

## Q2) Serverless API

Creating lambda functions get, post, delete, put

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Basic information' section is filled out with:

- Function name:** Get-API
- Runtime:** Python 3.12
- Architecture:** x86\_64

The 'Permissions' section shows the default execution role 'Lambda' selected. The 'Advanced settings' section is collapsed.

**Create Function** button is visible at the bottom right.

The screenshot shows the 'Get-API' function details page. The 'Function overview' section includes:

- Diagram:** Shows a single function node labeled 'Get-API'.
- Description:** Last modified 1 minute ago.
- Function ARN:** arnaws:lambda:us-east-1:462972487428:function:Get-API
- Function URL:** Info

The 'Code' tab is active, showing the AWS Lambda code editor interface with tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The code editor shows a single file named 'lambda\_function.py' with the following content:

```
print("Hello world")
```

Bottom navigation bar includes CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

The screenshot shows the AWS Lambda function editor interface. At the top, there's a navigation bar with 'Services' and a search bar. Below it, a toolbar with 'Add trigger' and other options. The main area has tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. Under 'Code', there's a 'Code source' tab with a file browser showing a 'lambda\_function' folder containing 'lambda\_function.py'. The code in the file is:

```
1 import json
2
3 def lambda_handler(event, context):
4     return "Hello from get_lambda"
```

On the left, there's an 'Environment' sidebar. At the bottom, there are tabs for 'CloudShell' and 'Feedback', along with copyright information.

The screenshot shows the 'Create function' wizard in the AWS Lambda console. It starts with a title 'Create function' and a note about choosing a creation method. Three options are shown:

- Author from scratch
- Use a blueprint
- Container image

Below this is the 'Basic information' section, which includes fields for:

- Function name: Put-API
- Runtime: Python 3.12
- Architecture: x86\_64
- Permissions: A collapsed section for changing the default execution role, which includes options for creating a new role or using an existing one.
- Advanced settings: A collapsed section for more advanced configuration.

At the bottom right, there are 'Cancel' and 'Create function' buttons.

Screenshot of the AWS Lambda Function Overview page for "Put-API".

**Function Overview:**

- Diagram:** Shows the function structure with "Put-API" at the top and "Layers" below it.
- Add trigger:** Button to add triggers.
- Add destination:** Button to add destinations.
- Description:** Last modified 36 seconds ago.
- Function ARN:** arn:aws:lambda:us-east-1:462972487428:function:Put-API
- Function URL:** Info

**Code Source:**

- Code:** Tab selected.
- Test:** Tab.
- Monitor:** Tab.
- Configuration:** Tab.
- Aliases:** Tab.
- Versions:** Tab.

**Code source:**

- Info:** Info link.
- Upload from:** Upload button.

**Test:** Test tab.

**Deploy:** Deploy button.

**Changes not deployed:** Status message.

Screenshot of the AWS Lambda Function Overview page for "Put-API".

**Function Overview:**

- Add trigger:** Button to add triggers.
- Add destination:** Button to add destinations.
- Function ARN:** arn:aws:lambda:us-east-1:462972487428:function:Put-API
- Function URL:** Info

**Code Source:**

- Code:** Tab selected.
- Test:** Tab.
- Monitor:** Tab.
- Configuration:** Tab.
- Aliases:** Tab.
- Versions:** Tab.

**Code source:**

- Info:** Info link.
- Upload from:** Upload button.

**Test:** Test tab.

**Deploy:** Deploy button.

**Changes not deployed:** Status message.

**Environment:** Environment sidebar.

**Code Editor:**

```
lambda_function
import json
def lambda_handler(event, context):
    return "API is updated"
```

Lambda > Functions > Create function

### 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.  
**Delete-API**  
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 3.12**

Architecture [Info](#)  
Choose the execution environment 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

Execution role  
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console [Edit](#).

Create a new role with basic Lambda permissions  
 Use an existing role  
 Create a new role from AWS policy templates

Existing role  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
**Lambda**  
View the Lambda role [Edit](#) on the IAM console.

#### Advanced settings

[Cancel](#) [Create function](#)

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

aWS Services Search [Alt+S]

CloudShell Feedback

Successfully created the function Delete-API. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > Delete-API

### Delete-API

Function overview [Info](#)

[Diagram](#) [Template](#)

**Delete-API**  
Layers (0)

+ Add trigger + Add destination

Throttle Copy ARN Actions ▾

Description  
-

Last modified  
9 seconds ago

Function ARN  
**arn:aws:lambda:us-east-1:462972487428:function:Delete-API**

Function URL [Info](#)  
-

Export to Application Composer Download ▾

Code Test Monitor Configuration Aliases Versions

Code source [Info](#)

File Edit Find View Go Tools Window **Test** Deploy Upload from ▾

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Screenshot of the AWS Lambda console showing the code editor for a function named "Delete-API".

The code source is:

```
import json
def lambda_handler(event, context):
    return "API deleted"
```

Function URL: [Info](#)

Code | Test | Monitor | Configuration | Aliases | Versions

Code source | Info

File Edit Find View Go Tools Window Test Deploy Changes not deployed

lambda\_function Environment Var

lambda\_function.py

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the AWS Lambda console showing the "Create function" wizard.

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: Post-API

Runtime: Python 3.12

Architecture: x86\_64

Permissions

Execution role: LabRole

Advanced settings

Create function

Successfully created the function Post-API. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > Post-API

## Post-API

Function overview Info

**Diagram** Template

Post-API  
Layers (0)

+ Add trigger Add destination

Description  
Last modified 9 seconds ago  
Function ARN arn:aws:lambda:us-east-1:462972487428:function:Post-API  
Function URL Info

Code Test Monitor Configuration Aliases Versions

Code source Info

https://us-east-1.console.aws.amazon.com/console/home?region=us-eas... Upload from

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

This screenshot shows the AWS Lambda console interface. At the top, a green banner indicates the function 'Post-API' has been successfully created. Below this, the 'Function overview' tab is selected, showing a diagram of the function with one layer named 'Post-API'. There are buttons to add triggers and destinations. To the right, there's a section for the function ARN and URL. Below the overview, the 'Code' tab is selected, showing the code editor with the Python file 'lambda\_function.py' containing the following code:

```
1 import json
2
3 def lambda_handler(event, context):
4     print(event)
5     text = "I love my {}".format(event['country'])
6     return text
```

Code Test Monitor Configuration Aliases Versions

Code source Info

File Edit Find View Go Tools Window Test Deploy Upload from

lambda\_function Environment Variables

```
1 import json
2
3 def lambda_handler(event, context):
4     print(event)
5     text = "I love my {}".format(event['country'])
6     return text
```

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

This screenshot shows the AWS Lambda code editor. The 'Code source' tab is selected. The code editor displays the Python file 'lambda\_function.py' with the same code as shown in the previous screenshot. On the left, there's a sidebar for environment variables. The bottom navigation bar includes 'CloudShell' and 'Feedback'.

## Creating RestAPI

The screenshot shows the 'Create REST API' wizard in the AWS API Gateway console. The 'API details' section is open, showing four options: 'New API' (selected), 'Clone existing API', 'Import API', and 'Example API'. The 'API name' field contains 'To-do-List'. The 'Description - optional' field is empty. The 'API endpoint type' section indicates 'Regional' as the selected option. At the bottom, there are links for 'cloudShell', 'Feedback', and copyright information.

## Creating Resource

The screenshot shows the 'Resources' page in the AWS API Gateway console. A success message 'Successfully created resource '/demoapi'' is displayed at the top. The left sidebar shows the 'API: To -do -List' section with 'Resources' selected. The main area displays a single resource entry for '/demoapi'. The 'Resource details' panel shows the path '/demoapi' and resource ID 'nfu6pk'. The 'Methods (0)' panel indicates 'No methods defined.' At the bottom, there are links for 'cloudShell', 'Feedback', and copyright information.

## Creating Get Method

Screenshot of the AWS API Gateway 'Create method' interface showing a successful creation message and configuration options for a GET method.

**Method details**

Method type: **GET**

Integration type:

- Lambda function** (selected): Integrate your API with a Lambda function.  
Icon: Lambda symbol.
- HTTP**: Integrate with an existing HTTP endpoint.  
Icon: HTTP symbol.
- Mock**: Generate a response based on API Gateway mappings and transformations.  
Icon: Mock symbol.

**Lambda proxy integration**: Send the request to your Lambda function as a structured event.

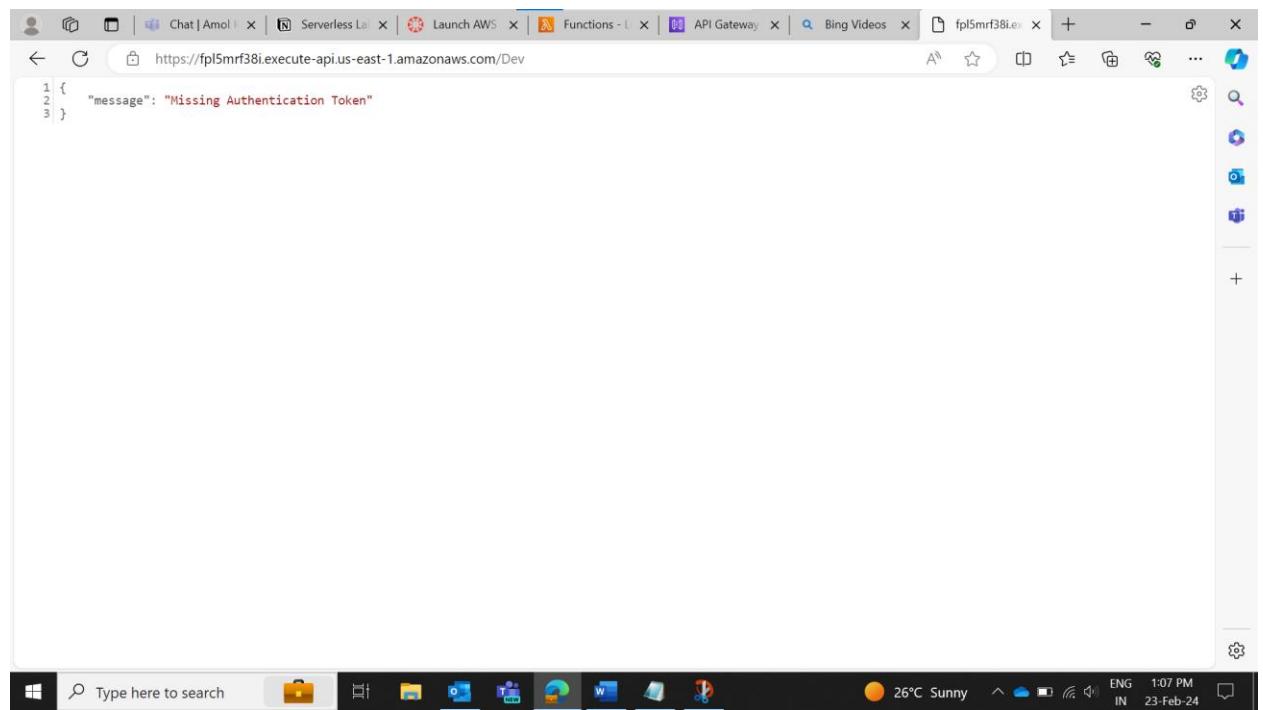
**Lambda function**: Provide the Lambda function name or alias. You can also provide an ARN from another account.  
Region: us-east-1  
Search bar: arn:aws:lambda:us-east-1:462972487428:function:Get-  
Cancel | **Create method**

## API Gateway add in lambda function after creating method in resource

## Creating Stage

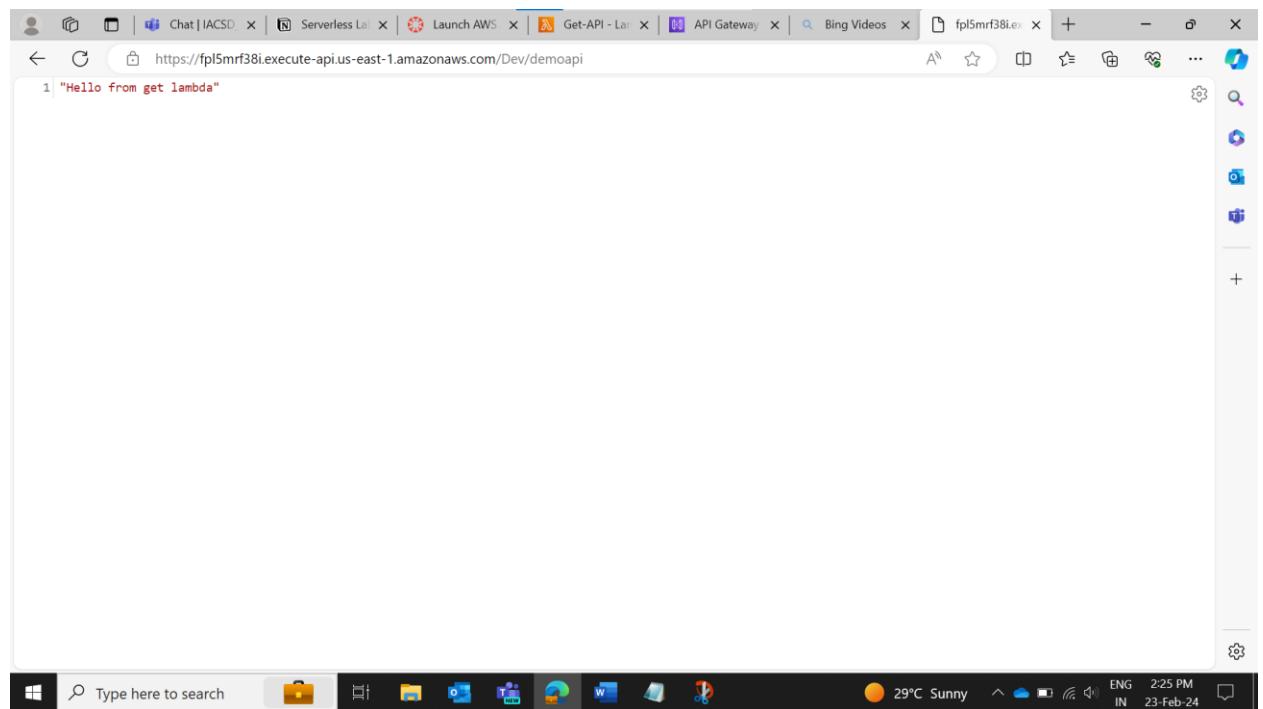
The screenshot shows the AWS API Gateway Stages page for the 'To -do -List' API. The left sidebar shows navigation options like APIs, Resources, Stages, and more. The main area displays the 'Stages' section with a single stage named 'Dev'. The 'Stage details' tab is selected, showing information such as Stage name (Dev), Cache cluster (Inactive), Default method-level caching (Inactive), Invoke URL (<https://fp15mrf38i.execute-api.us-east-1.amazonaws.com/Dev>), and Active deployment (7v5rof on February 23, 2024, 13:00 (UTC+05:30)). There are tabs for Logs and tracing, CloudWatch logs, Detailed metrics, and X-Ray tracing, all currently inactive.

The screenshot shows the same AWS API Gateway Stages page after a deployment. A green success message at the top states 'Successfully created deployment for To -do -List. This deployment is active for Dev.' The stage details for 'Dev' remain the same. A tooltip 'Copied' appears over the Invoke URL field, which now contains the copied URL: <https://fp15mrf38i.execute-api.us-east-1.amazonaws.com/Dev>. The rest of the interface is identical to the first screenshot.



```
1 {  
2     "message": "Missing Authentication Token"  
3 }
```

Adding Resource name at last to see output which I have write in lambda function



```
1 "Hello from get lambda"
```

## Creating Post Method

Screenshot of the AWS API Gateway 'Create method' interface.

The navigation bar shows: Services > APIs > Resources - To -do -List (fp15mrif38i) > Create method.

### Create method

#### Method details

Method type: POST

Integration type:

- Lambda function: Integrate your API with a Lambda function. (Selected)
- 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.

Region: us-east-1    ARN: arn:aws:lambda:us-east-1:462972487428:function:Post

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.

Buttons: Cancel, Create method

Screenshot of the AWS Lambda Functions console showing the "Post-API" function.

**Function overview**

**Diagram** (selected) **Template**

**Post-API** function diagram:

```
graph TD; PostAPI[Post-API] --- Layers[Layers (0)]; PostAPI --- APIGateway[API Gateway (2)];
```

**Description**

Last modified 17 minutes ago

Function ARN arn:aws:lambda:us-east-1:462972487428:function:Post-API

Function URL [Info](#)

**Code** **Test** **Monitor** **Configuration** **Aliases** **Versions**

**Code source** [Info](#)

**Upload from**

Screenshot of the AWS API Gateway console showing the "demoapi" resource.

**API Gateway**

**Resources** (selected)

APIs  
Custom domain names  
VPC links

API: To - do - List

Resources  
Stages  
Authorizers  
Gateway responses  
Models  
Resource policy  
Documentation  
Dashboard  
API settings

Usage plans  
API keys  
Client certificates  
Settings

**Create resource**

**/**

**/demoapi**

**DELETE**  
**GET**  
**POST** (selected)  
**PUT**

**Client certificate**

No client certificates have been generated.

**Request body**

1 {"Country": "India"}

**Test**

AWS Services Search [Alt+S] N. Virginia vocabs/user3011557@amol.kanchar@dctinc.com @ 4629-7248-7428

**API Gateway**

APIs Custom domain names VPC links

▼ API: To -do -List

**Resources**

Stages Authorizers Gateway responses Models Resource policy Documentation Dashboard API settings

Usage plans API keys Client certificates Settings

Create resource

Test

**POST** / /demoapi GET

**/ - POST method test results**

| Request | Latency | Status |
|---------|---------|--------|
| /       | 228     | 200    |

Response body  
"I love my India"

Response headers

```
{  
    "Content-Type": "application/json",  
    "X-Amzn-Trace-Id": "Root=1-65d85e83-dc8976bd92f7964e1dc50c59;Parent=634e39c1a9de175b;Sampled=0;lineage=94d6194f:0"  
}
```

Log

```
Execution log for request 1dbf82ba-e98d-465e-8c15-c579e20fdf8e  
Fri Feb 23 08:59:47 UTC 2024 : Starting execution for request: 1dbf82ba-e98d-465e-8c15-c579e20fdf8e  
Fri Feb 23 08:59:47 UTC 2024 : HTTP Method: POST, Resource Path: /  
Fri Feb 23 08:59:47 UTC 2024 : Method request path: {}  
Fri Feb 23 08:59:47 UTC 2024 : Method request query string: {}  
Fri Feb 23 08:59:47 UTC 2024 : Method request headers: {}  
Fri Feb 23 08:59:47 UTC 2024 : Method request body before transformations: {"Country":"India"}  
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
```

## Creating Put Method

Screenshot of the AWS API Gateway 'Create method' wizard.

The 'Method type' dropdown is set to 'PUT'.

The 'Integration type' section shows the following options:

- Lambda function** (selected): Integrate your API with a Lambda function. It includes a Lambda icon.
- HTTP**: Integrate with an existing HTTP endpoint. It includes an HTTP icon.
- Mock**: Generate a response based on API Gateway mappings and transformations. It includes a Mock icon.
- AWS service**: Integrate with an AWS Service. It includes an AWS icon.
- VPC link**: Integrate with a resource that isn't accessible over the public internet. It includes a VPC icon.

The 'Lambda proxy integration' section indicates that requests will be sent to a Lambda function as structured events.

The 'Lambda function' field shows 'us-east-1' selected, and the ARN 'arn:aws:lambda:us-east-1:462972487428:function:Put-' is entered in the search bar.

A note states: "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." A 'Default timeout' note specifies a 29-second default.

At the bottom right are 'Cancel' and 'Create method' buttons, with 'Create method' being highlighted.

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with 'Services' (selected), a search bar, and a location indicator 'N. Virginia'. The main title is 'Put-API' under the 'Functions' section. On the right, there are buttons for 'Throttle', 'Copy ARN', and 'Actions'. Below the title, a 'Function overview' section is expanded, showing a diagram where 'Put-API' (an icon with an orange A) is connected to 'Layers (0)' and 'API Gateway' (an icon with a purple H). There are buttons for '+ Add destination' and '+ Add trigger'. To the right of the diagram, there are sections for 'Description' (empty), 'Last modified' (15 minutes ago), 'Function ARN' (arn:aws:lambda:us-east-1:462972487428:function:Put-API), and 'Function URL' (Info). At the bottom, tabs for 'Code' (selected), 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions' are visible. The 'Code source' tab is also present at the bottom.

The screenshot shows the AWS API Gateway console. On the left, the sidebar includes sections for APIs, Custom domain names, VPC links, API To-Do List, Resources (selected), Stages, Authorizers, Gateway responses, Models, Resource policy, Documentation, Dashboard, API settings, Usage plans, API keys, Client certificates, and Settings. The main area displays a tree view of resources under the /demoapi endpoint, with the PUT method selected. A modal window titled "/demoapi - PUT method test results" is open, showing the request path (/demoapi), latency (301 ms), status (200), response body ("API is Updated"), and response headers. The response body is highlighted with a red underline. The log section shows the execution log for the request, including the start of execution, method details, request headers, and the full request body. The request body contains JSON with fields like Content-Type and X-Amzn-Trace-Id. The log continues with endpoint requests and the final sending of the request to the Lambda function.

```
Request
/demoapi

Latency
301

Status
200

Response body
"API is Updated"

Response headers
{
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-65d802f-b57e0ff9f67ae56e0da532ed;Parent=23715d7cffbf4f32d;Sampled=0;Lineage=646d60fa:0"
}

Log
Execution log for request f0f08742e-4e1a-4c15-a6ba-e0f5318a08f7
Fri Feb 23 09:15:27 UTC 2024 : Starting execution for request: f0f08742e-4e1a-4c15-a6ba-e0f5318a08f7
Fri Feb 23 09:15:27 UTC 2024 : HTTP Method: PUT, Resource Path: /demoapi
Fri Feb 23 09:15:27 UTC 2024 : Method request path: {}
Fri Feb 23 09:15:27 UTC 2024 : Method request query string: {}
Fri Feb 23 09:15:27 UTC 2024 : Method request headers: {}
Fri Feb 23 09:15:27 UTC 2024 : Method request body before transformations:
Fri Feb 23 09:15:27 UTC 2024 : Endpoint request URL: https://lambda.us-east-1.amazonaws.com:2015-03-31/functions/arn:aws:lambda:us-east-1:462972487428:function:Put-API/invocations
Fri Feb 23 09:15:27 UTC 2024 : Endpoint request headers: {x-amz-lambda-integration-tag=f0f08742e-4e1a-4c15-a6ba-e0f5318a08f7, Authorization="*****", x-amz-apigateway-api-id=f0f5318a08f7}
*****
*****bcc498, X-Amz-Date=20240223T0912Z, x-amz-apigateway-api-id=f0f5318a08f7, X-Amz-Source-Arn=arn:aws:execute-api:us-east-1:462972487428:f0f5318a08f7:stage(INVOKE-STAGE_PUT/demoapi), Accept=application/json, User-Agent=AmazonAPIGateway_f0f5318a08f7, X-Amz-Security-Token=IQoJb3IjZ2lUVXJEl1//wIAcXv2LWhc3QHJMGEQCIKvSmrF5ZtLQBXOsUzIqLQV0nOln0mInGAl1/yhDm1AbGhuIVB62jSayBQhxEAdA0DMEHf1yDUfJNfY1MCfHBDsuB5Y5Qq2VPBk0fBT0fMPnnfHT1vMxkhtpTeIRf2uwhfPbALYqrKm8J9NmWu1T11jYExX+bCwVwkgqvny4PxMt7bxjf6n3WaqqCFY-q9x0QJUSzLhoh1d1S12u19mVNCktEeg7+OpvrxauXuH2uPiVoAs9CVvSV [TRUNCATED]
Fri Feb 23 09:15:27 UTC 2024 : Endpoint request body after transformations:
Fri Feb 23 09:15:27 UTC 2024 : Sending request to https://lambda.us-east-1.amazonaws.com:2015-03-31/functions/arn:aws:lambda:us-
```

## Creating Delete Method

Screenshot of the AWS API Gateway 'Create method' interface showing the successful creation of a 'PUT' method.

**Method details**

**Method type:** DELETE

**Integration type:** Lambda function (selected)

- Lambda function: Integrate your API with a Lambda function. (Selected)
- HTTP: Integrate with an existing HTTP endpoint.
- Mock: Generate a response based on API Gateway mappings and transformations.

**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 ▾ Q arn:aws:lambda:us-east-1:462972487428:function:Delete X

**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.

Cancel Create method

Screenshot of the AWS Lambda Functions console showing the "Delete-API" function.

**Function overview**

- Description:** -
- Last modified:** 13 minutes ago
- Function ARN:** arn:aws:lambda:us-east-1:462972487428:function:Delete-API
- Function URL:** Info

**Diagram**

```
graph TD; DeleteAPI[Delete-API] --- APIGateway[API Gateway]; DeleteAPI --- Layers[Layers (0)];
```

**Code source**

Upload from ▾

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the AWS API Gateway console showing the test results for the DELETE method of the "/demoapi" resource.

**Test results for /demoapi - DELETE method**

| Request  | Latency | Status |
|----------|---------|--------|
| /demoapi | 289     | 200    |

**Response body:** "API deleted"

**Response headers:**

```
{ "Content-Type": "application/json", "X-Amzn-Trace-Id": "Root=1-65d862c4-e280884218a2b92e04eb82a;Parent=250ea65f3cbd982a;Sampled=0;lineage=5567a164:0" }
```

**Log:**

```
Execution log for request 9bad4837-22d2-4535-9b94-bbe4a7724cad
Fri Feb 23 09:17:56 UTC 2024 : Starting execution for request: 9bad4837-22d2-4535-9b94-bbe4a7724cad
Fri Feb 23 09:17:56 UTC 2024 : HTTP Method: DELETE, Resource Path: /demoapi
Fri Feb 23 09:17:56 UTC 2024 : Method request path: {}
Fri Feb 23 09:17:56 UTC 2024 : Method request query string: {}
Fri Feb 23 09:17:56 UTC 2024 : Method request headers: {}
Fri Feb 23 09:17:56 UTC 2024 : Method request body before transformations:
Fri Feb 23 09:17:56 UTC 2024 : Endpoint request URL: https://lambda.us-east-1.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-east-1:462972487428:function:Delete-API/invocations
Fri Feb 23 09:17:56 UTC 2024 : Endpoint request headers: {x-amzn-lambda-integration-
```

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Q3)

### Creating three S3 buckets

The screenshot shows the AWS S3 'Create bucket' wizard. The top navigation bar includes the AWS logo, Services, a search bar, and a keyboard shortcut [Alt+S]. The breadcrumb trail shows 'Amazon S3 > Buckets > Create bucket'. The main title is 'Create bucket' with an 'Info' link. A note states: 'Buckets are containers for data stored in S3. [Learn more](#)'.

**General configuration**

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

Bucket type:  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](#) textfile-buck  
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.  
[Choose bucket](#)  
Format: s3://bucket/prefix

**Object Ownership** [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)  
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled  
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership: Bucket owner enforced

aws | Services | Search [Alt+S]

Bucket owner enforced

### Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will 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.

**A** 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 and application failures. [Learn more](#)

Bucket Versioning

Disable

Enable

### Tags - optional (0)

You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

CloudShell Feedback

AWS Services Search [Alt+S] Global v vocabs/user3011557=amol.kanchar@dctinc.com @ 4629-7248-74

Amazon S3 > Buckets

▼ Account snapshot

Last updated: Feb 20, 2024 by Storage Lens. Metrics are generated every 24 hours. Metrics don't include directory buckets. [Learn more](#)

Total storage Object count Average object size You can enable advanced metrics in the "default-account-dashboard" configuration.

31.9 MB 14 2.3 MB

View Storage Lens dashboard

General purpose buckets Directory buckets

General purpose buckets (4) [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

Find buckets by name

C Copy ARN Empty Delete Create bucket

| Name             | AWS Region                      | Access                | Creation date                           |
|------------------|---------------------------------|-----------------------|---|
| textfile-buck    | US East (N. Virginia) us-east-1 | Objects can be public | February 23, 2024, 15:15:26 (UTC+05:30) |
| replicated-buck  | US East (N. Virginia) us-east-1 | Objects can be public | February 23, 2024, 15:19:30 (UTC+05:30) |
| myhelpful-buck   | US East (N. Virginia) us-east-1 | ⚠️ Public             | February 22, 2024, 19:34:10 (UTC+05:30) |
| capitalized-buck | US East (N. Virginia) us-east-1 | Objects can be public | February 23, 2024, 15:20:11 (UTC+05:30) |

AWS Services Search [Alt+S]

Amazon S3 > Buckets > [textfile-buck](#) > Create folder

## Create folder [Info](#)

Use folders to group objects in buckets. When you create a folder, S3 creates an object using the name that you specify followed by a slash (/). This object then appears as folder on the console. [Learn more](#)

Your bucket policy might block folder creation

If your bucket policy prevents uploading objects without specific tags, metadata, or access control list (ACL) grantees, you will not be able to create a folder using this configuration. Instead, you can use the [upload configuration](#) to upload an empty folder and specify the appropriate settings.

**Folder**

Folder name  /

Folder names can't contain "/". See rules for naming

**Server-side encryption [Info](#)**

Server-side encryption protects data at rest.

The following encryption settings apply only to the folder object and not to sub-folder objects.

Server-side encryption

Do not specify an encryption key  
The bucket settings for default encryption are used to encrypt the folder object when storing it in Amazon S3.

Specify an encryption key  
The specified encryption key is used to encrypt the folder object before storing it in Amazon S3.

If your bucket policy requires objects to be encrypted with a specific encryption key, you must specify the same encryption key when you create a folder. Otherwise, folder creation will fail.

Cancel **Create folder**

Successfully created folder "Files".

Amazon S3 > Buckets > textfile-buck

### textfile-buck [Info](#)

Objects [\(1\) Info](#)

Actions [Actions](#) [Create folder](#) [Upload](#)

Find objects by prefix  Show versions

| Name   | Type   | Last modified | Size | Storage class |
|--------|--------|---------------|------|---------------|
| Files/ | Folder | -             | -    | -             |

## Creating Lambda Function

Lambda > Functions > Create function

### 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.

Runtime [Info](#)  
Choose the language to use for writing 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.

Execution role  
Choose a role that defines the permissions of your functions. To create a custom role, go to the IAM console [\[Link\]](#).

Create a new role with basic Lambda permissions  
 Use an existing role  
 Create a new role from AWS policy templates

Existing role  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
  
View the Lambda role [\[Link\]](#) on the IAM console.

**Advanced settings**

[Cancel](#) [Create function](#)

aws | Services | Search [Alt+S]

Lambda > Add trigger

## Add trigger

**Trigger configuration** [Info](#)

**S3** aws asynchronous storage ▾

**Bucket**  
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.

[X](#) [G](#)

Bucket region: us-east-1

**Event types**  
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

[X](#)

**Prefix - optional**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

**Suffix - optional**  
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

**Recursive invocation**  
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

[Cancel](#) [Add](#)

## Trigger Added Successfully

The screenshot shows the AWS Lambda Functions interface. In the top navigation bar, 'Lambda' is selected under 'Services'. Below it, 'Functions' and 'TextFileFunction' are shown. On the right side of the top bar, there are buttons for 'Throttle', 'Copy ARN', and 'Actions'. A green success message box states: 'The trigger textfile-buck was successfully added to function TextFileFunction. The function is now receiving events from the trigger.' Below this message, the 'Function overview' section is visible, featuring a 'Diagram' tab which shows a flowchart with 'TextFileFunction' at the top, an 'S3' box below it, and a 'Layers' section. To the right of the diagram, there are buttons for 'Export to Application Composer' and 'Download'. The 'Configuration' tab is currently selected, showing the 'Triggers' section. Under 'Triggers', there is one entry: 'S3: textfile-buck' with the ARN 'arn:aws:s3:::textfile-buck'. There are also buttons for 'Find triggers', 'Edit', 'Delete', and 'Add trigger'.

The screenshot shows the AWS Lambda Code Editor interface. At the top, the menu includes 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test', 'Deploy', and 'Changes not deployed'. The status bar indicates 'N. Virginia' and the user 'vocabs/user3011557=amol.kanchar@dctinc.com @ 4629-7248-7428'. The main area displays a code editor with three tabs: 'Replicating.py', 'Capitalized.py', and 'Environment'. The 'Replicating.py' tab contains the following Python code:

```
1 import boto3
2
3 s3 = boto3.client('s3')
4
5 def lambda_handler(event, context):
6     source_bucket = event['Records'][0]['s3']['bucket']['name']
7     source_key = event['Records'][0]['s3']['object']['key']
8
9     destination_bucket = 'replicated-buck'
10
11     try:
12         copy_source = {'Bucket': source_bucket, 'Key': source_key}
13         s3.copy_object(CopySource=copy_source, Bucket=destination_bucket, Key=source_key)
14         print(f"File '{source_key}' replicated successfully to '{destination_bucket}'")
15         return {
16             'statusCode': 200,
17             'body': f"File '{source_key}' replicated successfully to '{destination_bucket}'"
18         }
19     except Exception as e:
20         print(f"Error replicating file '{source_key}': {e}")
21         return {
22             'statusCode': 500,
23             'body': f"Error replicating file '{source_key}': {e}"
24         }
25
26
```

The status bar at the bottom right shows '10:38 Python Spaces: 4'.

The screenshot shows the AWS Lambda code editor interface. The top navigation bar includes the AWS logo, Services, a search bar, and account information: vocabs/user3011557=amol.kanchan@dctinc.com @ 4629-7248-7428. The main area has tabs for 'Code source' and 'Info'. Below the tabs is a toolbar with File, Edit, Find, View, Go, Tools, Window, Test (selected), Deploy, and other icons. A sidebar on the left shows the file structure: Text\_Updating / Capitalized.py (selected) and Replicating.py. The code editor displays the following Python script:

```
1 import boto3
2
3 s3 = boto3.client('s3')
4
5 def lambda_handler(event, context):
6     source_bucket = event['Records'][0]['s3']['bucket']['name']
7     source_key = event['Records'][0]['s3']['object']['key']
8     source_key_uppercase = source_key.upper() # Convert source_key to uppercase
9
10    destination_bucket = 'capitalized-buck'
11
12    try:
13        response = s3.get_object(Bucket=source_bucket, Key=source_key)
14        file_content = response['Body'].read().decode('utf-8')
15
16        capitalized_content = file_content.upper()
17
18        destination_key = source_key_uppercase
19        s3.put_object(Bucket=destination_bucket, Key=destination_key, Body=capitalized_content.encode('utf-8'))
20
21        print(f"File '{source_key}' replicated successfully to '{destination_bucket}' with capitalized contents")
22        return {
23            'statusCode': 200,
24            'body': f"File '{source_key}' replicated successfully to '{destination_bucket}' with capitalized contents"
25        }
26    except Exception as e:
27        print(f"Error replicating file '{source_key}': {e}")
28        return {
29            'statusCode': 500,
30            'body': f"Error replicating file '{source_key}': {e}"
31        }
32
33
34
```

The status bar at the bottom right indicates 20:9 Python Spaces: 4.

The screenshot shows the 'Edit runtime settings' dialog for a function named 'TextFileFunction'. The top navigation bar includes the AWS logo, Services, a search bar, and account information: N. Virginia. The path in the left sidebar is Lambda > Functions > TextFileFunction > Edit runtime settings. The main form has sections for Runtime, Handler, Architecture, and a Save/Cancel button.

**Runtime settings**

**Runtime**  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.12

**Handler** replicating.copying\_to\_newbuck

**Architecture** x86\_64  
 x86\_64  arm64

Cancel

☰

## Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

| Files and folders (1 Total, 17.0 B)                   |        | Remove | Add files | Add folder |
|---|--------|--------|-----------|------------|
| All files and folders in this table will be uploaded. |        |        |           |            |
| <input type="text"/> Find by name                     |        | <      | 1         | >          |
| <input type="checkbox"/> Name                         | Folder | Type   |           |            |
| <input type="checkbox"/> repli.txt                    | -      | text   | < ━━━━ >  |            |

### Destination Info

Destination  
[s3://textfile-buck/Files/](#)

▶ **Destination details**  
Bucket settings that impact new objects stored in the specified destination.

▶ **Permissions**  
Grant public access and access to other AWS accounts.

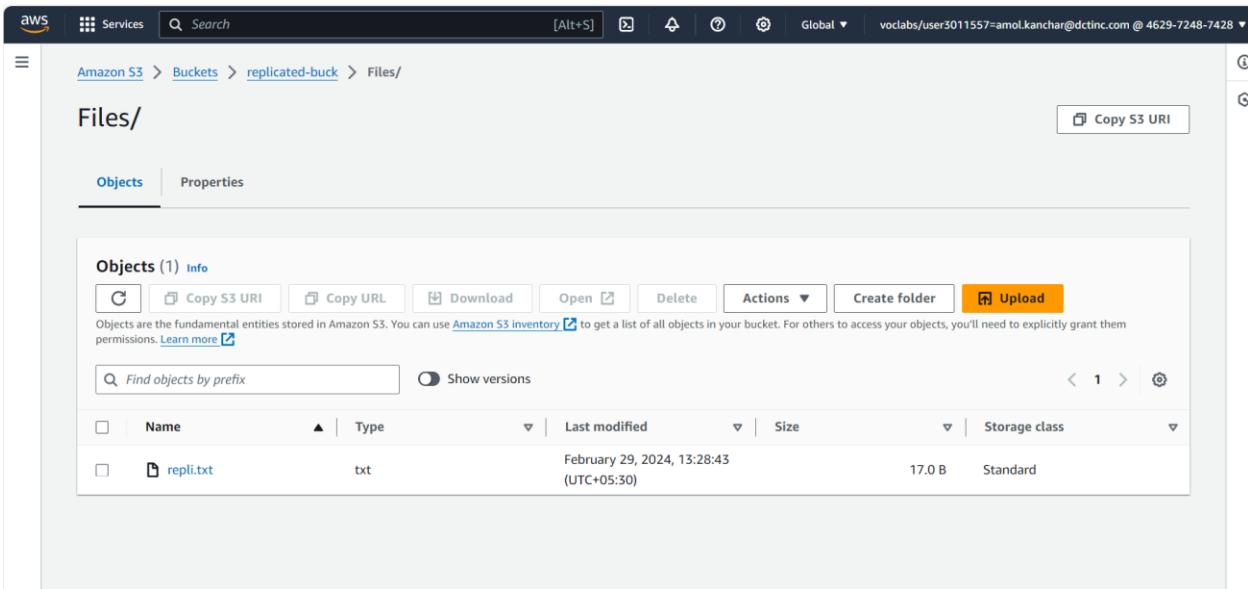
▶ **Properties**  
Specify storage class, encryption settings, tags, and more.

**Cancel** **Upload**

## CloudWatch logs

The screenshot shows the AWS CloudWatch service interface. The left sidebar is collapsed, and the main area displays the details for the log group `/aws/lambda/TextFileFunction`. The top navigation bar includes the AWS logo, services dropdown, search bar, and region information (N. Virginia). Below the navigation is a breadcrumb trail: CloudWatch > Log groups > `/aws/lambda/TextFileFunction`. The main content area has a title `/aws/lambda/TextFileFunction` and a "Log group details" section. This section contains tabs for Log class (Info), Standard, Metric filters (0), Subscription filters (0), Contributor Insights rules (-), KMS key ID (-), Anomaly detection (Configure), Data protection (-), and Sensitive data count (-). Below this is a "Log streams" tab bar with options: Settings, Tags, Anomaly detection, Metric filters, Subscription filters, Contributor Insights, and Data protection. The "Log streams" tab is selected, showing a list with one item: "Log streams (1)". A search bar at the top of the list allows filtering by prefix. To the right of the search bar are buttons for Create log stream, Delete, and Search all log streams. The list item shows a checkbox labeled "Log stream", a timestamp "Last event time" (2024-02-29 13:07:04 (UTC+05:30)), and a timestamp "2024/02/29[\$LATEST]2065674b2656450289201eb0bcc16ca5".

## Final Output



The screenshot shows the AWS S3 console interface. At the top, the navigation bar includes the AWS logo, Services, a search bar, and user information: 'voclabs/user3011557=amol.kanchar@dctinc.com @ 4629-7248-7428'. Below the navigation bar, the breadcrumb trail shows 'Amazon S3 > Buckets > replicated-buck > Files/'. The main area is titled 'Files' and contains a table of objects. The table has columns: Name, Type, Last modified, Size, and Storage class. One object is listed: 'repli.txt' (Type: txt, Last modified: February 29, 2024, 13:28:43 (UTC+05:30), Size: 17.0 B, Storage class: Standard). Action buttons at the top of the table include Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, and Upload.

| Name      | Type | Last modified                              | Size   | Storage class |
|-----------|------|--|--------|---------------|
| repli.txt | txt  | February 29, 2024, 13:28:43<br>(UTC+05:30) | 17.0 B | Standard      |

Now editing runtime before triggering :-

The screenshot shows the 'Edit runtime settings' page for a Lambda function named 'TextFileFunction'. The 'Runtime' section is set to 'Python 3.12'. The 'Handler' field contains the value 'capitalized.uppercasefile\_to\_newbuck'. The 'Architecture' section shows 'x86\_64' selected. A note at the bottom states: 'You can change either the function's runtime or the instruction set architecture in one update. To update both, you must repeat the update process.' At the bottom right are 'Cancel' and 'Save' buttons.

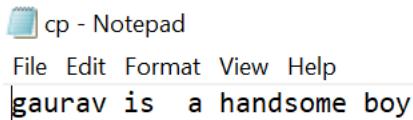
## Uploading file in s3:-

The screenshot shows the 'Upload succeeded' status page for an upload to 's3://replicated-buck/Files/'. The summary table shows 1 file uploaded successfully (25.0 B) and 0 files failed (0 B). The 'Files and folders' section lists 1 file: 'cp.txt' (text/plain, 25.0 B, Succeeded). At the bottom are links for 'CloudShell', 'Feedback', and 'Cookie preferences'.

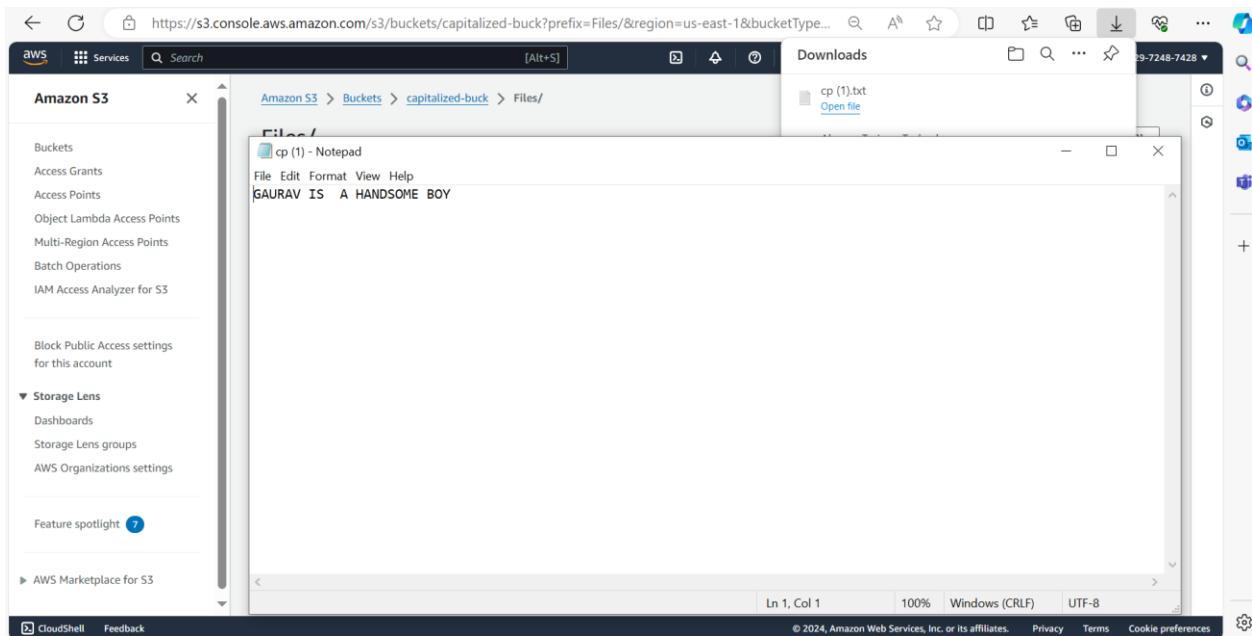
| Destination                 | Succeeded                | Failed            |
|-----------------------------|--------------------------|-------------------|
| s3://replicated-buck/Files/ | 1 file, 25.0 B (100.00%) | 0 files, 0 B (0%) |

The screenshot shows the AWS CloudWatch Log Events interface. The left sidebar is titled "CloudWatch" and includes sections for Favorites and recents, Dashboards, Alarms, Logs (selected), Log groups, Log Anomalies, Live Tail, Logs Insights, Metrics, X-Ray traces, Events, Application Signals, Network monitoring, and Insights. Under Logs, there are links for Settings, Getting Started, and What's new. The main content area shows a breadcrumb trail: CloudWatch > Log groups > /aws/lambda/TextFileFunction > 2024/02/29/[\$.LATEST]8dcb431ab5c94561817585ff4889bc0. Below this is a section titled "Log events" with a filter bar containing a search input, time range (Clear, 1m, 30m, 1h, 12h, Custom, Local timezone), and display options. The log entries table has columns for "Timestamp" and "Message". The first entry is "No older events at this moment. Retry". Subsequent entries show the Lambda function starting, processing records, replicating files, and reporting completion. The last entry is "No newer events at this moment. Auto retry paused. Resume".

File before:-



## File After:-



## Q1) Building a Serverless Web Application

### Creating S3 bucket

The screenshot shows the AWS S3 'Create bucket' wizard. It consists of two main sections: 'General configuration' and 'Object Ownership'.

**General configuration:**

- AWS Region:** US East (N. Virginia) us-east-1
- Bucket type:** General purpose (selected)
- Bucket name:** servelessapp-buck
- Copy settings from existing bucket - optional:** Only the bucket settings in the following configuration are copied. A 'Choose bucket' button is available.
- Format:** s3://bucket/prefix

**Object Ownership:**

- ACLs disabled (recommended):** All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ACLs enabled (selected):** Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

**Warning:** We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

**Object Ownership:**

- Bucket owner preferred (selected):** If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.
- Object writer:** The object writer remains the object owner.

### Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will 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 and application failures. [Learn more](#)

#### Bucket Versioning

- Disable  
 Enable

### Tags - optional (0)

You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

No tags associated with this bucket.

☰

### 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 and application failures. [Learn more](#)

**Bucket Versioning**

Disable  
 Enable

### Tags - optional (0)

You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

No tags associated with this bucket.

[Add tag](#)

### Default encryption Info

Server-side encryption is automatically applied to new objects stored in this bucket.

**Encryption type** [Info](#)

Server-side encryption with Amazon S3 managed keys (SSE-S3)  
 Server-side encryption with AWS Key Management Service keys (SSE-KMS)  
 Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)  
Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing on the Storage tab of the [Amazon S3 pricing page](#)

**Bucket Key**  
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

Disable  
 Enable

### ► Advanced settings

ⓘ After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

[Cancel](#) [Create bucket](#)

## Adding policy get object

AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [key concepts in Using AWS Identity and Access Management](#). Here are sample policies.

**Step 1: Select Policy Type**

A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket Policy](#), an [SNS Topic Policy](#), a [VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

Select Type of Policy [S3 Bucket Policy](#)

**Step 2: Add Statement(s)**

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect  Allow  Deny

Principal

AWS Service  Amazon S3  All Services (\*)

Actions  --Select Actions--  All Actions (\*)

Amazon Resource Name (ARN)

Add Conditions (Optional)

[Add Statement](#)

You added the following statements. Click the button below to Generate a policy.

| Principal(s) | Effect | Action       | Resource                         | Conditions |
|--------------|--------|--------------|----------------------------------|------------|
| *            | Allow  | s3:GetObject | arn:aws:s3:::servelessapp-buck/* | None       |

**Step 3: Generate Policy**

A policy is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

[Generate Policy](#) [Start Over](#)

aWS Services Search [Alt+S] Global vodcasts/user3011557=amol.kanchar@dctinc.com @ 4629-7248-7428 ▾

Successfully edited bucket policy.

**Bucket policy**

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

[Edit](#) [Delete](#)

```
[{"Version": "2012-10-17", "Id": "Policy1709196947095", "Statement": [{"Sid": "Stmt1709196917534", "Effect": "Allow", "Principal": "*", "Action": "s3:GetObject", "Resource": "arn:aws:s3:::servelessapp-buck/*"}]}
```

[Copy](#)

## Enabling Static Website hosting

The screenshot shows the AWS S3 console with the path [Amazon S3](#) > [Buckets](#) > [servelessapp-buck](#) > [Edit static website hosting](#). The main section is titled "Static website hosting" with the sub-instruction "Use this bucket to host a website or redirect requests." Below this, there are two radio button options: "Disable" (unchecked) and "Enable" (checked). Under "Hosting type", there are also two radio button options: "Host a static website" (checked) with the note "Use the bucket endpoint as the web address." and "Redirect requests for an object" (unchecked) with the note "Redirect requests to another bucket or domain." A callout box highlights this note with the text: "For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)." The "Index document" field is set to "contactus.html" and the "Error document - optional" field is set to "error.html". At the bottom, there are links for "CloudShell", "Feedback", and "Cookie preferences".

The screenshot shows the AWS S3 console after saving changes. A green banner at the top says "Successfully edited static website hosting." The configuration page now shows "Disabled" under "Static website hosting". The "Bucket website endpoint" field contains the URL <http://servelessapp-buck.s3-website-us-east-1.amazonaws.com>. Other sections like "Object Lock" and "Requester pays" remain unchanged. The status bar at the bottom includes links for "CloudShell", "Feedback", and "Cookie preferences".

## Creating DynamoDB table

The screenshot shows the AWS DynamoDB 'Create table' interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, a search bar, and a keyboard shortcut '[Alt+S]'. Below the navigation is a breadcrumb trail: 'DynamoDB > Tables > Create table'. The main title 'Create table' is centered above the configuration sections.

**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.  
Input field: Serverlessapp  
Validation: 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.  
Input field: Email ID  
Type: String  
Validation: 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.  
Input field: Enter the sort key name  
Type: String  
Validation: 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.



Services

Search [Alt+S]



## 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 |
|----------------------------|--------------------------|-------------------------|
| Table class                | DynamoDB Standard        | Yes                     |
| 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                     |
| 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.

Add new tag

You can add 50 more tags.

Cancel

Create table

The screenshot shows the AWS DynamoDB service page. On the left, there's a navigation sidebar with links like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Below that is a section for DAX Clusters, Subnet groups, Parameter groups, and Events. The main content area is titled "Tables (1/1) Info". It shows a table with one item: "Serverlessapp" (Active, Email ID (\$), Sort key -, Indexes 0, Deletion protection Off, Read capacity mode Provisioned (\$), Write capacity mode Provisioned (\$), Total size 0 bytes, Table class Standard). At the top right of this table are buttons for Actions, Delete, and Create table. A green banner at the top of the main content area says "The Serverlessapp table was created successfully." The bottom of the screen includes CloudShell, Feedback, and standard footer links.

## Creating Function

The screenshot shows the AWS Lambda "Create function" wizard. The top navigation bar has "Lambda > Functions > Create function". The main heading is "Create function" with a "Info" link. Below it, there are three options: "Author from scratch" (selected, with a note about starting with a Hello World example), "Use a blueprint" (with a note about building a Lambda application from sample code and configuration presets), and "Container image" (with a note about selecting a container image to deploy your function). The "Basic information" section contains fields for "Function name" (set to "ServerlessAppFunction"), "Runtime" (set to "Python 3.12"), and "Architecture" (set to "x86\_64"). The "Permissions" section notes that Lambda will create an execution role with permissions to upload logs to CloudWatch Logs. The "Execution role" section allows defining the permissions of the function's role, with options for creating a new role with basic Lambda permissions or a new role from AWS policy templates. The "Existing role" section lists a role named "LambdaRole" with a note to view it in the IAM console. The "Advanced settings" section is collapsed. At the bottom right are "Cancel" and "Create function" buttons.

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with 'Services' and a search bar. Below it, the function name 'ServerlessAppFunction' is displayed. On the right side, there are buttons for 'Throttle', 'Copy ARN', and 'Actions'. Under the 'Function overview' section, there's a diagram showing a single function node labeled 'ServerlessAppFunction' and a 'Layers' section with '(0)'. Below the diagram are buttons for '+ Add trigger' and '+ Add destination'. To the right, there's a 'Description' section with details like 'Last modified 17 minutes ago', 'Function ARN', and 'Function URL'. At the bottom, tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions' are visible.

**Code source**

```
File Edit Find View Go Tools Window Test Deploy
lambda_function Environment Var ...
lambda_function.py
1 import boto3
2
3 dynamodb = boto3.client('dynamodb')
4
5 def lambda_handler(event, context):
6     first_name = event['first_name']
7     last_name = event['last_name']
8     email_id = event['email_id']
9     message = event['message']
10
11     dynamodb.put_item(
12         TableName='Serverlessapp',
13         Item={
14             'First Name': {'S': first_name},
15             'Last Name': {'S': last_name},
16             'Email ID': {'S': email_id},
17             'Message': {'S': message}
18         }
19     )
20     print(email_id)
21
```

Testing lambda function and it will reflect the values in dynamodb table

The screenshot shows the 'Test' tab of the Lambda function configuration page. A success message 'Executing function: succeeded (logs)' is displayed. Below it, there's a 'Test event' section with a 'Details' link. The 'Test event' section includes buttons for 'Delete', 'Save', and 'Test'. A note says '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.' There are two options for 'Test event action': 'Create new event' (unchecked) and 'Edit saved event' (checked). An 'Event name' dropdown is set to 'wha'. The 'Event JSON' section contains a sample event payload:

```
1 [{ 2   "first_name": "Rinkesh", 3   "last_name": "More", 4   "email_id": "rinkeshmore@gmail.com", 5   "message": "World" 6 }]
```

Like this

The screenshot shows the AWS DynamoDB console. On the left, the navigation pane includes 'Dashboard', 'Tables', 'Explore items' (which is selected), 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations' (with a 'New' button), 'Reserved capacity', 'Settings', and a 'DAX' section with 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area shows a table named 'Serverlessapp' with one item. The table details are as follows:

| Email ID (String)      | First Name | Last Name | Message |
|------------------------|------------|-----------|---------|
| amolk04@gmail.com      | Amol       | Kanchar   | India   |
| rinkeshmore@gmail.c... | Rinkesh    | More      | World   |

Creating API gateway

The screenshot shows the AWS API Gateway console. It displays three options: 'WebSocket API', 'REST API', and 'REST API Private'. The 'REST API' option is highlighted with a red circle. The 'REST API' section contains the following text:

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Works with the following:  
Lambda, HTTP, AWS Services

Import Build

AWS Services Search [Alt+S] N. Virginia ▾

API Gateway > APIs > Create API > Create REST API

## Create REST API

**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.

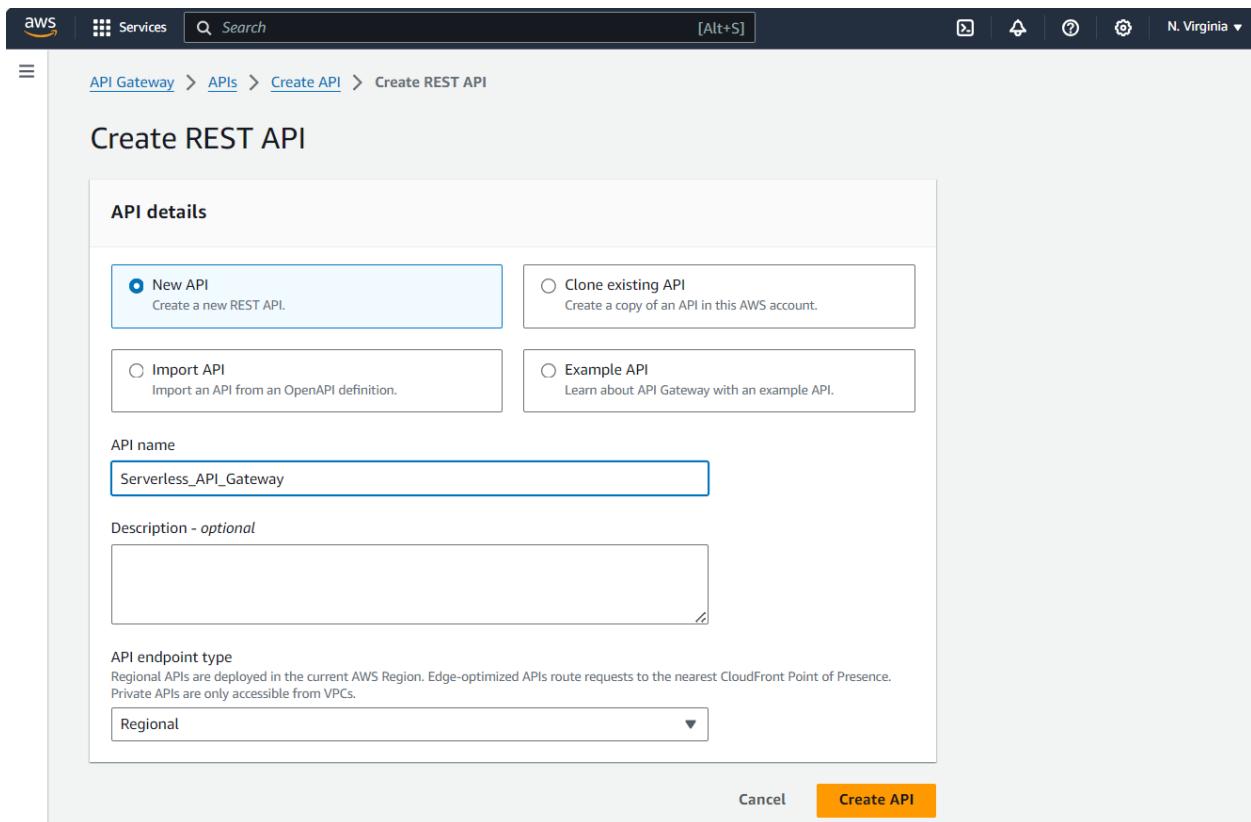
Example API  
Learn about API Gateway with an example API.

API name

Description - *optional*

API endpoint type  
Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence.  
Private APIs are only accessible from VPCs.

Cancel **Create API**



AWS Services Search [Alt+S] N. Virginia ▾

Successfully created REST API 'Serverless\_API\_Gateway (wmsswkxil)'

API Gateway > APIs > Resources - Serverless\_API\_Gateway (wmsswkxil) > Create resource

## Create resource

**Resource details**

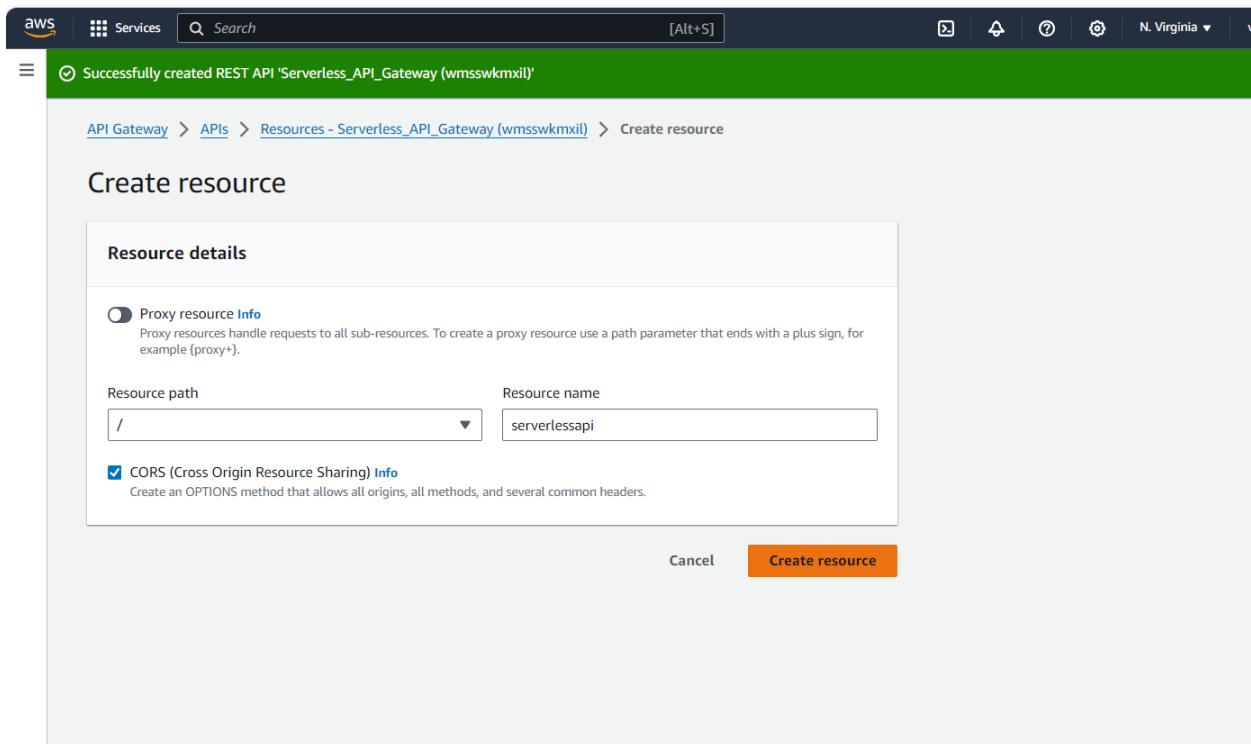
Proxy resource [Info](#)  
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

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**



## Creating Post method

Screenshot of the AWS API Gateway 'Create method' dialog.

The top navigation bar shows 'Services' selected, and the breadcrumb path is 'API Gateway > APIs > Resources - Serverless\_API\_Gateway (wmsswkml) > Create method'. A green success message at the top says 'Successfully created resource '/serverlessapi'.'

**Method details**

**Method type:** POST

**Integration type:**

- Lambda function**: Integrate your API with a Lambda function.  
Icon: Lambda symbol.
- HTTP**: Integrate with an existing HTTP endpoint.  
Icon: HTTP symbol.
- Mock**: Generate a response based on API Gateway mappings and transformations.  
Icon: Mock symbol.

**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.  
Region dropdown: us-east-1  
Search bar: Q arn:aws:lambda:us-east-1:462972487428:function:Serv X

**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.

**Create method** button.

Screenshot of the AWS API Gateway Resources page for the /serverlessapi endpoint.

**API Gateway**

**Resources**

**/serverlessapi - POST - Method execution**

ARN: arn:aws:execute-api:us-east-1:462972487428:wmssswkxil/\*/POST/serverlessapi

Resource ID: 7bolyd

Method request → Integration request → Lambda integration

Client ← Method response ← Integration response ← Integration response ← Method response ← Test

**Method request settings**

- Authorization: NONE
- Request validator: None
- API key required: False
- SDK operation name: Generated based on method and path

**Request paths (0)**

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Testing post method

Screenshot of the AWS API Gateway Test page for the /serverlessapi endpoint.

**API Gateway**

**Resources**

**/serverlessapi - POST**

**Request body:**

```

1 "first_name": "Gaurav",
2 "last_name": "Kothavade",
3 "email_id": "kothavadegaurav@gmail.com",
4 "message": "universe"
5
6

```

**Test**

**/serverlessapi - POST method test results**

|                  |  |             |
|------------------|--|-------------|
| Request          | Latency: 1250  | Status: 200 |
| Response body    | null   |             |
| Response headers | Content-Type: application/json X-Amzn-Trace-Id: Root=1-65e96c23-1fa3033f9078ed6541c4764f; Parent=4fcfeffad39690136; Sampled=0; lineage=e7a433cb:0  |             |
| Log              | Execution log for request c0054d30-40ea-476f-8013-83d480de0839 Thu Feb 29 10:56:35 UTC 2024 : Starting execution for request: c0054d30-40ea-476f-8013-83d480de0839 Thu Feb 29 10:56:35 UTC 2024 : HTTP Method: POST, Resource Path: /serverlessapi |             |

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## It reflects in the dynamodb table

The screenshot shows the AWS DynamoDB 'Explore items' interface. The left sidebar has 'Explore items' selected. The main area shows a table named 'Serverlessapp' with three items returned. The table structure includes columns for Email ID, First Name, Last Name, and Message.

| Email ID (String)      | First Name | Last Name | Message  |
|------------------------|------------|-----------|----------|
| kothavadegaurav@g...   | Gaurav     | Kothavade | Universe |
| amolk04@gmail.com      | Amol       | kanchar   | India    |
| rinkeshmore@gmail.c... | Rinkesh    | More      | World    |

## Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

**Stage**

\*New stage\*

**Stage name**

dev

**Deployment description**

Deployment of website

**Cancel** **Deploy**

Authorization: NONE

API key required: False

## Uploading html file

The screenshot shows the AWS S3 console interface for uploading files. The top navigation bar includes 'Services', a search bar, and user information: 'vocabs/user3011557=amol.kanchar@dctinc.com @ 4629-7248-7428'. The main area shows the 'Upload' step for the 'servelessapp-buck' bucket. A large central box allows dragging and dropping files or choosing them via 'Add files' or 'Add folder'. Below this, a table lists the selected file: 'contactus.html' (1 Total, 2.0 KB). The 'Destination' section shows the target bucket as 's3://servelessapp-buck'. At the bottom, there are links for 'Permissions', 'CloudShell', and 'Feedback', along with copyright and legal notices.

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose [Add files](#) or [Add folder](#).

**Files and folders (1 Total, 2.0 KB)**  
All files and folders in this table will be uploaded.

| Name           | Type |
|----------------|------|
| contactus.html | text |

**Destination**

Destination  
s3://servelessapp-buck

▶ Destination details  
Bucket settings that impact new objects stored in the specified destination.

▼ Permissions

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws Services Search [Alt+S]

Destination  
s3://servelessapp-buck

▶ Destination details Bucket settings that impact new objects stored in the specified destination.

▼ Permissions Grant public access and access to other AWS accounts.

**Access control list (ACL)**  
Grant basic read/write permissions to other AWS accounts. [Learn more](#)

**!** AWS recommends using S3 bucket policies or IAM policies for access control. [Learn more](#)

Access control list (ACL)  
 Choose from predefined ACLs  
 Specify individual ACL permissions  
Predefined ACLs  
 Private (recommended)  
Only the object owner will have read and write access.  
 Grant public-read access  
Anyone in the world will be able to access the specified objects. The object owner will have read and write access. [Learn more](#)

**!** **Granting public-read access is not recommended**  
Anyone in the world will be able to access the specified objects. [Learn more](#)

I understand the risk of granting public-read access to the specified objects.

▶ Properties Specify storage class, encryption settings, tags, and more.

Cancel **Upload**

aws Services Search [Alt+S] Global v vclabs/user3011557=amol.kanchar@dctinc.com @ 4629-7248-7428 v

☰ Upload succeeded View details below.

### Upload: status

The information below will no longer be available after you navigate away from this page.

#### Summary

| Destination            | Succeeded                | Failed            |
|------------------------|--------------------------|-------------------|
| s3://servelessapp-buck | 1 file, 2.0 KB (100.00%) | 0 files, 0 B (0%) |

[Files and folders](#) Configuration

#### Files and folders (1 Total, 2.0 KB)

| Name             | Folder | Type      | Size   | Status    | Error |
|------------------|--------|-----------|--------|-----------|-------|
| contactus.htm... | -      | text/html | 2.0 KB | Succeeded | -     |

Find by name < 1 >

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences