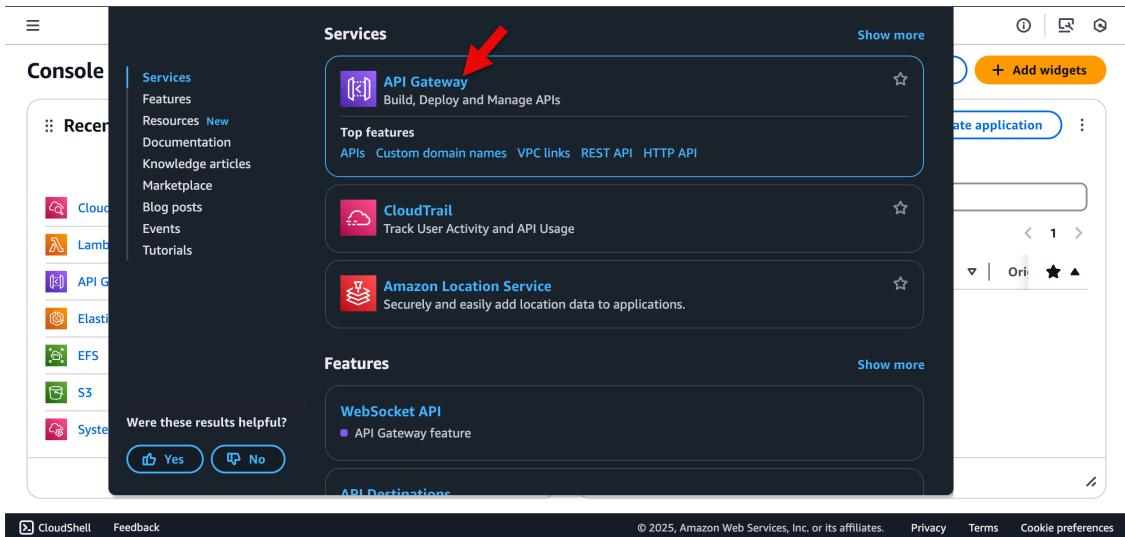


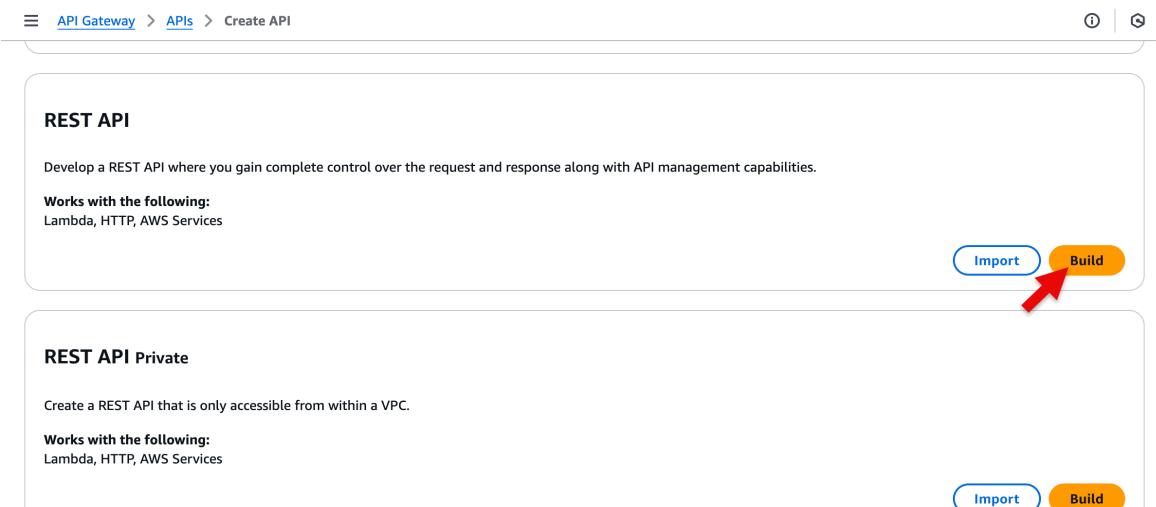
ANASIEZE IKENNA – CLOUD ENGINEER

Task 1 - Create an API Gateway with a Mock Integration

- Open the **AWS Management Console** and navigate to **API Gateway**



- Click **Build** under **REST API**.



- Set the Resource name to message, and click **Create resource**. This will automatically set the resource path to /message.

Create resource

Resource details

Proxy resource Info
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**

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

[Cancel](#) **Create resource** 

- Set **Integration type** to **Mock** and click **Create method**.

Create method

Method details

Method type **GET**

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.


Method request settings

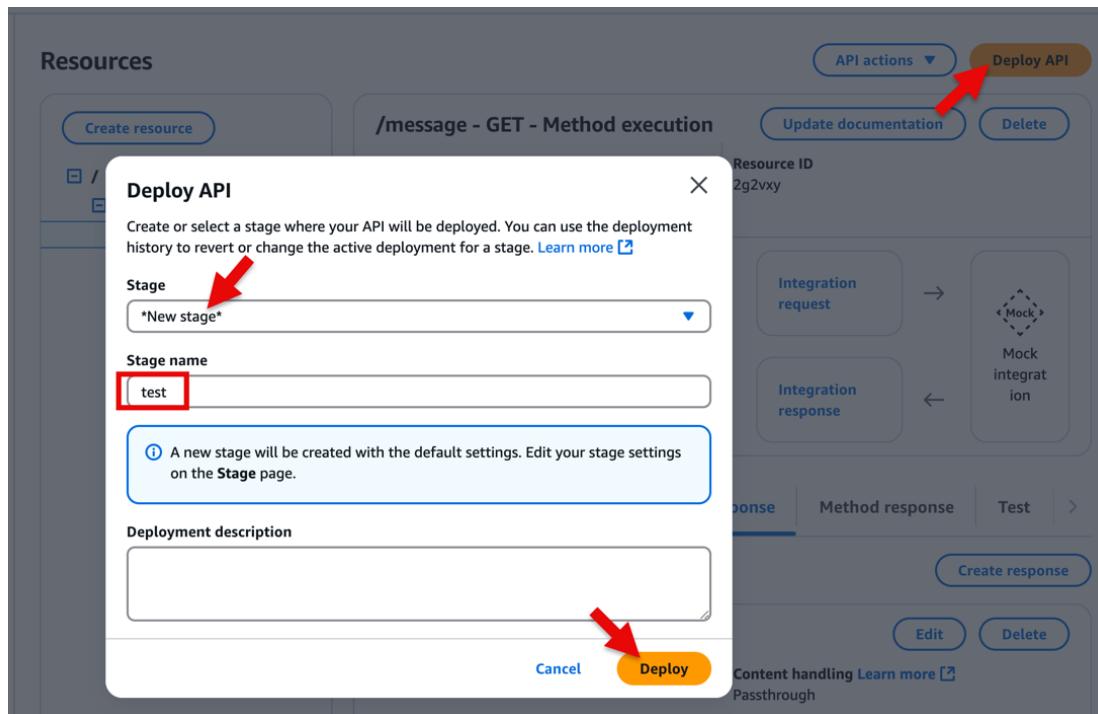
URL query string parameters

HTTP request headers

Request body

[Cancel](#) **Create method** 

- Click **Deploy API** button and choose to create a **New Stage**. Name the stage as test. Click **Deploy**.



Task 2 - Create an AWS Lambda Function

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: petroleum

Runtime: Node.js 24.x

Durable execution - new

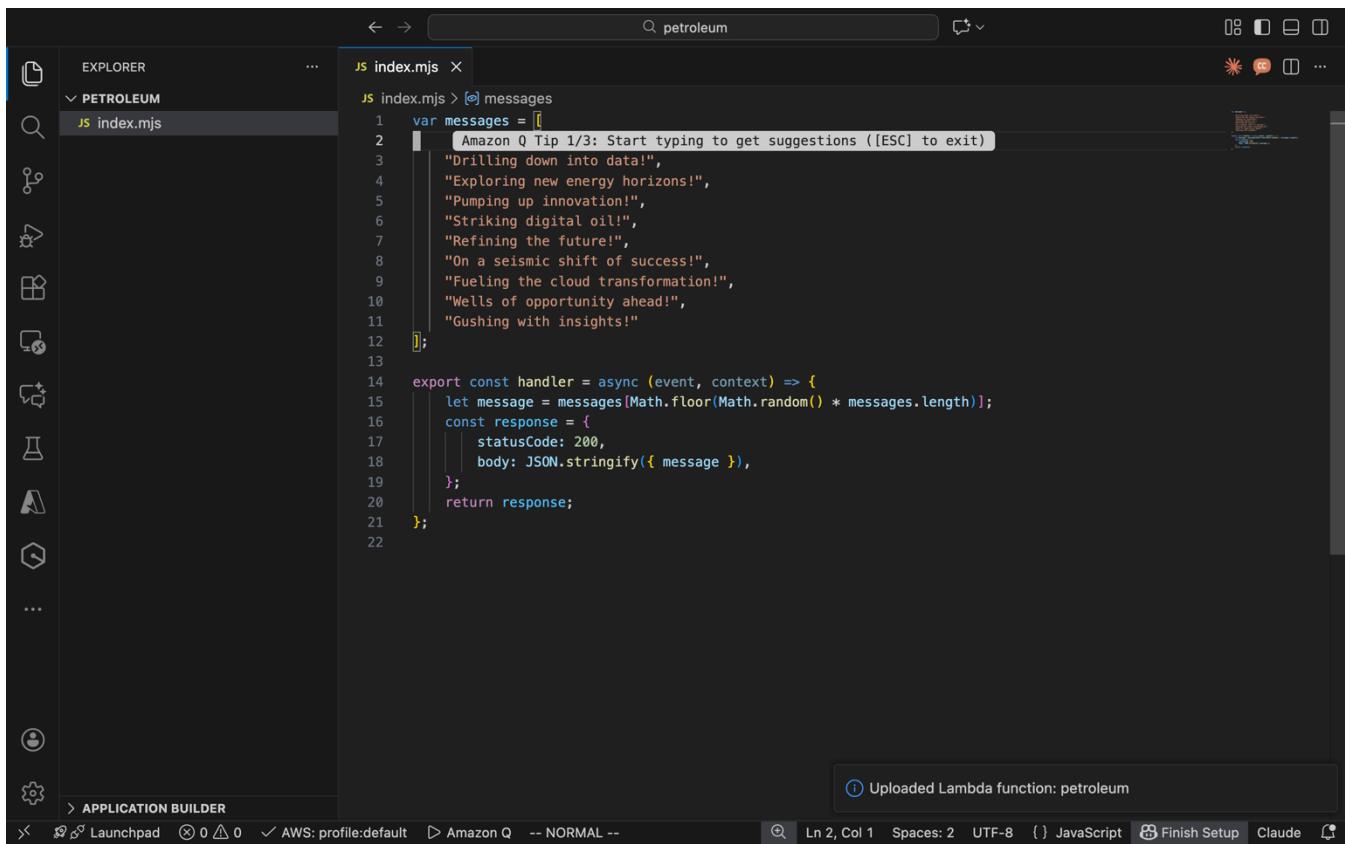
Architecture: x86_64

Permissions

CloudShell Feedback Console mobile app

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

- Replace the default **CODE** & deploy



```

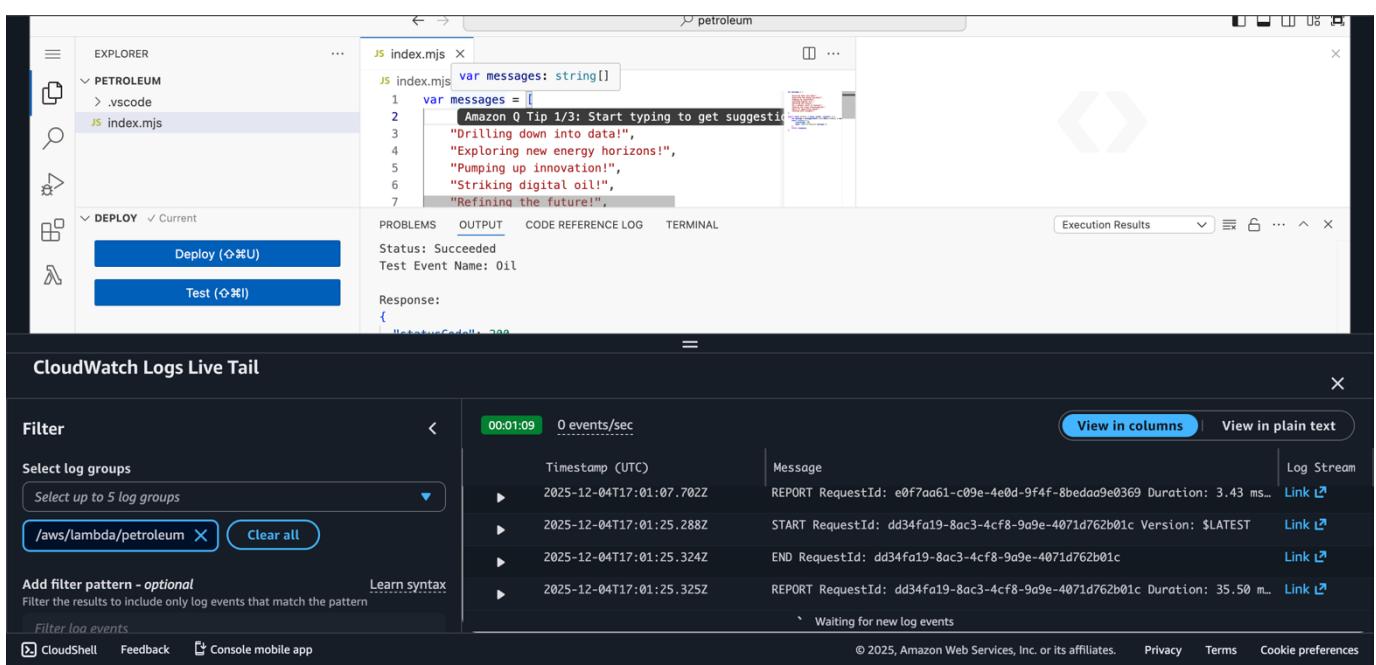
JS index.mjs x
JS index.mjs > [messages
1 var messages = [
2   "Drilling down into data!",
3   "Exploring new energy horizons!",
4   "Pumping up innovation!",
5   "Striking digital oil!",
6   "Refining the future!",
7   "On a seismic shift of success!",
8   "Fueling the cloud transformation!",
9   "Wells of opportunity ahead!",
10  "Gushing with insights!"
11];
12
13
14 export const handler = async (event, context) => {
15   let message = messages[Math.floor(Math.random() * messages.length)];
16   const response = {
17     statusCode: 200,
18     body: JSON.stringify({ message }),
19   };
20   return response;
21 };
22

```

Uploaded Lambda function: petroleum

Ln 2, Col 1 Spaces: 2 UTF-8 {} JavaScript

- Click **Test** to execute the function with test event JSON as input.



CloudWatch Logs Live Tail

Filter

Select log groups

Timestamp (UTC) 0 events/sec

Message

REPORT RequestId: e0f7aa61-c09e-4e0d-9f4f-8bedaa9e0369 Duration: 3.43 ms...

START RequestId: dd34fa19-8ac3-4cf8-9a9e-4071d762b01c Version: \$LATEST

END RequestId: dd34fa19-8ac3-4cf8-9a9e-4071d762b01c

REPORT RequestId: dd34fa19-8ac3-4cf8-9a9e-4071d762b01c Duration: 35.50 ms...

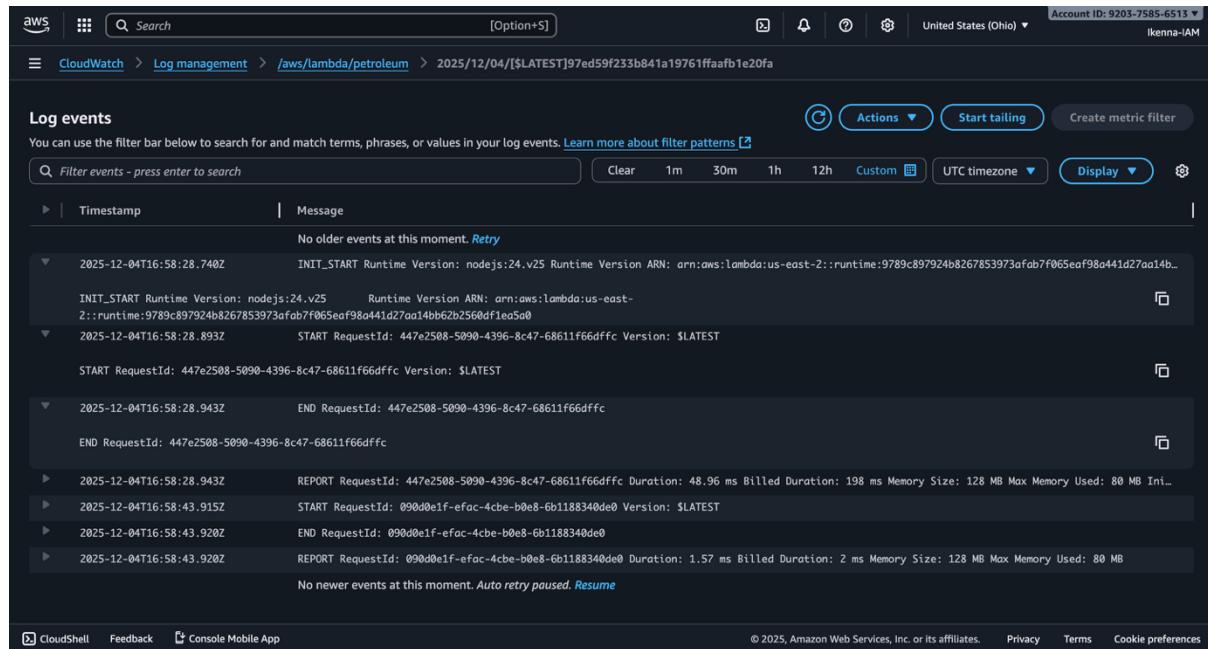
Waiting for new log events

CloudShell Feedback Console mobile app

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

Task 3 - Set up CloudWatch Live Tail

- On the **Code** tab, choose **Run and Debug**

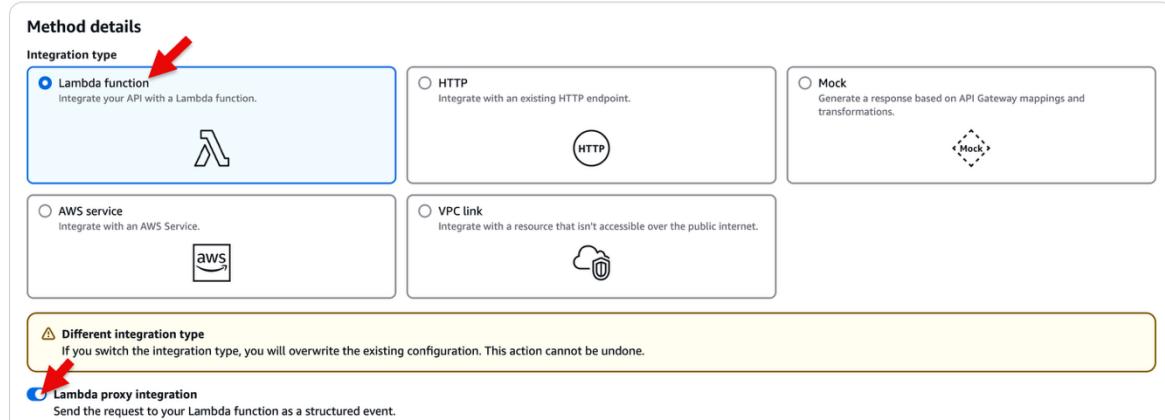


The screenshot shows the AWS CloudWatch Log management interface. The URL is `/aws/lambda/petroleum`. The log events table has two columns: 'Timestamp' and 'Message'. The 'Message' column contains log entries for a Lambda function execution. The entries include:

- 2025-12-04T16:58:28.740Z: INIT_START Runtime Version: nodejs:24.v25 Runtime Version ARN: arn:aws:lambda:us-east-2::runtime:9789c897924b8267853973afab7f065eaf98a441d27aa14b..
- 2025-12-04T16:58:28.893Z: START RequestId: 447e2508-5090-4396-8c47-68611f66dff Version: \$LATEST
- 2025-12-04T16:58:28.943Z: END RequestId: 447e2508-5090-4396-8c47-68611f66dff
- 2025-12-04T16:58:28.943Z: REPORT RequestId: 447e2508-5090-4396-8c47-68611f66dff Duration: 48.96 ms Billed Duration: 198 ms Memory Size: 128 MB Max Memory Used: 80 MB Init Duration: 100 ms
- 2025-12-04T16:58:43.915Z: START RequestId: 090d0e1f-efac-4cbe-b0e8-6b1188340de0 Version: \$LATEST
- 2025-12-04T16:58:43.920Z: END RequestId: 090d0e1f-efac-4cbe-b0e8-6b1188340de0
- 2025-12-04T16:58:43.920Z: REPORT RequestId: 090d0e1f-efac-4cbe-b0e8-6b1188340de0 Duration: 1.57 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 80 MB

Task 4 - Integrate API Gateway with AWS Lambda

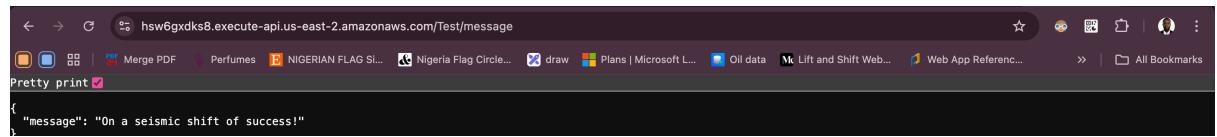
Edit integration request



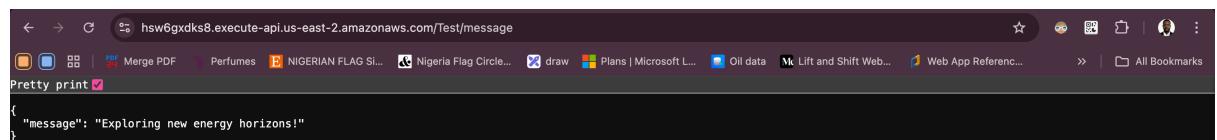
The screenshot shows the 'Edit integration request' page. The 'Method details' section has a 'Integration type' dropdown. The 'Lambda function' option is selected and highlighted with a blue border. A red arrow points to this selection. Below the dropdown, a warning message is displayed in a yellow box: '⚠ Different integration type If you switch the integration type, you will overwrite the existing configuration. This action cannot be undone.' At the bottom of the integration type list, there is another option: 'Lambda proxy integration' with a red arrow pointing to it.

- Visit the **Invoke URL** of the GET method, and test the API in a web browser & Refresh multiple times to see a **different message** each time.

```
Pretty print 
{
  "message": "Exploring new energy horizons!"
}
```



```
hsw6gxdks8.execute-api.us-east-2.amazonaws.com/Test/message
Pretty print 
{
  "message": "On a seismic shift of success!"
}
```



```
hsw6gxdks8.execute-api.us-east-2.amazonaws.com/Test/message
Pretty print 
{
  "message": "Exploring new energy horizons!"
}
```