notifications, Text messaging (SMS)). The main content area displays the 'PortfolioTopic' details: Name (PortfolioTopic), ARN (arn:aws:sns:us-east-1:545009862004:PortfolioTopic), Type (Standard), Display name (-), and Topic owner (545009862004). Below this, the 'Subscriptions' tab is selected, showing one existing subscription:

ID	Endpoint	Status	Protocol
b8c8bbe2-742b-4f65-831e-791d6f...	sujalphaiju777@gmail.com	Confirmed	EMAIL

A red box highlights the 'Create subscription' button at the top right of the subscriptions table. Other buttons visible include Edit, Delete, Request confirmation, Confirm subscription, and Tags.

Set the Protocol to Email and enter your Gmail address as the Endpoint. Click on Create Subscription, then go to your inbox and confirm the subscription by clicking the link in the confirmation email.

The screenshot shows the 'Create subscription' wizard in the AWS SNS console. The 'Topic ARN' field contains 'arn:aws:sns:us-east-1:545009862004:PortfolioTopic'. The 'Protocol' dropdown is set to 'Email', which is highlighted with a red box. The 'Endpoint' field contains 'sujalphaiju777@gmail.com', also highlighted with a red box. A note below the endpoint field states: 'After your subscription is created, you must confirm it.' This note is also highlighted with a red box. At the bottom right, there are 'Cancel' and 'Create subscription' buttons, with 'Create subscription' being highlighted with a red box.

Topic ARN
arn:aws:sns:us-east-1:545009862004:PortfolioTopic

Protocol
Email

Endpoint
sujalphaiju777@gmail.com

After your subscription is created, you must confirm it. [Info](#)

Subscription filter policy - optional [Info](#)
This policy filters the messages that a subscriber receives.

Redrive policy (dead-letter queue) - optional [Info](#)
Send undeliverable messages to a dead-letter queue.

Create subscription

Creating Lambda
Function:

Go to Lambda, create a new function named "PortfolioContact", select Python 3.13 as the runtime, and then click on Create Function to proceed.

The screenshot shows the AWS Lambda 'Create function' wizard. The 'Basic information' step is active, with the following details:

- Function name:** PortfolioContact (highlighted with a red box)
- Runtime:** Python 3.13 (highlighted with a red box)
- Architecture:** x86_64
- Permissions:** Change default execution role
- Additional Configurations:** Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

On the right side, there is a sidebar titled 'Tutorials' which includes a section on 'Create a simple web app' with a 'Start tutorial' button.

At the bottom right of the main form, there are 'Cancel' and 'Create function' buttons, with 'Create function' highlighted by a red box.

Copy the code provided below and paste it into the Code section of your PortfolioContact Lambda function. Make sure to update the TopicArn with the ARN of your SNS topic, then click on Deploy to apply the changes.

The screenshot shows the AWS Lambda Functions console with the 'PortfolioContact' function selected. The 'Code' tab is active. On the left, the 'EXPLORER' sidebar shows a folder named 'PORTFOLIOCONTACT' containing a file 'lambda_function.py'. Below the sidebar, there are 'DEPLOY' and 'TEST EVENTS' sections. The 'DEPLOY' section contains two buttons: 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)', with 'Deploy (Ctrl+Shift+U)' highlighted by a red box. The main area is a code editor titled 'lambda_function.py' with the following content:

```
lambda_function.py
1 #!/usr/bin/python3
2
3 def lambda_handler(event, context):
4     try:
5         Email: {email}
6         Message: {message}
7         """
8
9         # Send to SNS
10        response = sns.publish(
11            TopicArn='arn:aws:sns:us-east-1:545009862004:PortfolioTopic',
12            Subject='Website Form Submission',
13            Message=sns_message
14        )
15
16        return {
17            'statusCode': 200,
18            'body': json.dumps({'message': 'Notification sent successfully'})
19        }
20
21    except Exception as e:
22        return {
23            'statusCode': 500,
24            'body': json.dumps({'error': str(e)})
25        }
26
27
28
29
30
31
32
33
34
35
36
37
38
```

The code editor has a red border around the code area. To the right of the code editor, there is a 'Tutorials' sidebar with a 'Create a simple web app' section. This section includes a brief description, a bulleted list of learning objectives, and 'Learn more' and 'Start tutorial' buttons.

```
import json
import boto3

sns = boto3.client('sns')

def lambda_handler(event, context):
    try:
        # Parse the incoming request body
        body = json.loads(event['body'])

        name = body.get('name')
        email = body.get('email')
        message = body.get('message')

        # Format the SNS message
        sns_message = f"""
New Contact Form Submission:
Name: {name}
Email: {email}
Message: {message}
"""

        # Send to SNS
        response = sns.publish(
            TopicArn='arn:aws:sns:us-east-1:545009862004:PortfolioTopic',
            Subject='Website Form Submission',
            Message=sns_message
        )

        return {
            'statusCode': 200,
            'body': json.dumps({'message': 'Notification sent successfully'})
    
```

```
}

except Exception as e:
    return {
        'statusCode': 500,
        'body': json.dumps({'error': str(e)})
}
```

Navigate to IAM Roles and select the role named "PortfolioContact-role" associated with your PortfolioContact Lambda function.

The screenshot shows the AWS IAM Roles page. The left sidebar has sections for Identity and Access Management (IAM) like Dashboard, Access management (User groups, Users, Roles, Policies, Identity providers, Account settings), and Access reports (Access Analyzer, External access, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies). The main area shows a table titled 'Roles (5)' with columns for Role name, Trusted entities, and Last activity. A role named 'PortfolioContact-role-4jbxw3t1' is highlighted with a red box. Below the table, there are sections for 'Roles Anywhere' (Access AWS from your non AWS workloads using X.509 Standard or AWS Certificate Manager Private Certificate Authority), and 'Temporary credentials' (using temporary credentials for enhanced security).

Role name	Trusted entities	Last activity
AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Service)	-
AWSServiceRoleForSupport	AWS Service: support (Service-Linked)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-
PortfolioContact-role-4jbxw3t1	AWS Service: lambda	28 minutes ago
UpdatePortfolioViewCounter-role-mckwhehx	AWS Service: lambda	12 minutes ago

Attach the AmazonSNSFullAccess policy to the PortfolioContact-role to grant the necessary permissions for publishing messages to the SNS topic.

The screenshot shows the AWS IAM Roles page for the role **PortfolioContact-role-4jbxw3t1**. The left sidebar shows the navigation path: IAM > Roles > PortfolioContact-role-4jbxw3t1. The main area displays the **Summary** and **Permissions** tabs. In the **Permissions** tab, there are two attached policies listed:

Policy name	Type	Attached entities
AmazonSNSFullAccess	AWS managed	1
AWSLambdaBasicExecutionRole-02e88e9e...	Customer managed	1

A red box highlights the **Add permissions** button and the **Attach policies** dropdown menu in the top right corner of the permissions section.

Page URL: <https://us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details/PortfolioContact-role-4jbxw3t1/attach-policies>

Creating API Gateway:

Go to API Gateway, create a new API, select HTTP API, and click on Build to begin the setup process.

The screenshot shows the AWS API Gateway 'Create API' interface. The browser title bar reads 'API Gateway - Create API'. The URL is 'us-east-1.console.aws.amazon.com/apigateway/main/precreate?region=us-east-1'. The navigation bar includes 'aws', a search bar, and account information 'myadmin @ 5450-0986-2004'. The main content area has three sections:

- HTTP API**: Describes building low-latency and cost-effective REST APIs. It lists 'Works with the following:' Lambda, HTTP backends. A blue 'Import' button and an orange 'Build' button are shown, with the 'Build' button being highlighted by a red box.
- WebSocket API**: Describes building a WebSocket API for real-time use cases. It lists 'Works with the following:' Lambda, HTTP, AWS Services. An orange 'Build' button is shown.
- REST API**: Describes developing a REST API with complete control over requests and responses. It lists 'Works with the following:' Lambda, HTTP, AWS Services. An orange 'Build' button is shown.

At the bottom of the page are links for 'CloudShell', 'Feedback', '© 2025, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

Enter "PortfolioContactAPI" as the API name. Choose Lambda as the integration type, and select the "PortfolioContact" Lambda function you just created.

The screenshot shows the AWS API Gateway 'Create HTTP API' configuration interface. The 'Configure API' step is active. In the 'API details' section, the 'API name' field contains 'PortfolioContactAPI'. The 'IP address type' is set to 'IPv4'. Under 'Integrations', there is one entry for 'Lambda' with the function ARN 'arn:aws:lambda:us-east-1:545009862004:function:PortfolioCo'. The 'Next' button at the bottom right is highlighted with a red box.

Step 1
Configure API

Step 2 - optional
Configure routes

Step 3 - optional
Define stages

Step 4
Review and create

Configure API

API details

API name
An HTTP API must have a name. The name is a non-unique value you use to identify and organize your APIs. To programmatically refer to this API, use the API ID that API Gateway generates for you.

IP address type | [Info](#)
Select the type of IP addresses that can invoke the default endpoint for your API. You don't need to redeploy your API for the update to take effect.

IPv4
Includes only IPv4 addresses.

Dualstack
Includes IPv4 and IPv6 addresses.

Integrations (1) | [Info](#)
Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For an HTTP integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Lambda

AWS Region: us-east-1

Lambda function: arn:aws:lambda:us-east-1:545009862004:function:PortfolioCo

Version: 2.0

Add integration

Cancel | Review and create | **Next**

Set the method to POST and specify /submit as the resource path.

The screenshot shows the 'Configure routes - optional' step of the 'Create HTTP API' wizard. On the left, a sidebar lists steps: Step 1 (Configure API), Step 2 - optional (Configure routes, which is selected and highlighted in blue), Step 3 - optional (Define stages), and Step 4 (Review and create). The main area contains a 'Configure routes' section with an 'Info' link. It explains that API Gateway uses routes to expose integrations to consumers of your API. Routes for HTTP APIs consist of two parts: an HTTP method and a resource path (e.g., GET /pets). You can define specific HTTP methods for your integration (GET, POST, PUT, PATCH, HEAD, OPTIONS, and DELETE) or use the ANY method to match all methods that you haven't defined on a given resource. Below this, there are three input fields: 'Method' (set to POST), 'Resource path' (set to /submit), and 'Integration target' (set to PortfolioContact). A red box highlights the 'Method' and 'Resource path' fields. At the bottom right, there are 'Cancel', 'Review and create', 'Previous', and 'Next' buttons, with the 'Next' button being highlighted in orange.

Add a stage named prod and enable Auto Deploy.

Step 1
Configure API

Step 2 - optional
Configure routes

Step 3 - optional
Define stages

Step 4
Review and create

Define stages - optional

Configure stages Info

Stages are independently configurable environments that your API can be deployed to. You must deploy to a stage for API configuration changes to take effect, unless that stage is configured to autodeploy. By default, all HTTP APIs created through the console have a default stage named \$default. All changes that you make to your API are autodeployed to that stage. You can add stages that represent environments such as development or production.

Stage name	Auto-deploy
\$default	<input checked="" type="checkbox"/>
prod	<input checked="" type="checkbox"/>

[Add stage](#)

[Cancel](#) [Previous](#) **Next**

Review your settings and click on Create API to finalize the setup.

The screenshot shows the 'Review and create' step of the AWS API Gateway 'Create HTTP API' wizard. The left sidebar lists steps: Step 1 (Configure API), Step 2 (optional, Configure routes), Step 3 (optional, Define stages), Step 4, and the current step, Step 5 (Review and create). The main area is divided into three sections: 'API name and integrations', 'Routes', and 'Stages'. In 'API name and integrations', the API name is 'PortfolioContactAPI' and the IP address type is 'IPv4'. In 'Routes', there is one route: 'POST /submit → PortfolioContact (Lambda)'. In 'Stages', there are two stages: '\$default (Auto-deploy: enabled)' and 'prod (Auto-deploy: enabled)'. At the bottom right, there are 'Cancel', 'Previous', and 'Create' buttons, with 'Create' being highlighted by a red box.

API Gateway - Create HTTP API

us-east-1.console.aws.amazon.com/apigateway/main/create?region=us-east-1

aws Search [Alt+S]

API Gateway > APIs > Create API > Create HTTP API

Review and create

API name and integrations

API name
PortfolioContactAPI

IP address type
IPv4

Integrations

- PortfolioContact (Lambda)

Routes

Routes

- POST /submit → PortfolioContact (Lambda)

Stages

Stages

- \$default (Auto-deploy: enabled)
- prod (Auto-deploy: enabled)

Cancel Previous Create

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Navigate to the CORS settings for the API. Configure the settings exactly as you did for the previous API by setting Access-Control-Allow-Origin to *. Allow the methods GET, POST, and OPTIONS, and add Content-Type and Authorization to the allowed headers. Finally, save the changes.

The screenshot shows the AWS API Gateway CORS configuration page. The URL in the browser is `us-east-1.console.aws.amazon.com/apigateway/main/develop/cors?api=vrls7lrjg5®ion=us-east-1&routes=98lqd3e`. The left sidebar shows navigation options like APIs, Custom domain names, Domain name access associations, VPC links, and sections for Develop, Deploy, Monitor, and Protect. The main content area is titled "Cross-Origin Resource Sharing" and contains several configuration fields:

- Access-Control-Allow-Origin:** A text input field containing "*" (which is highlighted with a red box). An "Add" button is next to it.
- Access-Control-Allow-Headers:** A text input field containing "content-type" and "authorization" (both highlighted with red boxes). An "Add" button is next to it.
- Access-Control-Allow-Methods:** A dropdown menu showing "Choose Allowed Methods" with three selected items: "GET", "POST", and "OPTIONS" (all highlighted with red boxes).
- Access-Control-Expose-Headers:** A text input field with an "Add" button.
- Access-Control-Max-Age:** A text input field containing "0".
- Access-Control-Allow-Credentials:** A toggle switch set to "NO".

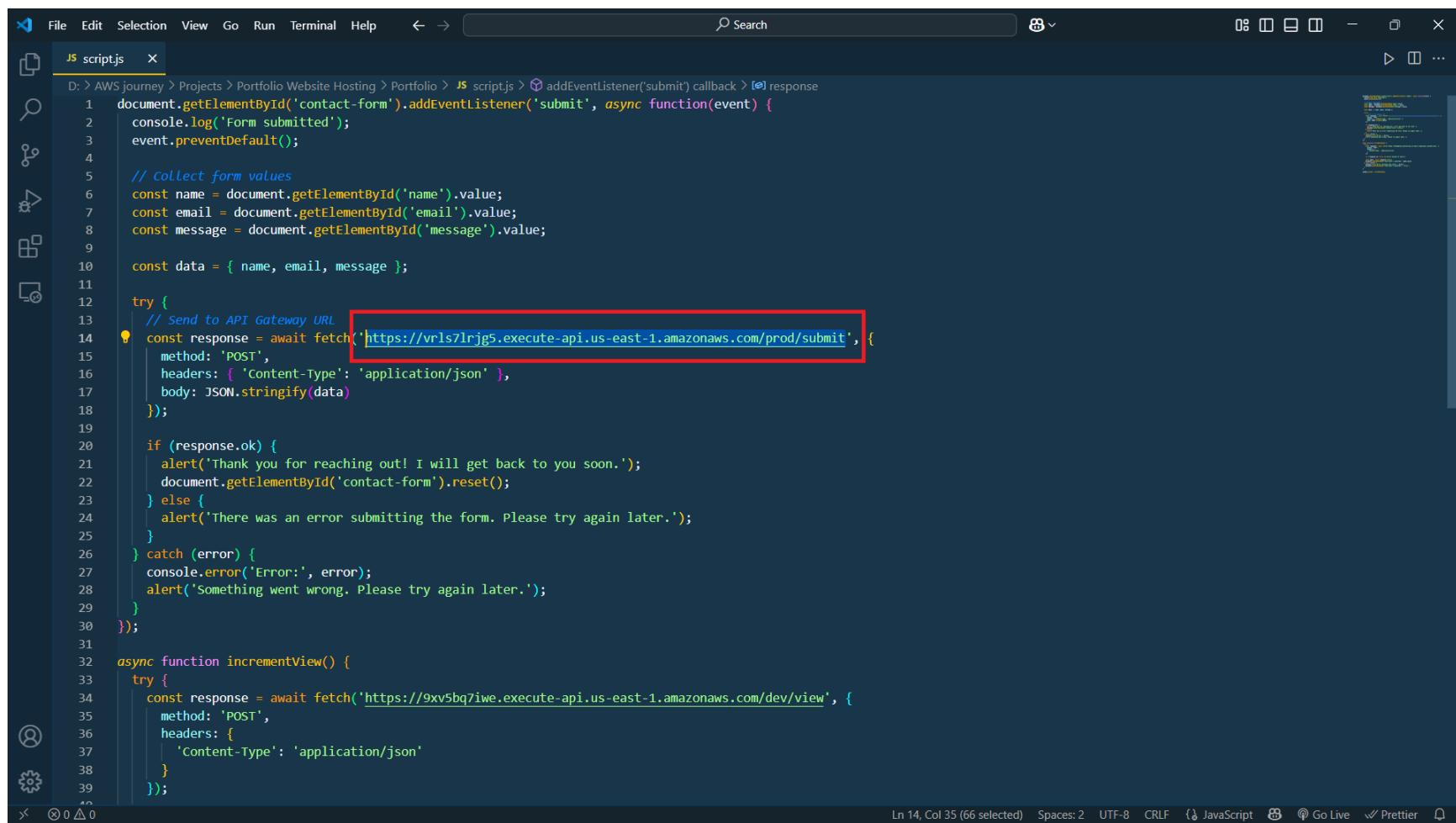
At the bottom right, there are "Cancel" and "Save" buttons, with "Save" being highlighted with a red box.

Click on your API and copy the Invoke URL for the prod stage.

The screenshot shows the AWS API Gateway API settings page for the 'PortfolioContactAPI' (vrls7lrjg5). The left sidebar has sections for API Gateway (selected), APIs, Custom domain names, Domain name access associations, VPC links, and a expanded section for 'API: PortfolioContactAPI (vrls7lrjg5)' which is highlighted with a red box. Below these are sections for Develop (Routes, Authorization, Integrations, CORS, Reimport, Export), Deploy (Stages), and Monitor (Metrics, Logging). The main content area shows the ARN (arn:aws:apigateway:us-east-1::apis/vrls7lrjg5) and the 'Stages for PortfolioContactAPI (2)' table. The table has columns for Stage name, Invoke URL, Attached deployment, Auto deploy, and Last updated. It lists two stages: '\$default' with Invoke URL <https://vrls7lrjg5.execute-api.us-east-1.amazonaws.com> and 'prod' with Invoke URL <https://vrls7lrjg5.execute-api.us-east-1.amazonaws.com/prod>. The 'prod' row is also highlighted with a red box. Below the table is a 'Tags (0)' section with a 'Manage tags' button, indicating 'No tags' associated with the resource. The bottom of the page includes CloudShell, Feedback, and standard footer links for Privacy, Terms, and Cookie preferences.

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
\$default	https://vrls7lrjg5.execute-api.us-east-1.amazonaws.com	qwhjof	enabled	2025-05-17
prod	https://vrls7lrjg5.execute-api.us-east-1.amazonaws.com/prod	qwhjof	enabled	2025-05-17

Paste the copied URL into your script.js file, appending /submit at the end. Update the script.js file in your S3 bucket with the new version. Then, create an invalidation in CloudFront to clear the cached file. If necessary, perform a hard refresh in your browser to see the changes.



```
File Edit Selection View Go Run Terminal Help ⏪ ⏩ Search D: > AWS journey > Projects > Portfolio Website Hosting > Portfolio > JS script.js > addEventListener('submit') callback > response
1 document.getElementById('contact-form').addEventListener('submit', async function(event) {
2   console.log('Form submitted');
3   event.preventDefault();
4
5   // Collect form values
6   const name = document.getElementById('name').value;
7   const email = document.getElementById('email').value;
8   const message = document.getElementById('message').value;
9
10  const data = { name, email, message };
11
12  try {
13    // Send to API Gateway URL
14    const response = await fetch('https://vr1s7lrjg5.execute-api.us-east-1.amazonaws.com/prod/submit', {
15      method: 'POST',
16      headers: { 'Content-Type': 'application/json' },
17      body: JSON.stringify(data)
18    });
19
20    if (response.ok) {
21      alert('Thank you for reaching out! I will get back to you soon.');
22      document.getElementById('contact-form').reset();
23    } else {
24      alert('There was an error submitting the form. Please try again later.');
25    }
26  } catch (error) {
27    console.error('Error:', error);
28    alert('Something went wrong. Please try again later.');
29  }
30});
31
32 async function incrementView() {
33  try {
34    const response = await fetch('https://9xv5bq7iwe.execute-api.us-east-1.amazonaws.com/dev/view', {
35      method: 'POST',
36      headers: {
37        'Content-Type': 'application/json'
38      }
39    });
}
Ln 14, Col 35 (66 selected) Spaces: 2 UTF-8 CRLF ⚙️ JavaScript ⚙️ Go Live ✅ Prettier
```