In AWS, create a not-root (IAM) user and work with that user

1. Login as root

https://aws.amazon.com/

2. Search for the IAM service and create a new "user group" named "admins"

Example URL:

https://us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/home

3. Attach the desired "permission policies", in this case "AdministratorAccess"

4. In IAM, create a new user (in my case, I named it "user_educloud2024_admin")

- provide it access to the AWS management console
- pick "I want to create an IAM user"
- set a password
- add user to group "admins"
- optionally, apply a tag to the user, such as the meta-info "created_on"

5. Confirm the user creation

- Remember the user name
- Remember the user password
- Remember the 12 digits of the account ID

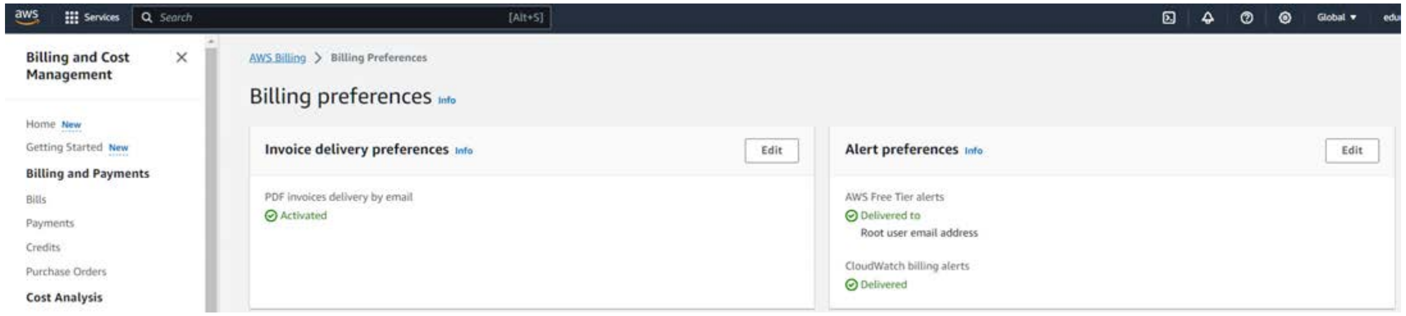
6. Look for "Billing and Cost Management", usually at the upper-right corner of the browser window.

Make sure you have selected the "Global" or US East (N. Virginia) region in the AWS Management Console.

Pick that menu option ("Billing and Cost Management"). The URL could be something like:

https://us-east-1.console.aws.amazon.com/costmanagement/home?region=us-east-1#/home

7. Look for and pick “Billing preferences”, usually at the left panel



Billing preferences Info

Invoice delivery preferences Info

PDF invoices delivery by email
Activated

Alert preferences Info

AWS Free Tier alerts
Delivered to
Root user email address

CloudWatch billing alerts
Delivered

8. Activate:

- CloudWatch billing alerts

- AWS Free Tier alerts Delivered to Root user email address

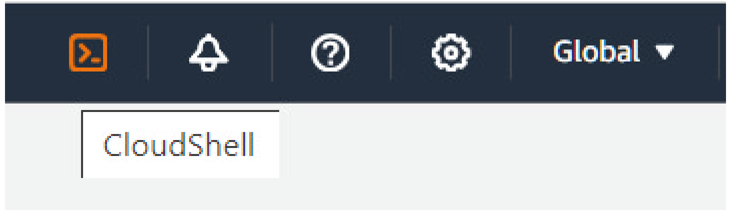
- PDF invoices delivery by email

9. Logout the root user

10. Login as the IAM user you just created.

In AWS, use the Cloud Shell to edit Python code and build a ZIP package, to create a “cloud function”, via AWS Lambda

1. Assuming you are logged-in to AWS, launch a “Cloud Shell”



2. Create a folder for the Python code + edit a lambda_function.py, using the “nano” editor

```
mkdir <my folder>

cd <my folder>

nano lambda_function.py
```

3. Type the Python code at https://arturmarques.com/edu/cn/files/w04/lambda_function.py.txt

```python
import json
from datetime import datetime
import pytz

def lambda_handler(event, context):
    # Define the European capitals with their respective time zones
    capitals = {
        "Lisbon" : "Europe/Lisbon",
        "London": "Europe/London",
        "Paris": "Europe/Paris",
        "Berlin": "Europe/Berlin",
        "Madrid": "Europe/Madrid",
        "Rome": "Europe/Rome"
    }
    dict_params_received = event.get('queryStringParameters', {})
    city_asked = dict_params_received.get('city', None)

    if city_asked and city_asked in capitals:
        # If a specific city is requested, return its current time
        timezone = pytz.timezone(capitals[city_asked])
        current_time = datetime.now(timezone).strftime('%Y-%m-%d %H:%M:%S')
        body = {city_asked: current_time}
    else:
        # If no specific city is requested, return times for all cities
        times_in_capitals = {}
        for capital, timezone in capitals.items():
            tz = pytz.timezone(timezone)
            current_time = datetime.now(tz).strftime('%Y-%m-%d %H:%M:%S')
            times_in_capitals[capital] = current_time
        # for
        body = times_in_capitals
    # if-else
    # Return the result as a JSON object
    return {
        'statusCode': 200,
        'body': json.dumps(body)
    }
# def lambda_handler
```

4. Save and exit the nano editor

CTRL^S

CTRL^X

5. The Python **pytz** library (for working with Time Zones) is a problem for most Python runtimes for cloud functions, because it is not installed, by default

In some PaaS cases a requirements.txt file solves the issue.

In this case:

# Install the pytz in the same folder where the Python source code is

```
pip install pytz -t .
```

zip the entire folder's contents to a ZIP file (at the parent folder in the following example):

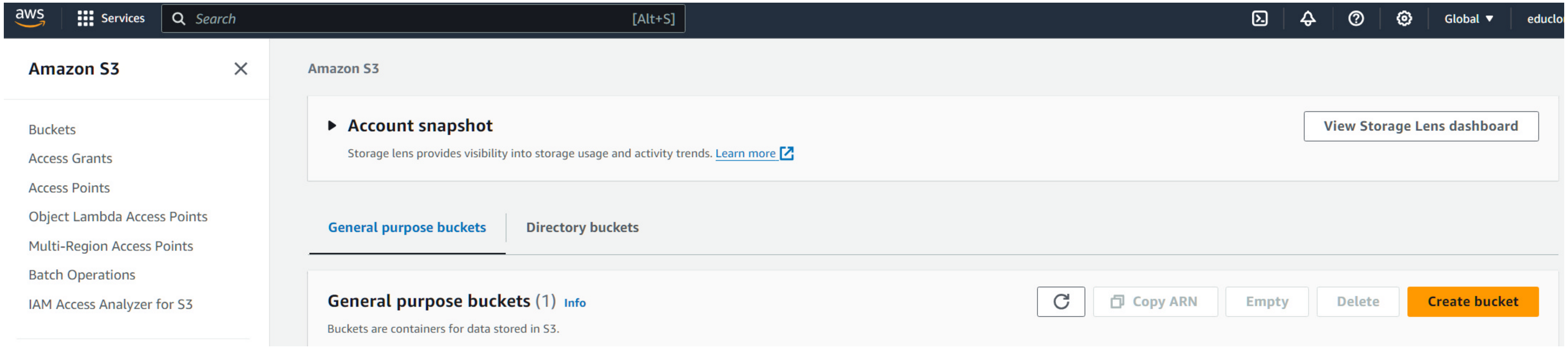
```
zip -r ../cf1.zip .
```

6. Create a bucket to store the ZIP out of the Cloud Shell

Search for the "S3" service.

https://s3.console.aws.amazon.com/s3/home?region=us-east-1

Pick "create bucket"



Amazon S3

Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3

Amazon S3

Account snapshot

Storage lens provides visibility into storage usage and activity trends. Learn more

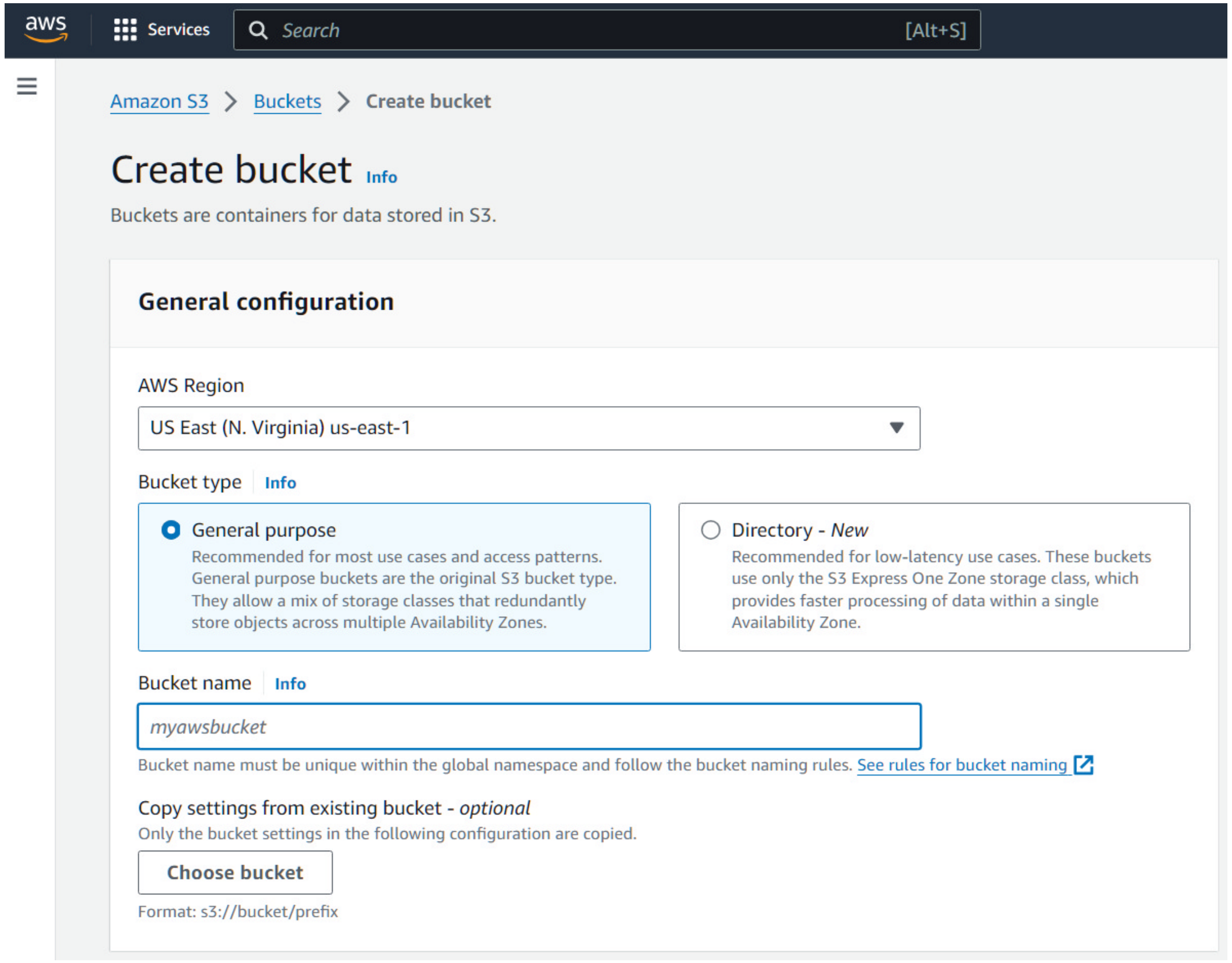
View Storage Lens dashboard

General purpose buckets | Directory buckets

General purpose buckets (1) Info

Buckets are containers for data stored in S3.

Copy ARN | Empty | Delete | Create bucket



Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type Info

[✓ 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

myawsbucket

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

Just name (with a unique name) the bucket, accept all the defaults, and scroll down to create it.

I named my bucket “educloud2024bucket”.

This bucket will have an internal URL such as:

s3://your-bucket-name/

s3://educloud2024bucket/

7. Upload, from Cloud Shell, the ZIP file to the bucket

aws s3 cp <your zip> s3://your-bucket-name/

```
aws s3 cp cf1.zip s3://educloud2024bucket/
```

This will make the copied file available at the following example URL:

https://s3.amazonaws.com/educloud2024bucket/cf1.zip

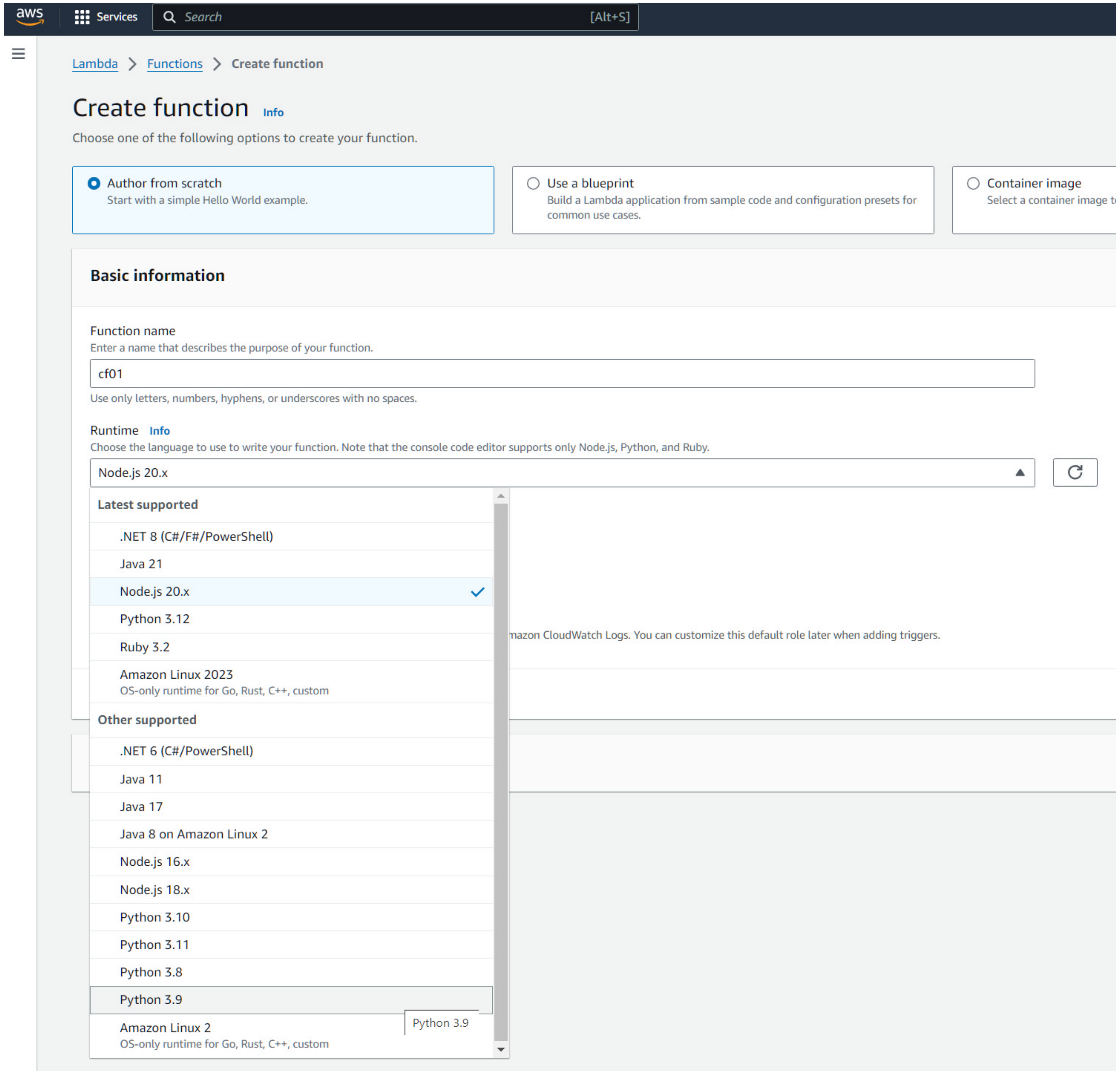
This URL will only work for authenticated and authorized users – not a problem, because it will only be needed for an upload moment, when using the AWS Lambda service.

8. Create a Lambda function, to run the code in the ZIP

Search for the "Lambda" service.

https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions

Pick "create function" with a Python 3.9 runtime.



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 t

Basic information

Function name
Enter a name that describes the purpose of your function.
cf01
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.
Node.js 20.x

Latest supported
- .NET 8 (C#/F#/PowerShell)
- Java 21
- Node.js 20.x ✓
- Python 3.12
- Ruby 3.2
- Amazon Linux 2023
  OS-only runtime for Go, Rust, C++, custom

Other supported
- .NET 6 (C#/PowerShell)
- Java 11
- Java 17
- Java 8 on Amazon Linux 2
- Node.js 16.x
- Node.js 18.x
- Python 3.10
- Python 3.11
- Python 3.8
- Python 3.9
- Amazon Linux 2
  OS-only runtime for Go, Rust, C++, custom

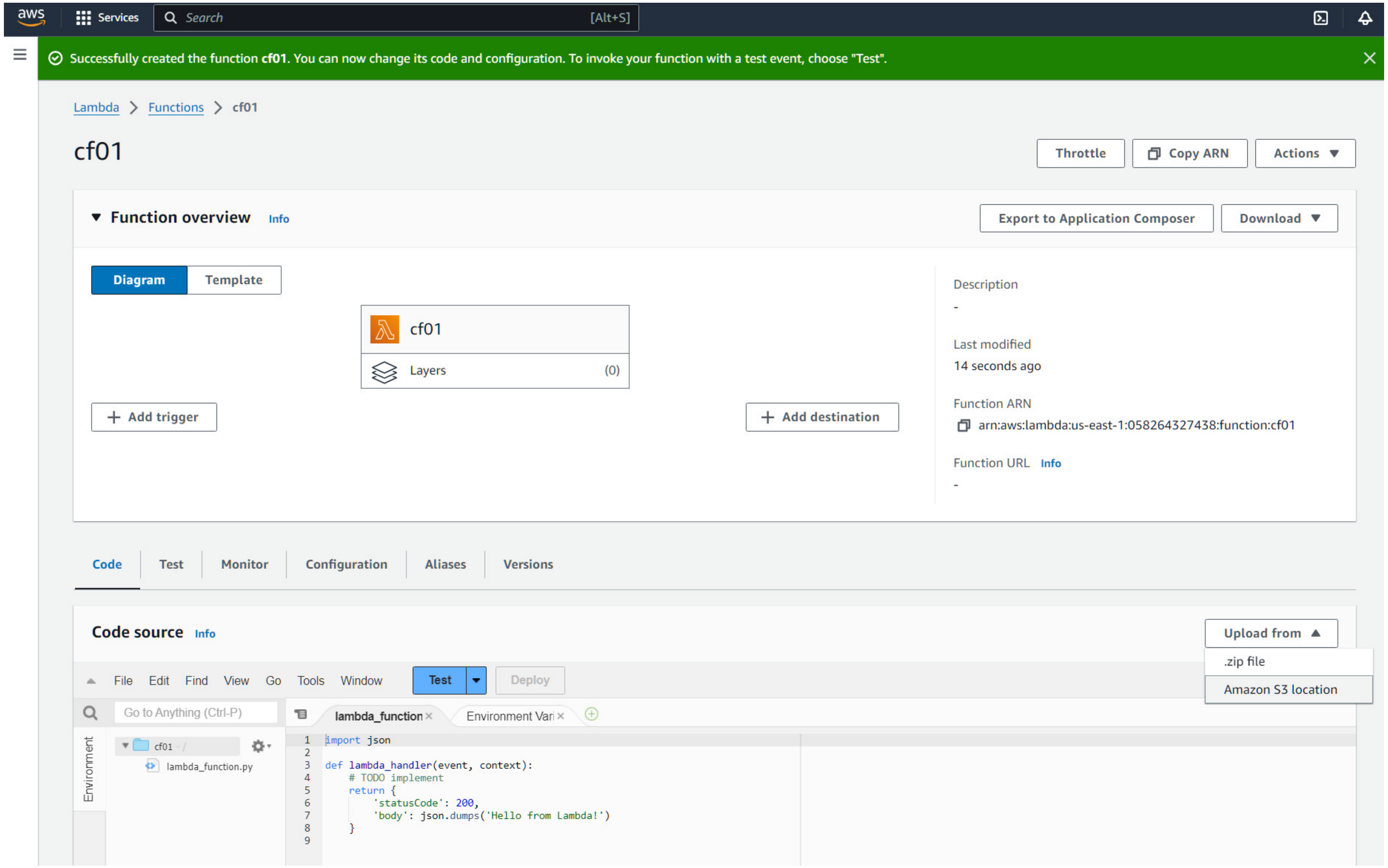
Python 3.9

mazon CloudWatch Logs. You can customize this default role later when adding triggers.

Once the function is created, choose “upload from” “Amazon S3 location” and supply a valid URL.

I used the URL: https://s3.amazonaws.com/educloud2024bucket/cf1.zip

The upload will result in file(s) and folder(s) in the Code tab. Make sure the main file is “lambda_function.py” and that the main function is “lambda_handler”.



Successfully created the function **cf01**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Description

-

Last modified

14 seconds ago

Function ARN

arn:aws:lambda:us-east-1:058264327438:function:cf01

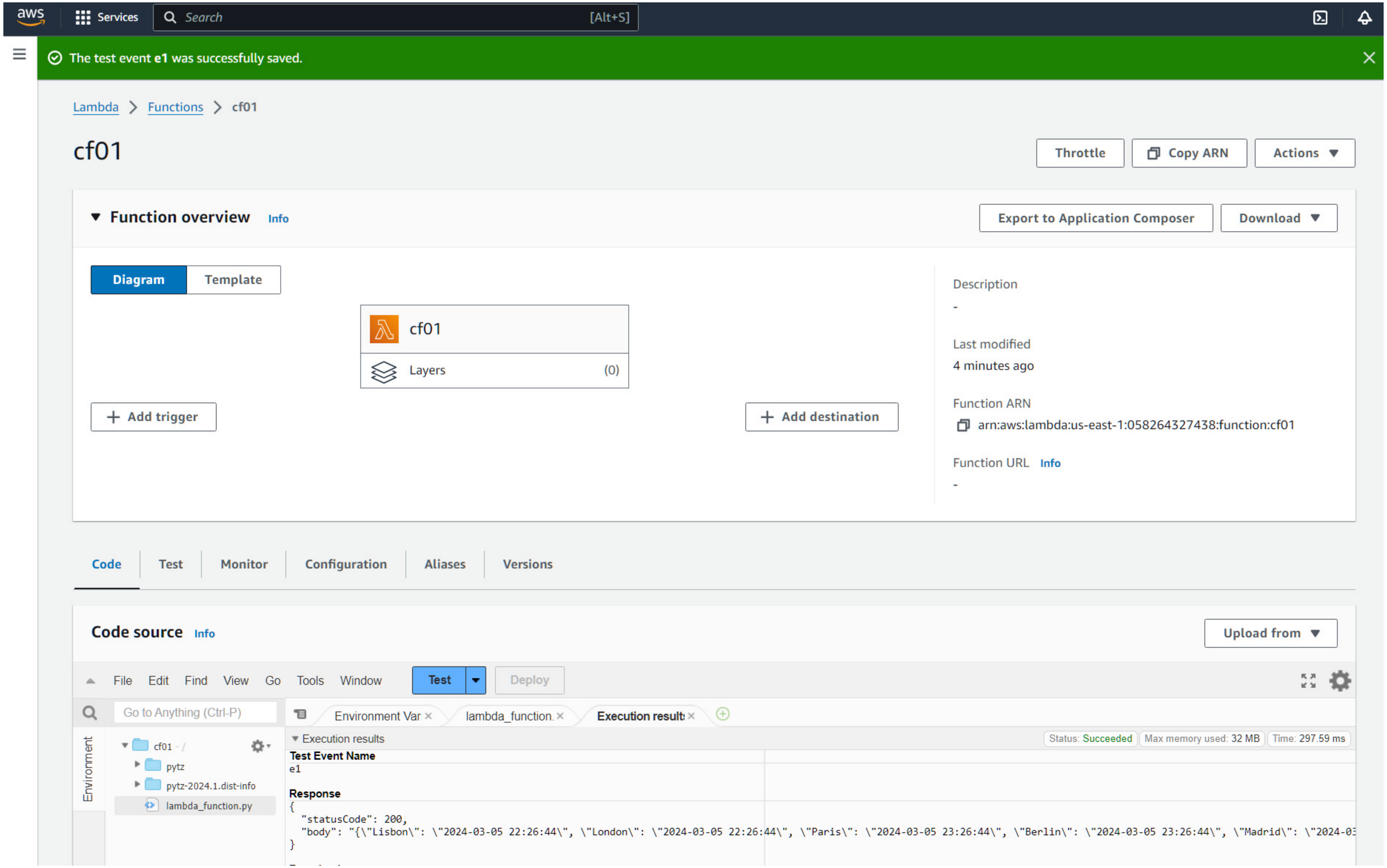
Function URL Info

-

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }

```



The test event **e1** was successfully saved.

cf01

Description
-

Last modified
4 minutes ago

Function ARN
arn:aws:lambda:us-east-1:058264327438:function:cf01

Function URL Info
-

Execution results   Status: Succeeded   Max memory used: 32 MB   Time: 297.59 ms

Test Event Name
```
e1
```

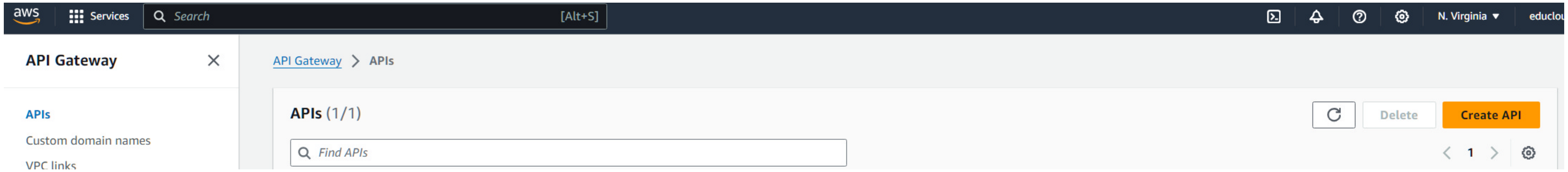
Response
```
{
  "statusCode": 200,
  "body": "{\"Lisbon\": \"2024-03-05 22:26:44\", \"London\": \"2024-03-05 22:26:44\", \"Paris\": \"2024-03-05 23:26:44\", \"Berlin\": \"2024-03-05 23:26:44\", \"Madrid\": \"2024-03
}
```

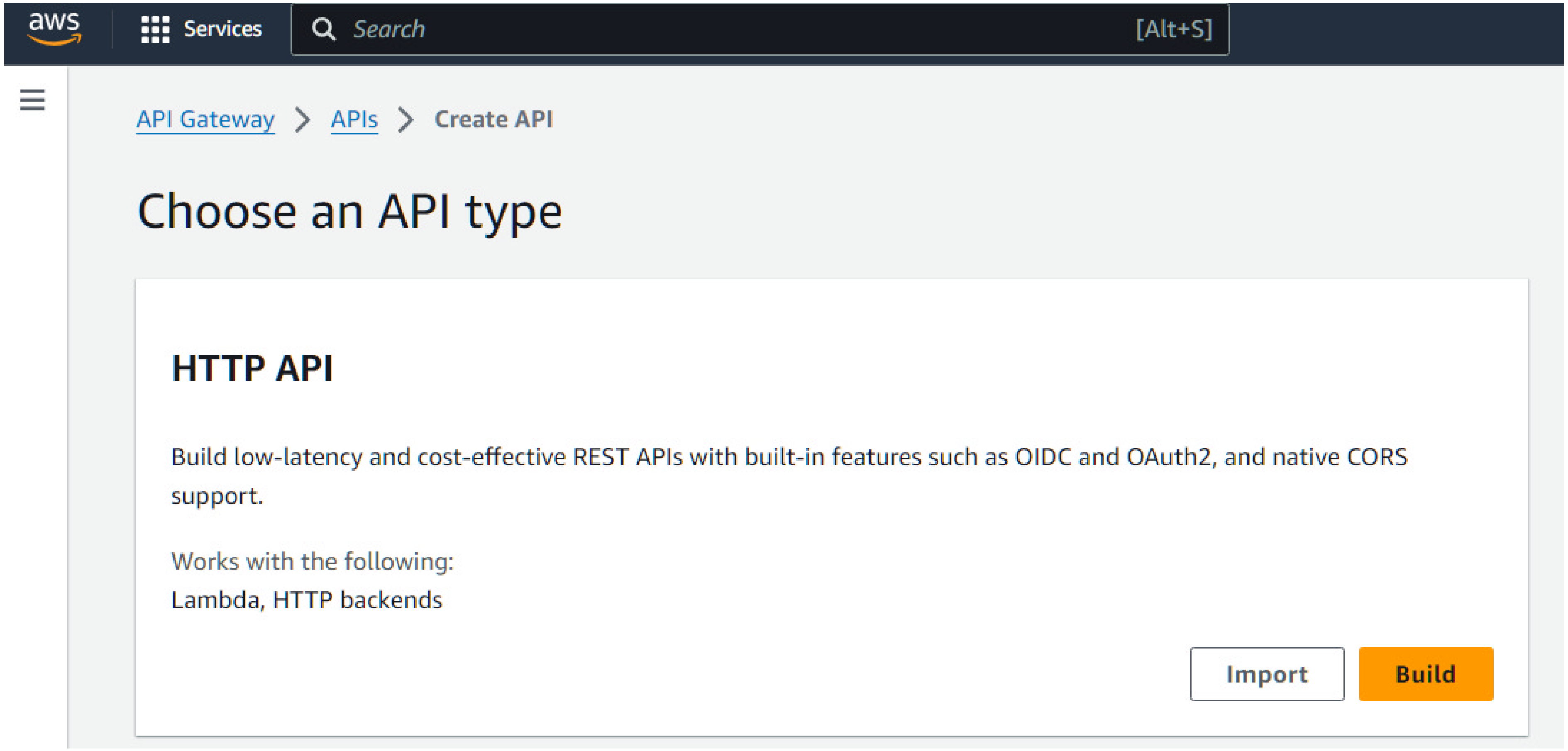
9 . Create a HTTP API so the function can be HTTP triggered by some route

Look for the “API Gateway” service.

https://us-east-1.console.aws.amazon.com/apigateway/main/apis?region=us-east-1

Create an HTTP “API”





API Gateway > APIs > Create API

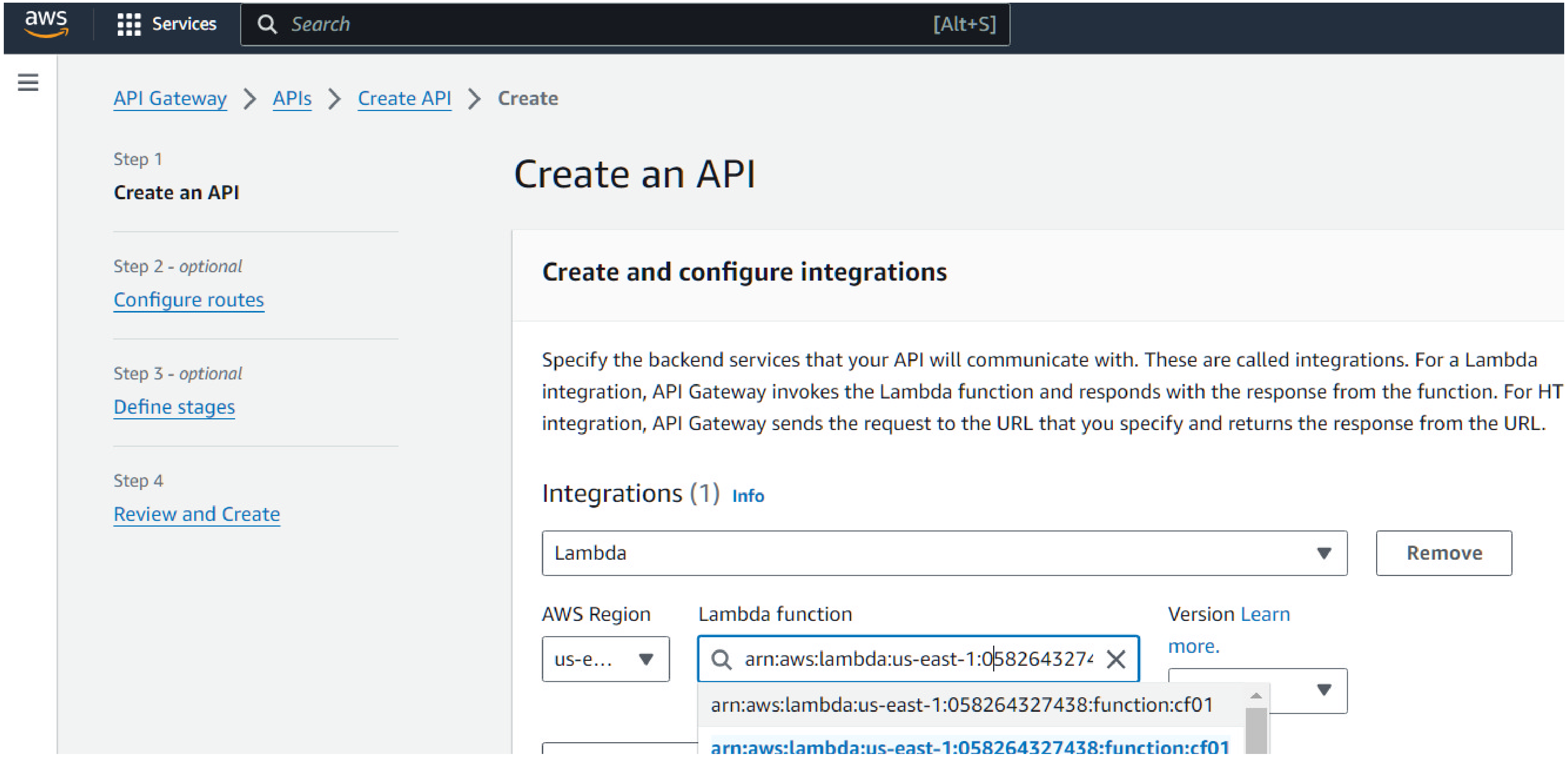
Choose an API type

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

10. Start by adding “Lambda” integration to the previously created Lambda function



Create an API

Create and configure integrations

Specify the backend services that your API will communicate with. These are called integrations. For a Lambda integration, API Gateway invokes the Lambda function and responds with the response from the function. For HT integration, API Gateway sends the request to the URL that you specify and returns the response from the URL.

Integrations (1) Info

Lambda

Remove

AWS Region: us-e...

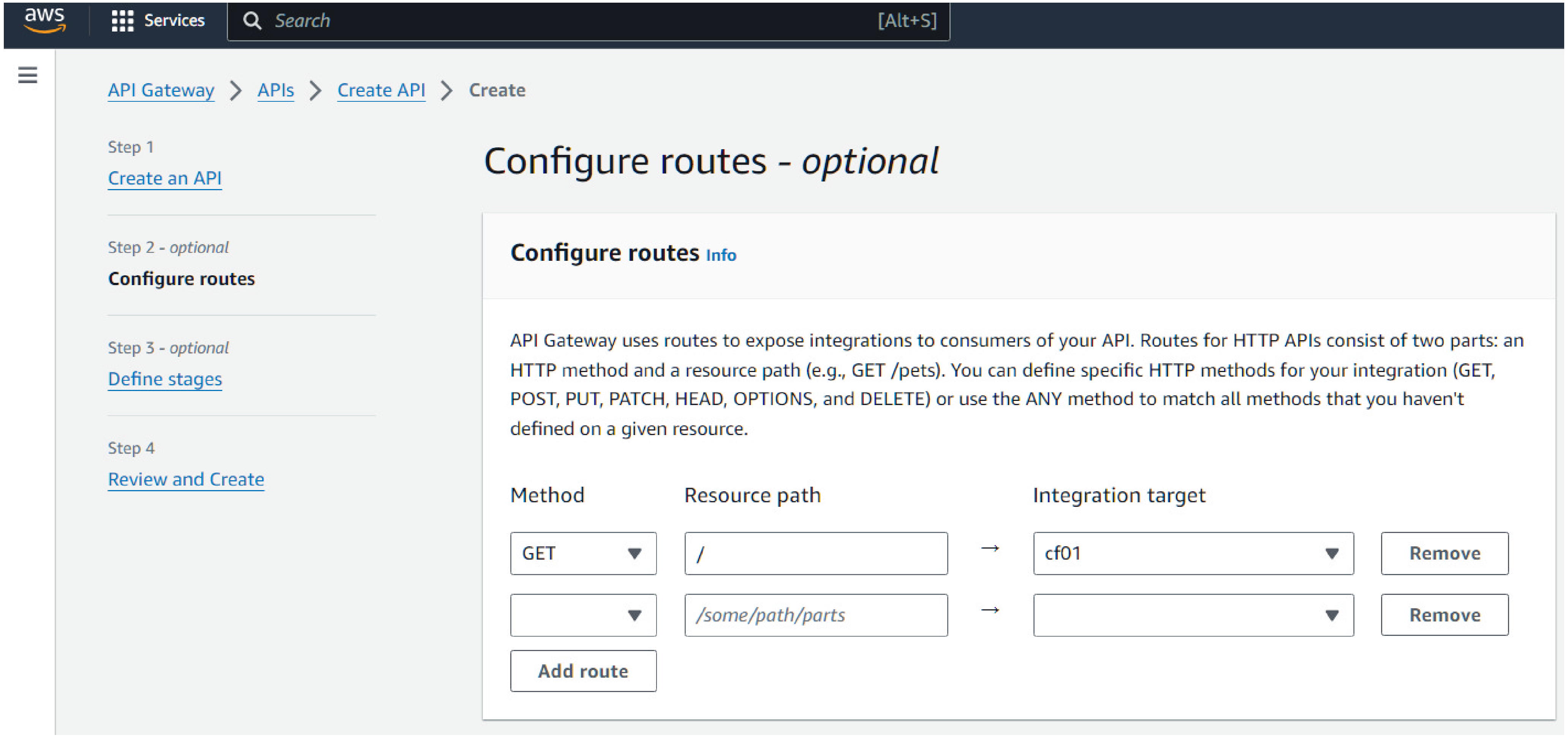
Lambda function: arn:aws:lambda:us-east-1:058264327

Version Learn more.

arn:aws:lambda:us-east-1:058264327438:function:cf01

arn:aws:lambda:us-east-1:058264327438:function:cf01

11. Add a route for GET /



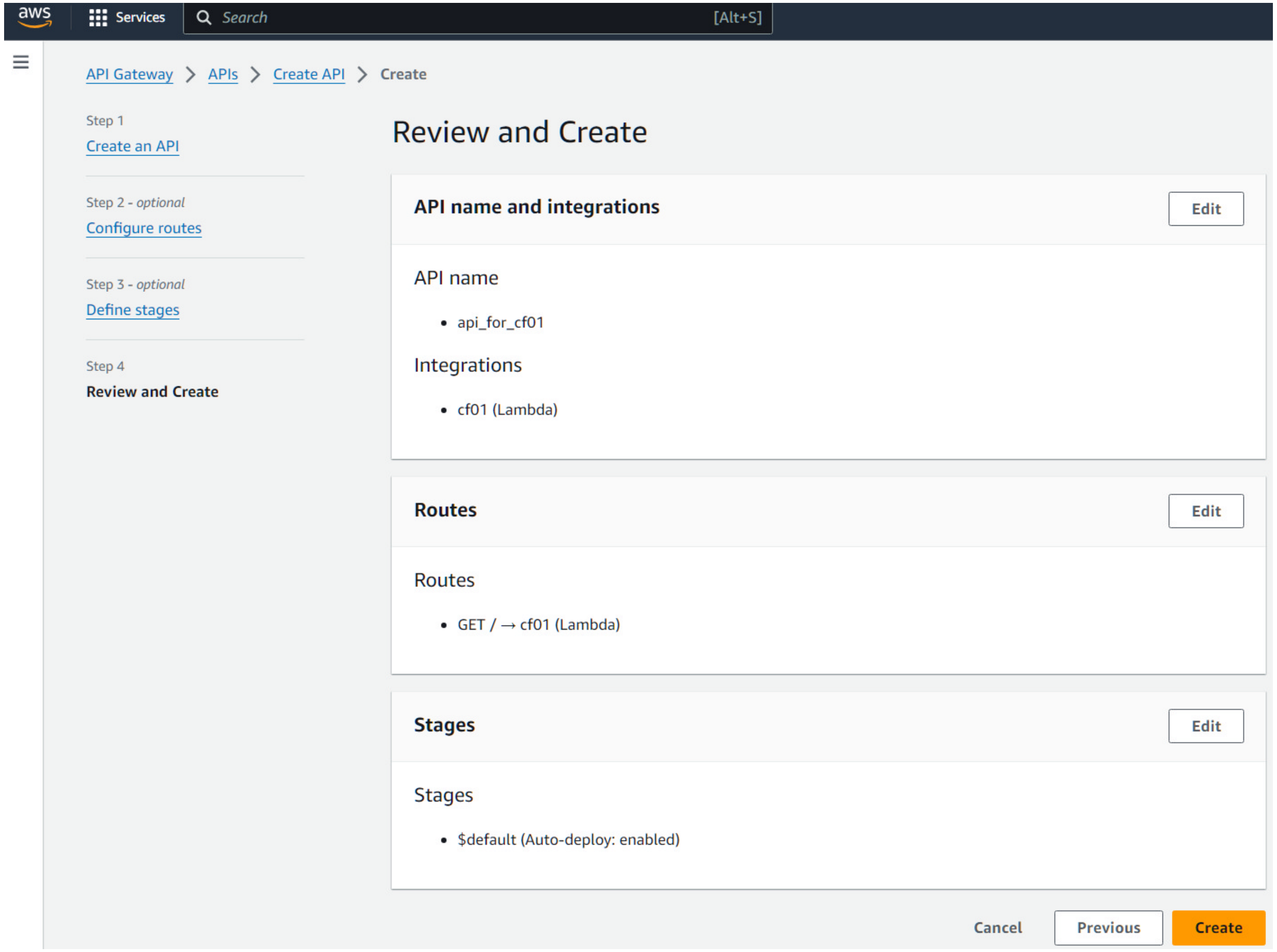
Configure routes - *optional*

Configure routes Info

API Gateway uses routes to expose integrations to consumers of your API. Routes for HTTP APIs consist of two parts: an HTTP method and a resource path (e.g., GET /pets). You can define specific HTTP methods for your integration (GET, POST, PUT, PATCH, HEAD, OPTIONS, and DELETE) or use the ANY method to match all methods that you haven't defined on a given resource.

Method	Resource path		Integration target	
GET	/	→	cf01	Remove
	/some/path/parts	→		Remove

12. Accept the default for “stage” and press “create”



Review and Create

API name and integrations

API name

- api_for_cf01

Integrations

- cf01 (Lambda)

Routes

Routes

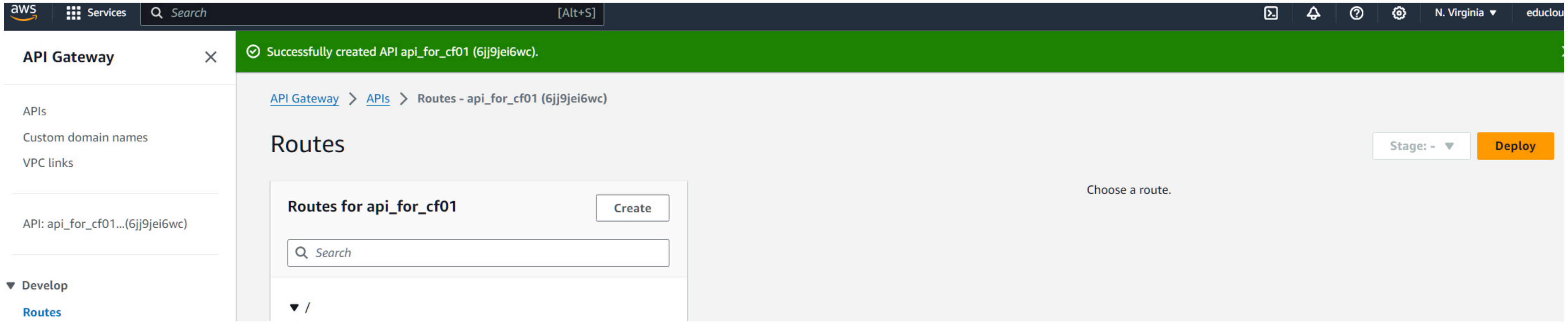
- GET / → cf01 (Lambda)

Stages

Stages

- $default (Auto-deploy: enabled)

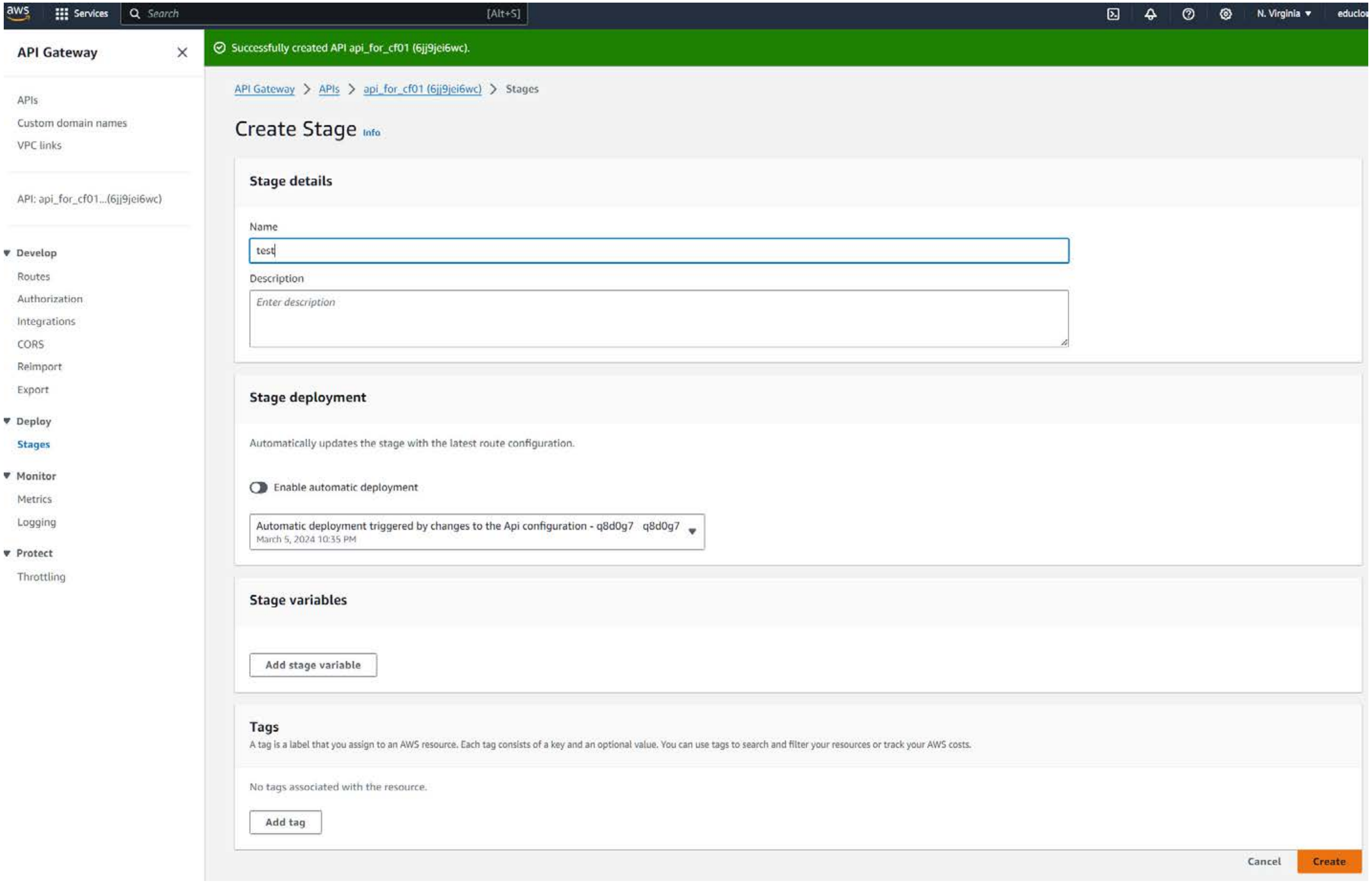
13. Deploy the API



Successfully created API api_for_cf01 (6jj9jei6wc).

Choose a route.

Upon deployment, accept to create a “test” (or other name) stage.



Successfully created API api_for_cf01 (6jj9jei6wc).

API Gateway > APIs > api_for_cf01 (6jj9jei6wc) > Stages

Create Stage Info

Stage details

Name: test

Description: Enter description

Stage deployment

Automatically updates the stage with the latest route configuration.

Enable automatic deployment

Automatic deployment triggered by changes to the Api configuration - q8d0g7 q8d0g7
March 5, 2024 10:35 PM

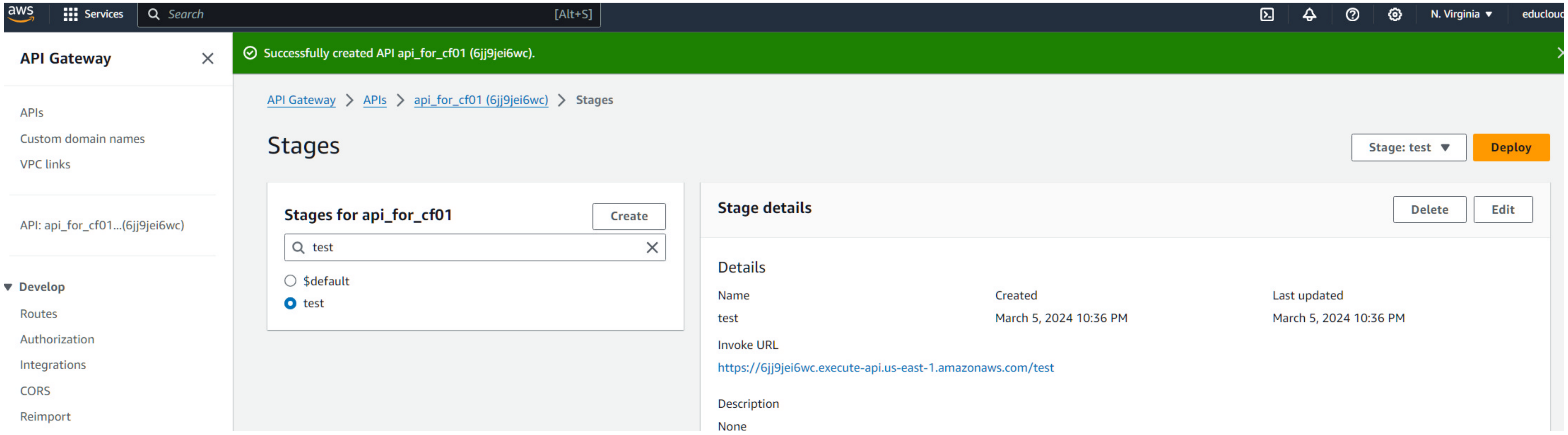
Stage variables

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

A deployment URL will become visible, pick “deploy”, you’ll be asked to create a “stage”.



Successfully created API api_for_cf01 (6jj9jei6wc).

API Gateway > APIs > api_for_cf01 (6jj9jei6wc) > Stages

API: api_for_cf01...(6jj9jei6wc)

Stages

Stages for api_for_cf01

test

( ) $default
(•) test

Stage details

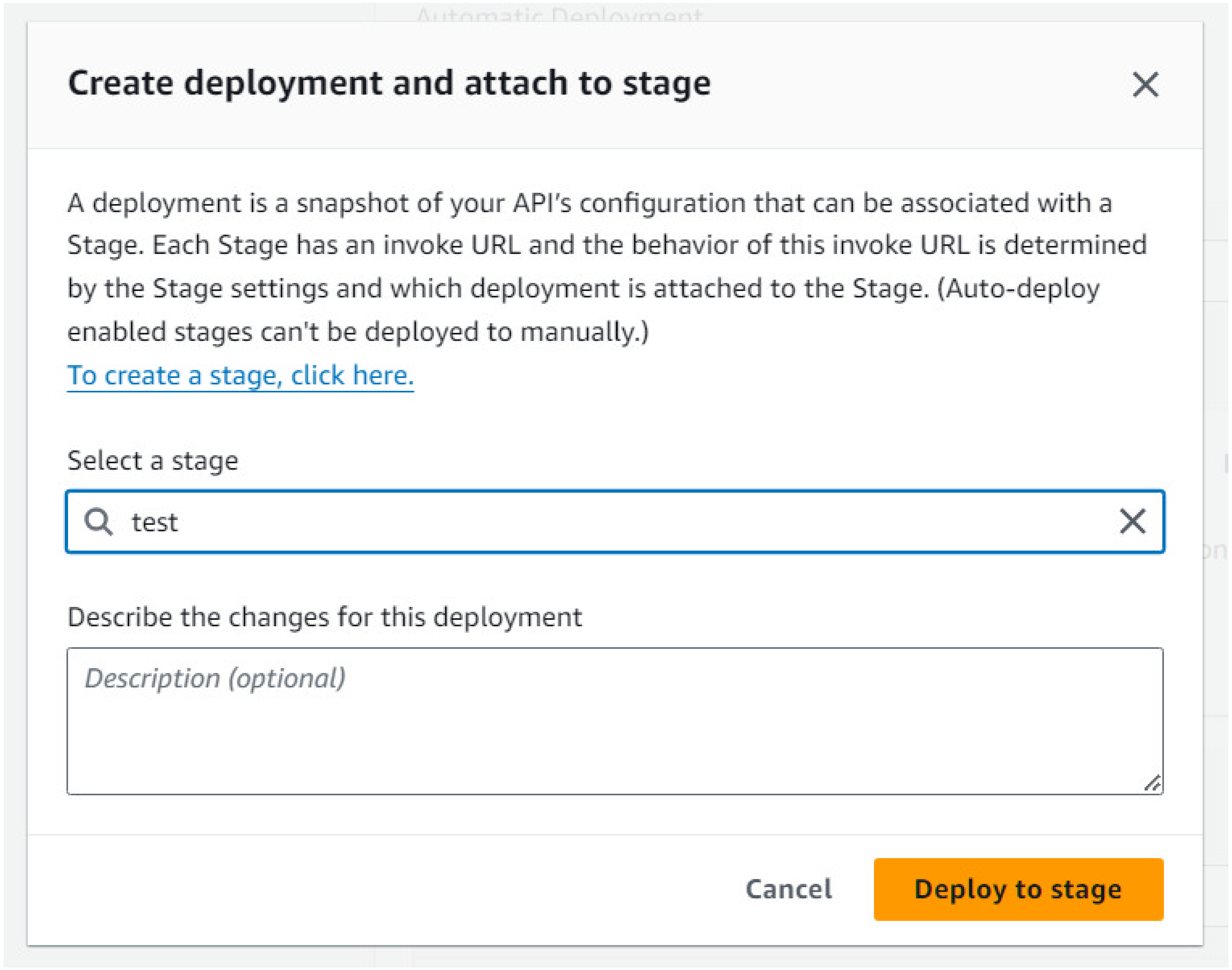
Details

Name	Created	Last updated
test	March 5, 2024 10:36 PM	March 5, 2024 10:36 PM

Invoke URL
https://6jj9jei6wc.execute-api.us-east-1.amazonaws.com/test

Description
None

Repeat the deploy to the just named stage (for example “test”).



Create deployment and attach to stage

A deployment is a snapshot of your API’s configuration that can be associated with a Stage. Each Stage has an invoke URL and the behavior of this invoke URL is determined by the Stage settings and which deployment is attached to the Stage. (Auto-deploy enabled stages can't be deployed to manually.)

To create a stage, click here.

Select a stage: test

Describe the changes for this deployment: Description (optional)

Make sure the request URL matches the API route

https://6jj9jei6wc.execute-api.us-east-1.amazonaws.com/test/ [CORRECT]

is different from

https://6jj9jei6wc.execute-api.us-east-1.amazonaws.com/test [WRONG]

For query_string, don't forget the proper format (an example follows):

https://<your id>.execute-api.us-east-1.amazonaws.com/<your stage name>/?city=Lisbon