

API GATEWAY:

Go to lambda function and create a lambda function.

The screenshot shows the AWS Lambda homepage under the 'Compute' category. The main heading is 'AWS Lambda' with the subtext 'lets you run code without thinking about servers.' Below this, a note states: 'You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.' To the right, a 'Get started' box contains the text: 'Author a Lambda function from scratch, or choose from one of many preconfigured examples.' A prominent orange 'Create a function' button is at the bottom of this box. At the bottom of the main content area, there's a code editor preview with the title 'How it works' and tabs for '.NET', 'Java', 'Node.js', 'Python', 'Ruby', and 'Custom runtime'. The 'Node.js' tab is selected, showing the code: '1 exports.handler = async (event) => {'. To the right of the code editor are buttons for 'Run' and 'Next: Lambda responds to events'.

Give function as author from scratch, give name and mention run time and save.

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The top navigation bar includes 'Lambda > Functions > Create function'. The first step, 'Create function', has the subtext 'Choose one of the following options to create your function.' It features three radio button options: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. The 'Author from scratch' option includes a note: 'Start with a simple Hello World example.' The second step, 'Basic information', has a 'Function name' field containing 'HelloLambda'. A note below it says: '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 (_).' The 'Runtime' section shows 'Node.js 22.x' selected. A note below it says: 'Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.' The third step, 'Architecture', shows 'x86_64' selected. A note below it says: 'Choose the instruction set architecture you want for your function code.'

The screenshot shows the AWS Lambda Functions console. In the top navigation bar, 'HelloLambda' is selected under 'Functions'. A green success message at the top states: 'Successfully created the function HelloLambda. You can now change its code and configuration. To invoke your function with a test event, choose "Test".'. On the left, there's a 'Function overview' section with tabs for 'Diagram' (selected) and 'Template'. It displays a box labeled 'HelloLambda' with a 'Layers' section showing '(0)'. Below these are buttons for '+ Add trigger' and '+ Add destination'. On the right, there's a 'Description' section with 'Last modified 2 minutes ago', a 'Function ARN' section with the value 'arn:aws:lambda:us-east-1:235351028455:function:HelloLambda', and a 'Function URL' section with a link. At the bottom of the main content area are tabs for 'Code' (selected), 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. To the right, there's a sidebar with 'Info' selected, containing a 'Create a' section with a bulleted list: 'Build a new function', 'Import a function', 'Import from URI', 'Import from file', and 'Import from GitHub'. Below that is a 'Learn more' section with a 'Start' button.

```
def lambda_handler(event, context):  
    return {  
        'statusCode': 200,  
        'body': 'Hello from API Gateway!'  
    }
```

Write this code and deploy it.

The screenshot shows the AWS Lambda Code source editor for the 'HelloLambda' function. The top navigation bar shows 'HelloLambda' under 'Functions'. A green success message at the top states: 'Successfully created the function HelloLambda. You can now change its code and configuration. To invoke your function with a test event, choose "Test".'. Below this is a 'Code source' section with an 'Info' tab. On the left is an 'EXPLORER' sidebar showing a folder named 'HELLOLAMBDA' containing an 'index.js' file. The main area is a code editor with the file 'index.js' open. The code is identical to the one shown above. At the bottom of the code editor, there's a note: 'Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)'. At the very bottom of the editor are buttons for 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)'. To the right of the code editor, there are buttons for 'Open in Visual Studio Code' and 'Upload from ...'. The sidebar also includes a 'DEPLOY [UNDEPLOYED CHANGES]' section with a note: '⚠ You have undeployed changes.' and buttons for 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)'.

Go to api gateway

The screenshot shows the AWS API Gateway homepage. At the top, there's a navigation bar with the AWS logo, search bar, and account information. Below the header, the page title is "API Gateway" with the subtitle "Create and manage APIs at scale". A subtext states: "Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs." To the right, there's a "Get started" section with a "Create an API" button. The main content area has two sections: "How it works" (diagram showing the flow from users to API Gateway to various services like Lambda, S3, and databases) and "Pricing" (information about pay-as-you-go pricing). A red arrow points from the "Create an API" button towards the "Import" button on the next screen.

Click on create API

Select rest api click on build.

The screenshot shows the "Create API" wizard on the "REST API" step. The top navigation bar includes the AWS logo, search bar, and account information. The main content area has a subtext: "Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards." It also lists "Works with the following:" (Lambda, HTTP, AWS Services) and "Build" and "Import" buttons. A red arrow points from the "Import" button towards the "Build" button on the next screen.

Click on create new api, give the api name, select regional click on create.

aws | Search [Alt+S] | Account ID: 2353-5102-
Region: United States (N. Virginia) ▾

API Gateway > APIs > Create API > Create REST API

New API
Create a new REST API.

Import API
Import an API from an OpenAPI definition.

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

Example API
Learn about API Gateway with an example.

API name
HelloAPI

Description - optional

API endpoint type
Regional

IP address type | [Info](#)
Select the type of IP addresses that can invoke the default endpoint for your API.

IPv4
Supports only edge-optimized and Regional API endpoint types.

Dualstack
Supports all API endpoint types.

Click on resource and click create resource.

aws | Search [Alt+S] | Account ID: 2353-5102-
Region: United States (N. Virginia) ▾

API Gateway > APIs > Resources - HelloAPI (bvtktxugib)

API Gateway

- APIs
- Custom domain names
- Domain name access associations
- VPC links

▼ API: HelloAPI

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

Resources

Successfully created REST API 'HelloAPI' (bvtktxugib).

Create resource ↑

Resource details

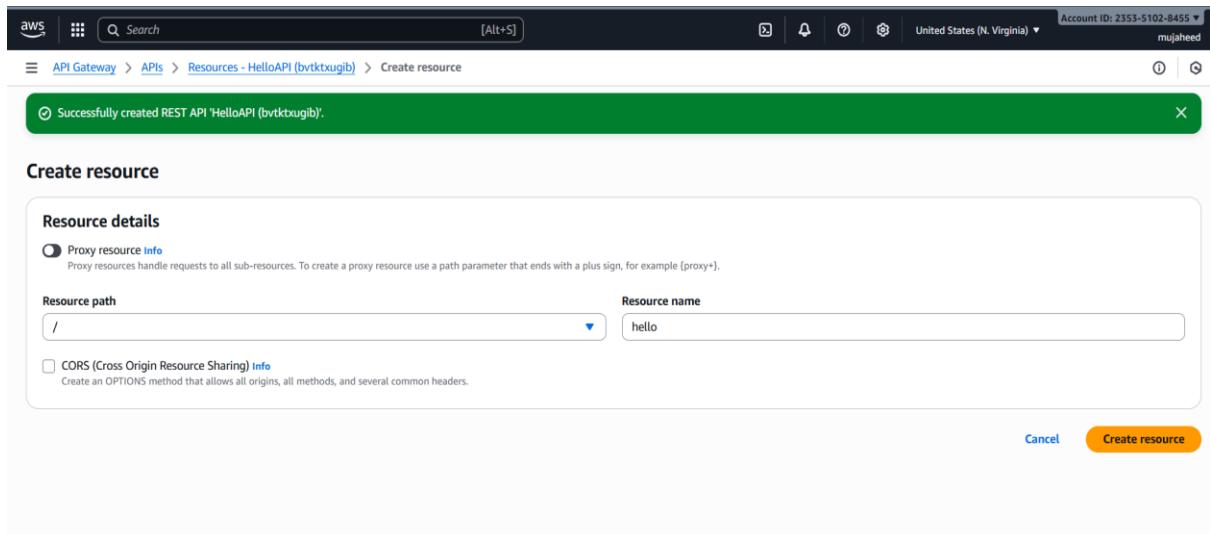
Path: / [Update documentation](#) [Enable CORS](#)

Resource ID: gdifqrbhil

Methods (0) [Delete](#) [Create method](#)

Method type	Integration type	Authorization	API key
No methods			

No methods defined.



API Gateway > APIs > Resources - HelloAPI (bvtktxugib)

Successfully created resource '/hello'

API Gateway

APIs
Custom domain names
Domain name access associations
VPC links

▼ API: HelloAPI

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

Resources

Create resource

/ /hello

Resource details

Path /hello

Resource ID jd0ws

Methods (0)

No methods

No methods defined.

API actions Deploy API

Click on create method

API Gateway > APIs > Resources - HelloAPI (bvtktxugib)

Successfully created resource '/hello'

API Gateway

APIs
Custom domain names
Domain name access associations
VPC links

▼ API: HelloAPI

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

Resources

Create resource

/ /hello

Resource details

Path /hello

Resource ID jd0ws

Methods (0)

Method type Integration type Authorization API key

No methods

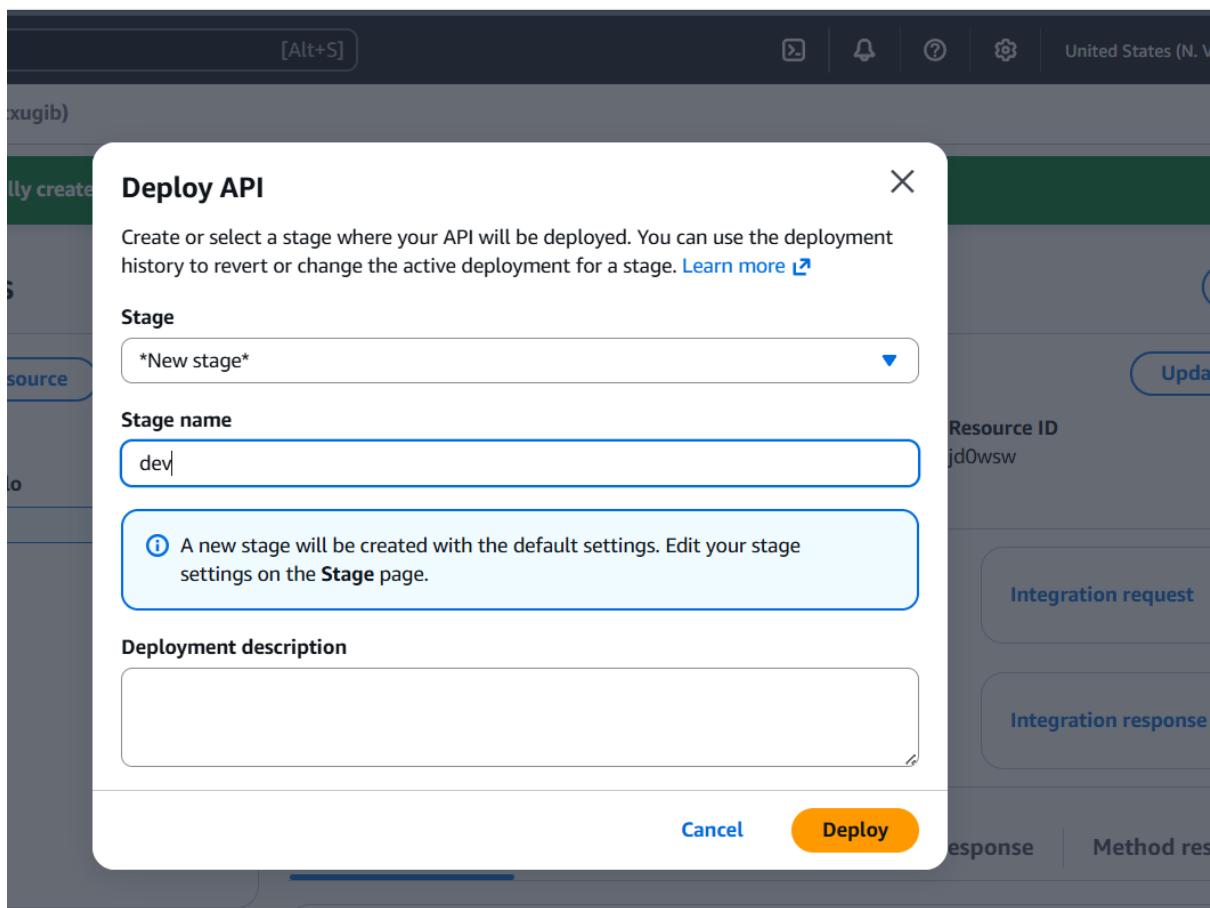
No methods defined.

API actions Deploy API

- Give the method as GET
- Select lambda function
- Select your lambda function.

Click on deploy api

The screenshot shows the AWS API Gateway Resources page for an API named "HelloAPI". A green success message at the top states: "Successfully created method 'GET' in 'hello'. Redeploy your API for the update to take effect." On the right, there are "API actions" and a prominent orange "Deploy API" button, which is highlighted with a red arrow. The left sidebar shows navigation options like APIs, Custom domain names, Domain name access associations, VPC links, and a detailed view for "API: HelloAPI" with sections for Resources, Stages, Authorizers, Gateway responses, Models, Resource policy, and Documentation.



It will give invoke URL copy that URL and paste on browser and add /hello at last it will give you Hello from API "Gateway!"

Screenshot of the AWS API Gateway Stages page for the HelloAPI (bvtktxugib) API.

The left sidebar shows the API Gateway navigation menu and the current API selection: API: HelloAPI > Stages.

The main content area displays the "Stages" section for the "dev" stage. The "Stage details" table includes:

Stage name	Rate	Web ACL
dev	10000	-
Cache cluster	Burst	Client certificate
Inactive	5000	-
Default method-level caching		
Inactive		

Below the table are two sections: "Invoke URL" (https://bvtktxugib.execute-api.us-east-1.amazonaws.com/dev) and "Active deployment" (73tgi3 on October 29, 2025, 15:46 (UTC+05:30)).

The bottom section, "Logs and tracing", has an "Edit" button.