

# API Gateway REST API Setup – Documentation

## 1. Introduction

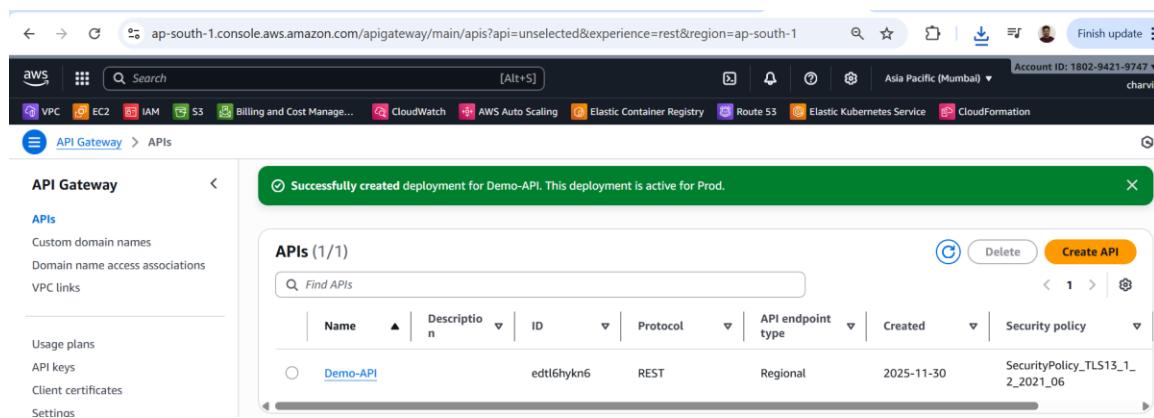
This document provides the step-by-step procedure for creating a REST API using Amazon API Gateway.

## 2. API Creation Steps

Follow the below steps to create the API and test it successfully.

### Step 1: Open API Gateway

Navigate to AWS Console → API Gateway → Create API.

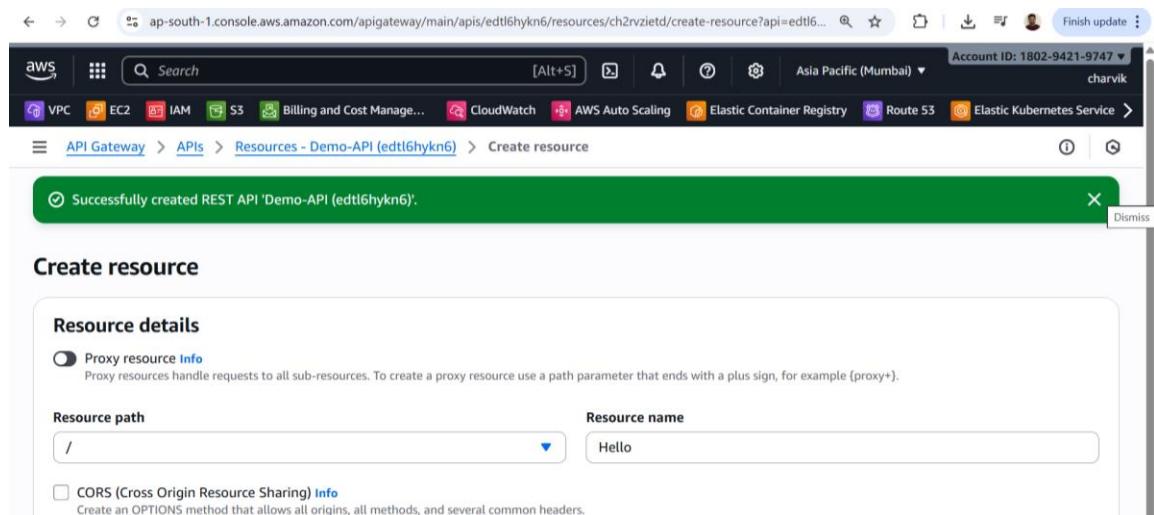


The screenshot shows the AWS API Gateway 'APIs' page. A green success message at the top states: "Successfully created deployment for Demo-API. This deployment is active for Prod." The main table displays one API entry:

Name	Description	ID	Protocol	API endpoint type	Created	Security policy
Demo-API	edtl6hykn6	REST	Regional	2025-11-30	SecurityPolicy_TLS13_1_2_2021_06	

### Step 2: Create REST API

Choose REST API → Build → Name: Demo-API → Create API.



The screenshot shows the 'Create resource' dialog. A green success message at the top states: "Successfully created REST API 'Demo-API (edtl6hykn6)'." The dialog has the following fields:

- 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+}.
  - CORS (Cross Origin Resource Sharing) Info  
Create an OPTIONS method that allows all origins, all methods, and several common headers.
- Resource path**: /
- Resource name**: Hello

### Step 3: Create Resource

Click Resources → Actions → Create Resource.

Resource Path: /hello

Resource Name: hello

The screenshot shows the AWS API Gateway Resources page. On the left sidebar, under the 'API: Demo-API' section, 'Resources' is selected. In the main area, there is a 'Create resource' button and a list item with the path '/Hello'. To the right, the 'Resource details' section shows the path '/Hello' and a Resource ID of p6vwsf. Below it, the 'Methods (0)' section indicates 'No methods defined.'

### Step 4: Create GET Method

Under /hello → Actions → Create Method → Choose GET → Integration type: Mock.

The screenshot shows the 'Create method' page for the '/Hello' resource. Under 'Method type', 'GET' is selected. Under 'Integration type', the 'Mock' option is selected, indicated by a blue border around its box. Other options shown include 'Lambda function', 'HTTP', 'AWS service', and 'VPC link'.

## Step 5: Configure Mock Integration Response

Set response body:

```
{  
  "message": "API working successfully!"  
}
```

Method response status code  
200

Content handling | Learn more ↗  
Passthrough

Header mappings Info  
No headers for this status code

▼ Mapping templates

Content type  
application/json

Generate template  
Remove

Template body

```
1 + {}
2 "message": "API working successfully!"
3 {}
4 {}
```

## Step 6: Deploy the API

Actions → Deploy API → New Stage: prod.

Successfully created deployment for Demo-API. This deployment is active for Prod.

Stage
Prod

Stage details Info

Stage name  
Prod

Rate Info  
10000

Burst Info  
5000

Cache cluster Info  
Inactive

Default method-level caching Info  
Inactive

Web ACL  
-

Client certificate  
-

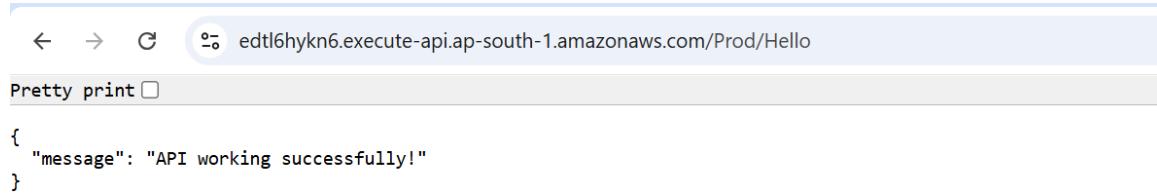
Invoke URL  
https://edtl6hykn6.execute-api.ap-south-1.amazonaws.com/Prod

Active deployment  
xtrt8c on November 30, 2025, 08:57 (UTC+05:30)

### Step 7: Test Invoke URL

Open browser and test:

<https://<api-id>.execute-api.<region>.amazonaws.com/prod>Hello>



A screenshot of a web browser window. The address bar shows the URL: "edtl6hykn6.execute-api.ap-south-1.amazonaws.com/Prod/Hello". Below the address bar is a toolbar with icons for back, forward, and search. The main content area displays a JSON response with the following code:

```
{  
  "message": "API working successfully!"  
}
```

Your API is now successfully created and returning the expected JSON response.