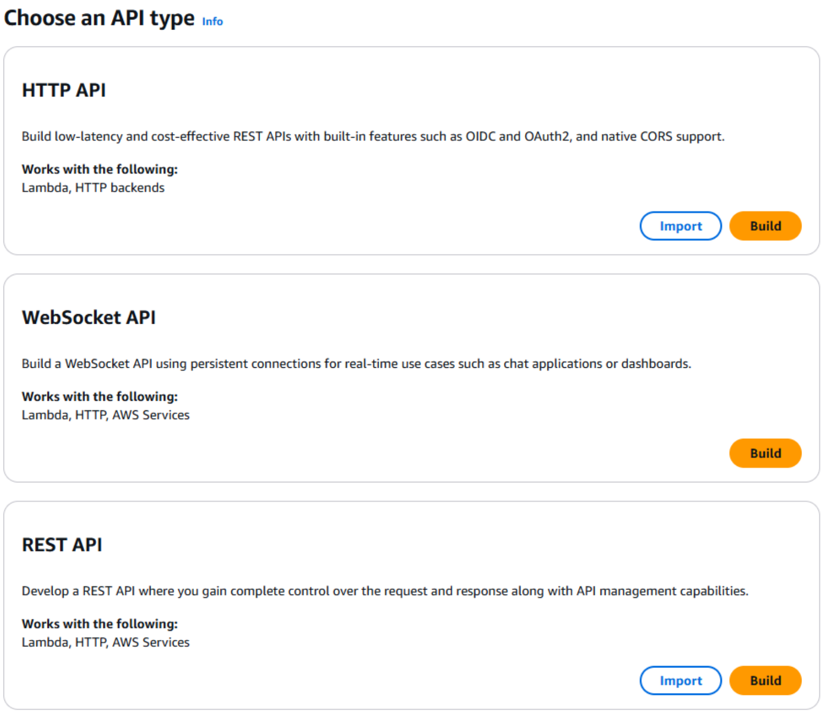
# Create a REST API



## Choose an API type Info

### HTTP API

Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.

**Works with the following:**
Lambda, HTTP backends

### WebSocket API

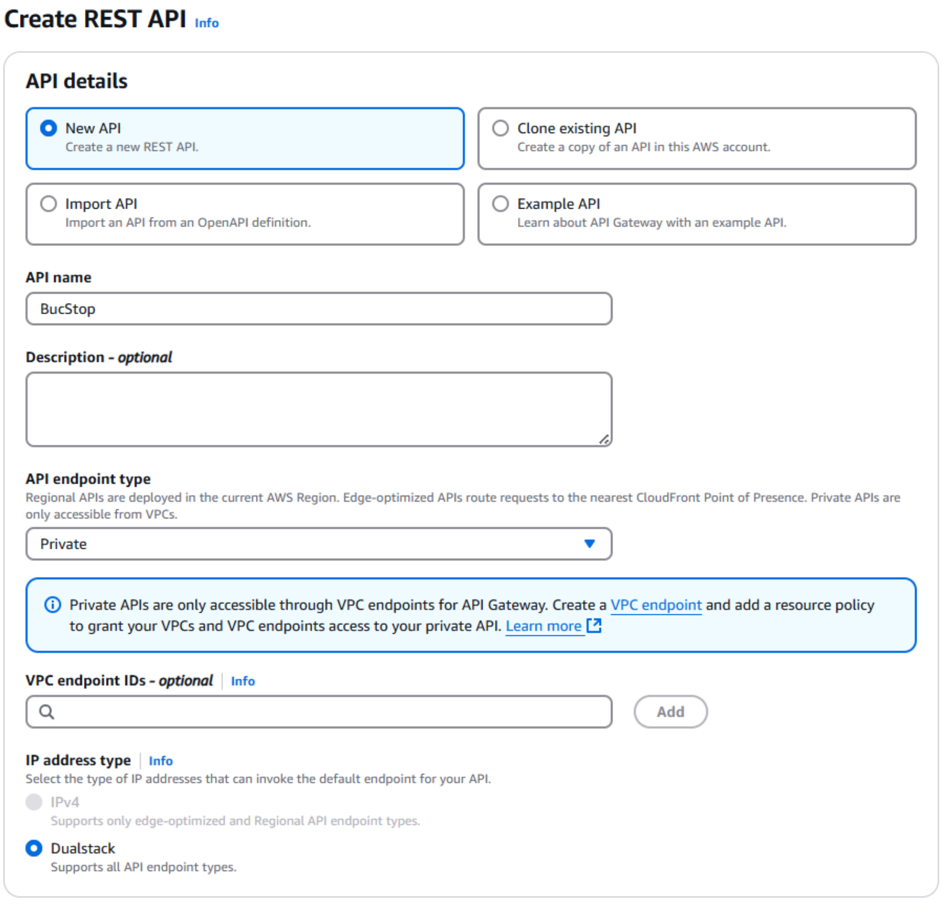
Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.

**Works with the following:**
Lambda, HTTP, AWS Services

### REST API

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



## Create REST API Info

### 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**
BucStop

**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.
Private ▼

ⓘ Private APIs are only accessible through VPC endpoints for API Gateway. Create a VPC endpoint and add a resource policy to grant your VPCs and VPC endpoints access to your private API. Learn more

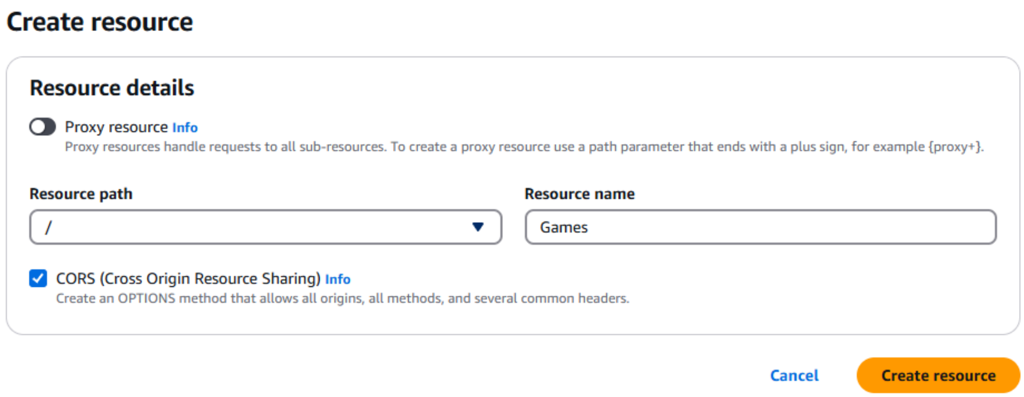
**VPC endpoint IDs -** ***optional*** | Info

**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.

## Create a 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+}.

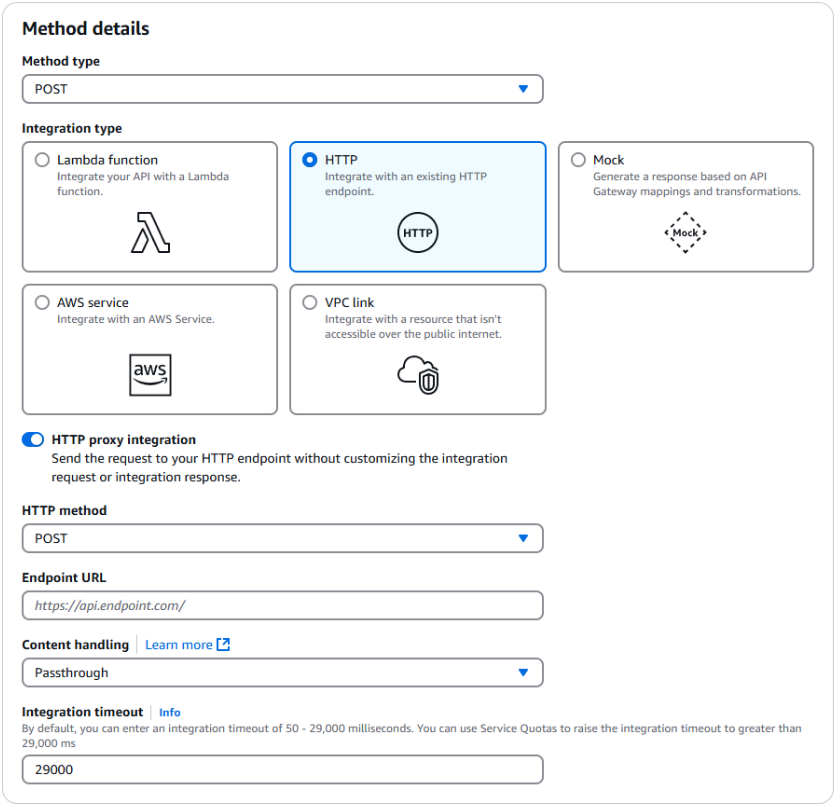
Resource path: /

Resource name: Games

☑ CORS (Cross Origin Resource Sharing) Info

Create an OPTIONS method that allows all origins, all methods, and several common headers.

## Create POST method



#### Method details

Method type: POST

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.

● HTTP proxy integration

Send the request to your HTTP endpoint without customizing the integration request or integration response.

HTTP method: POST

Endpoint URL: *https://api.endpoint.com/*

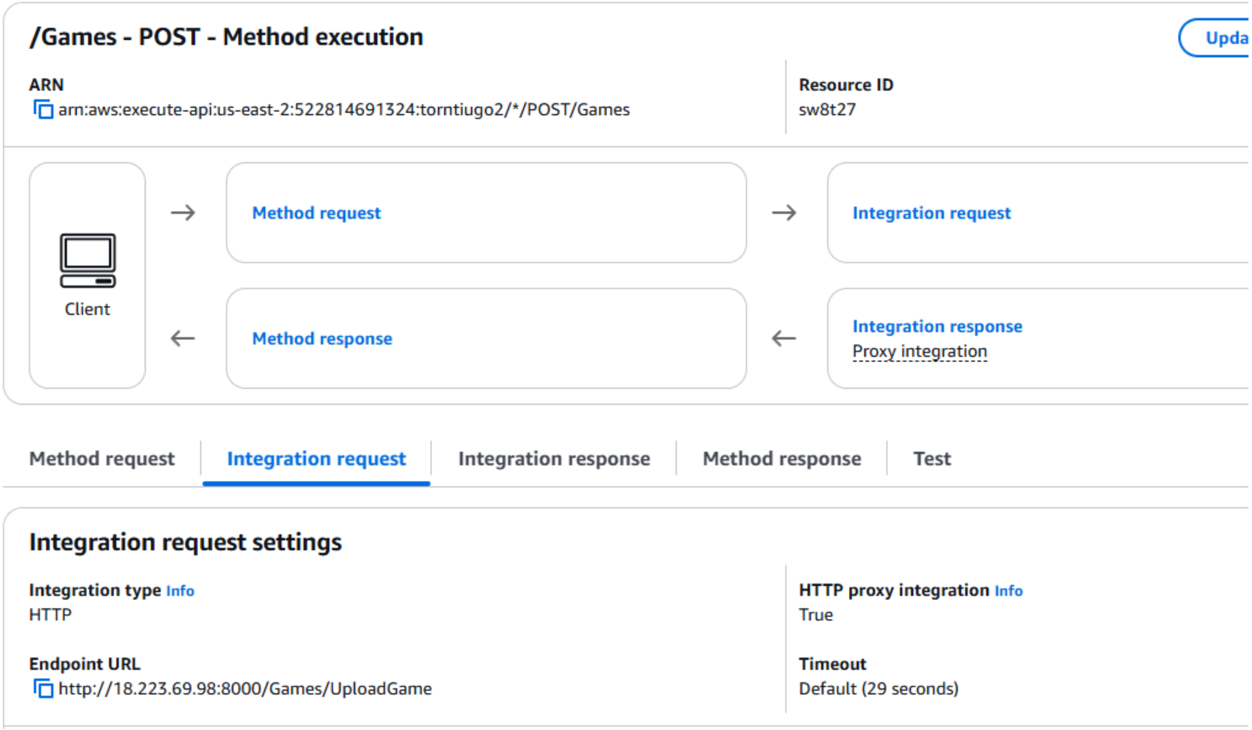
Content handling | Learn more: Passthrough

Integration timeout | Info

By default, you can enter an integration timeout of 50 - 29,000 milliseconds. You can use Service Quotas to raise the integration timeout to greater than 29,000 ms

29000

## Configure integration (Select Integration Request in the POST RESOURCE)



/Games - POST - Method execution

ARN
arn:aws:execute-api:us-east-2:522814691324:torntiugo2/*/POST/Games

Resource ID
sw8t27

Integration request settings

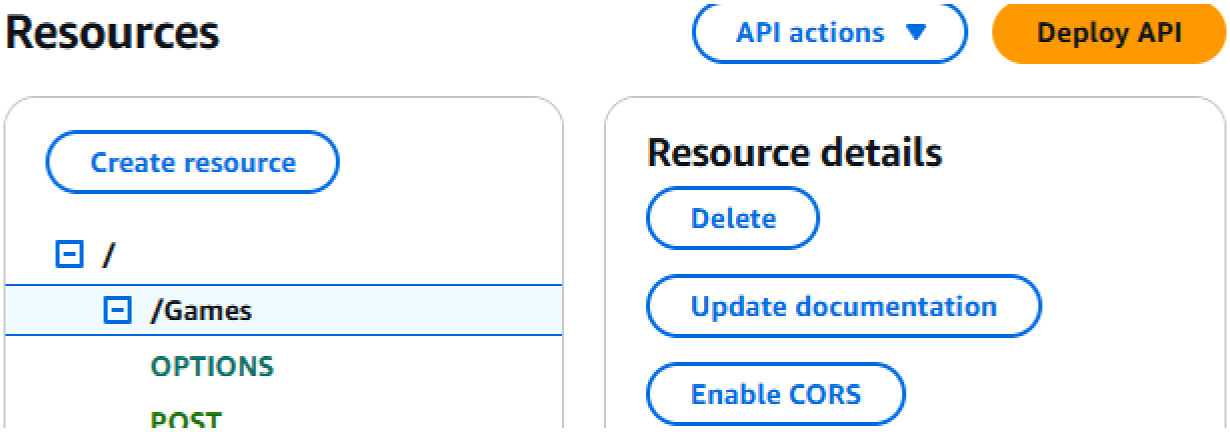
Integration type Info
HTTP

Endpoint URL
http://18.223.69.98:8000/Games/UploadGame

HTTP proxy integration Info
True

Timeout
Default (29 seconds)

## Enable CORS



Allow

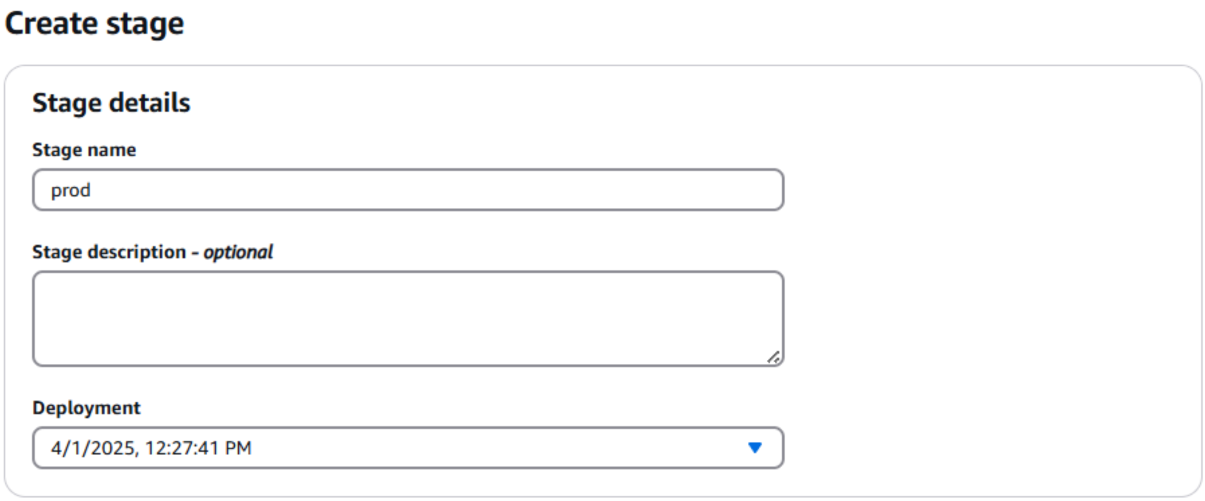
Origin: *

Methods: POST, OPTIONS

## Then Deploy API

Resources
API actions ▼
Deploy API

**Create a stage**



Create stage

Stage details

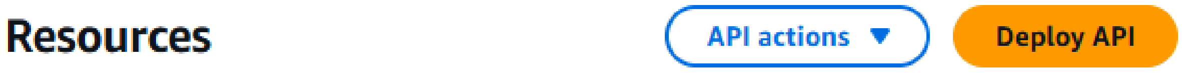
Stage name

prod

Stage description - *optional*

Deployment

4/1/2025, 12:27:41 PM



**Get invoke url**

https://torntiugo2.execute-api.us-east-2.amazonaws.com/prod

**Ensure code is up to date with the newest changes (Changes to Program.CS and GameController.cs)**

**Start the EC2 instance and run the webapp.**

**Run this command to test if the invoke URL works.**

You will insert your invoke URL in place of https://torntiugo2.execute-api.us-east-2.amazonaws.com/prod/Games

```
curl.exe -X POST https://torntiugo2.execute-api.us-east-2.amazonaws.com/prod/Games -F gameFile=@file.js
```

**In terminal you should see: “{"message":"File uploaded successfully!"}”**

**ls /mnt/ebs-storage to check that the file is there.**

| Test Case | Input | Expected Output |
|---|---|---|
| Create the API GW | Follow the steps above to create the AWS API gateway | Make sure you hit Deploy API at the end. |
| Start EC2 instance. | SSH into the EC2 instance and dotnet run | Web app starts |
| Create test.js file | Create a test.js file to use in the curl command | Test.js file created. |
| Test POST file command | THIS COMMAND NEEDS TO BE RUN IN THE TERMINAL OF YOUR COMPUTER, NOT THE EC2 INSTANCE.<br>curl  -X POST https://torntiugo2.execute-api.us-east-2.amazonaws.com/prod/Games -F "gameFile=@test.js" | “{"message":"File uploaded successfully!"}” |
| Verifying file is there | ls /mnt/ebs-storage | You should see the name of the test file you submitted. |

## CODE CHANGES

Change Program.cs

- Lines 9-10
- Lines 34-38
- Comment out 52
- Lines 54-58
- Comment out 76

Change GameController.cs

- Add POST method
- Lines 100-131

Test:

```
curl -X POST https://torntiugo2.execute-api.us-east-2.amazonaws.com/prod/Games -F "gameFile=@test.js"
```