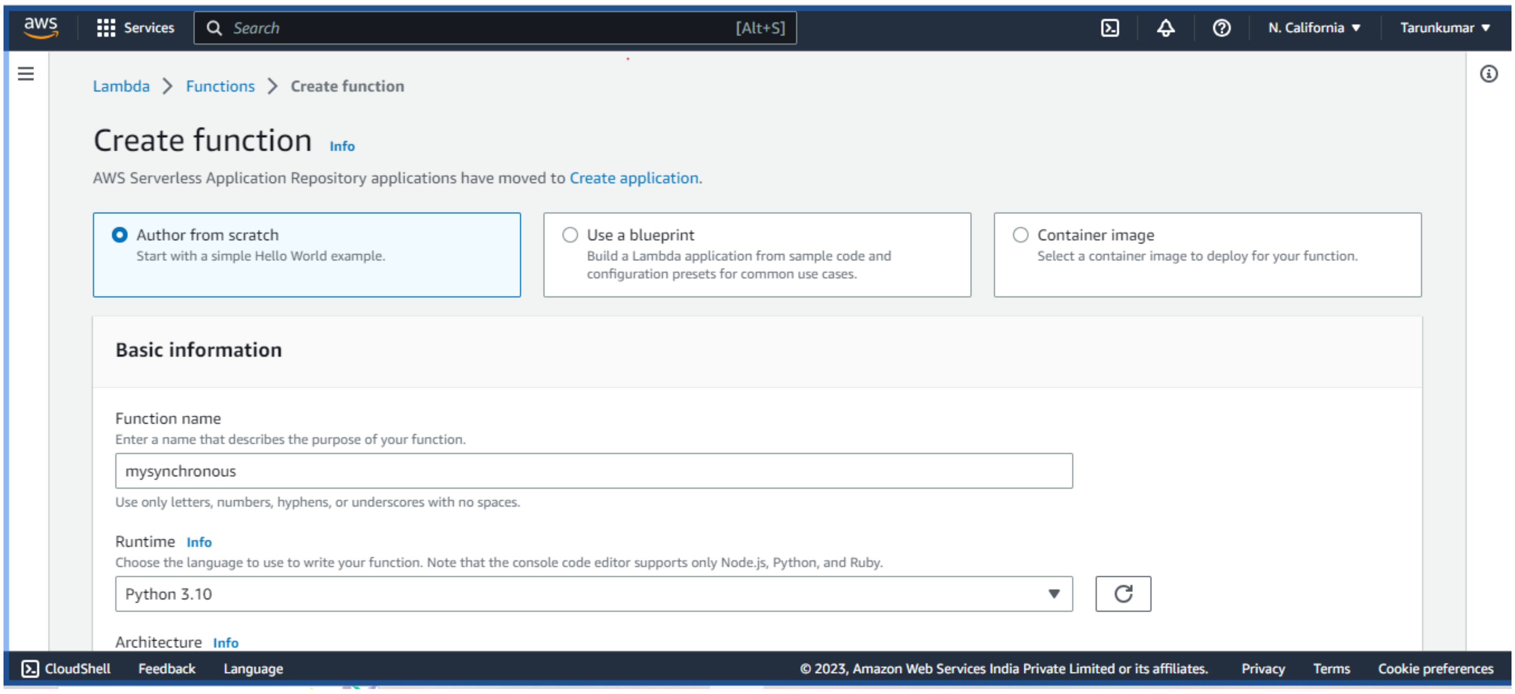
SYNCHRONOUS INVOCATION

Go to lambda from the console. Create a lambda function



Lambda > Functions > Create function

Create function

AWS Serverless Application Repository applications have moved to Create application.

- (●) 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

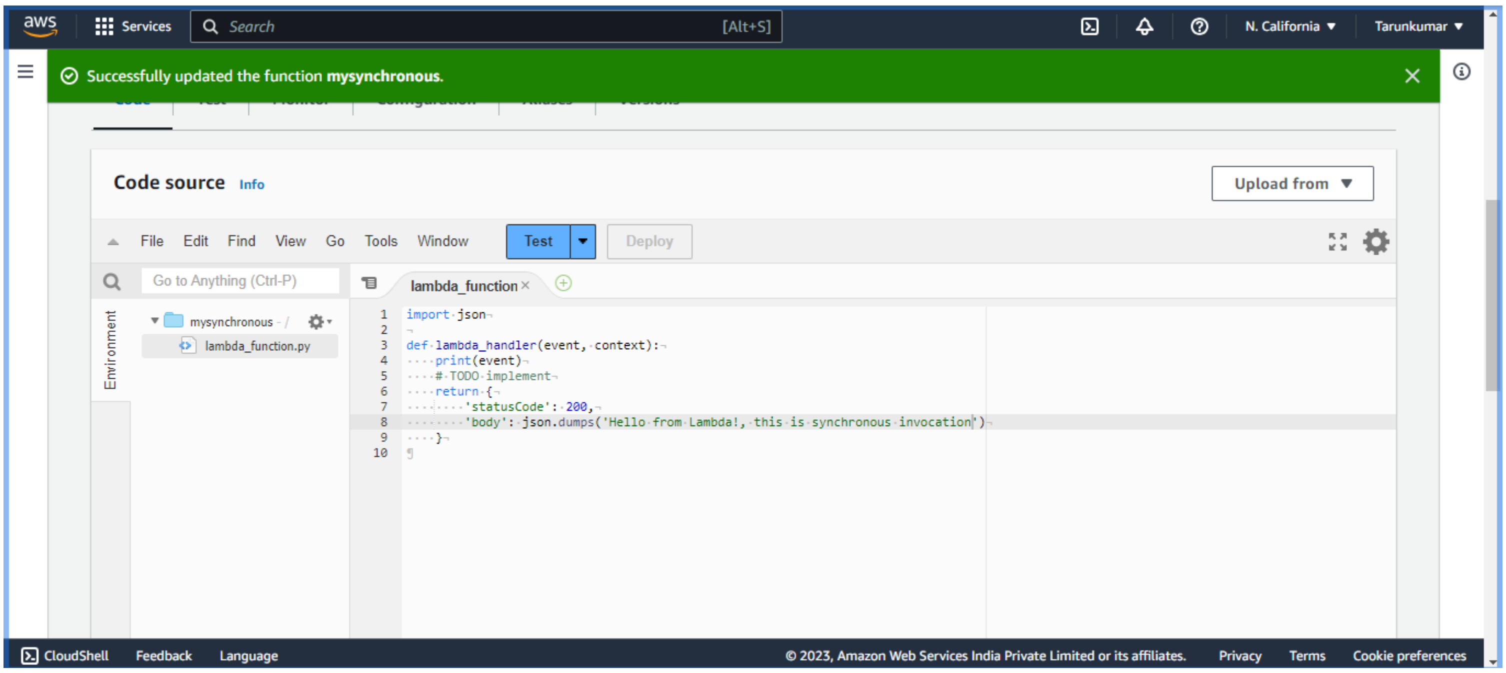
Function name
Enter a name that describes the purpose of your function.
mysynchronous
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.10

Architecture

© 2023, Amazon Web Services India Private Limited or its affiliates.

Select python 3.10 runtime environment. Deploy the code and test



Successfully updated the function **mysynchronous**.

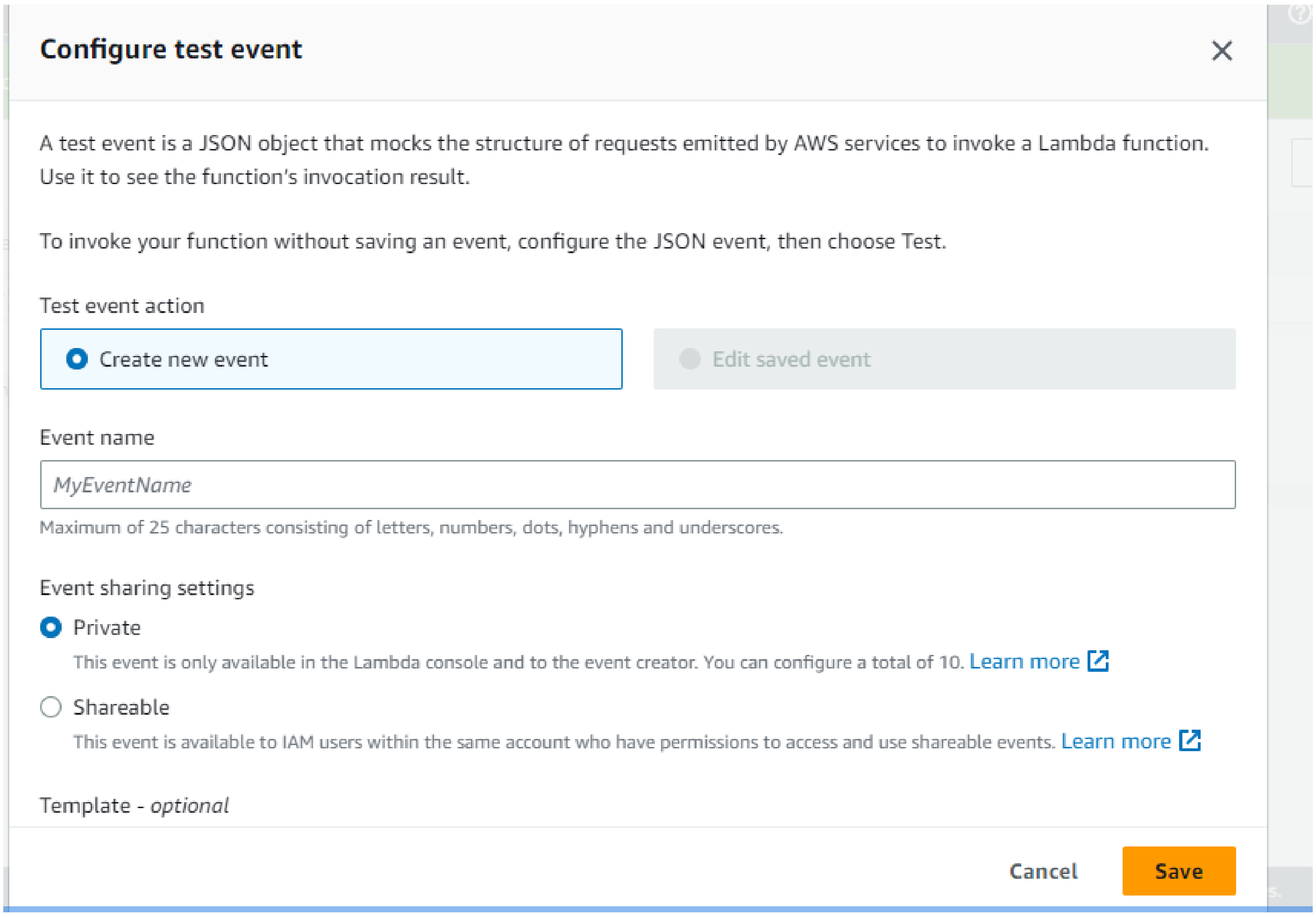
Code source

```python
import json

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

© 2023, Amazon Web Services India Private Limited or its affiliates.

Configure test event for testing the code



Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

[✓ Create new event] Edit saved event

Event name

MyEventName

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

[✓ Private]

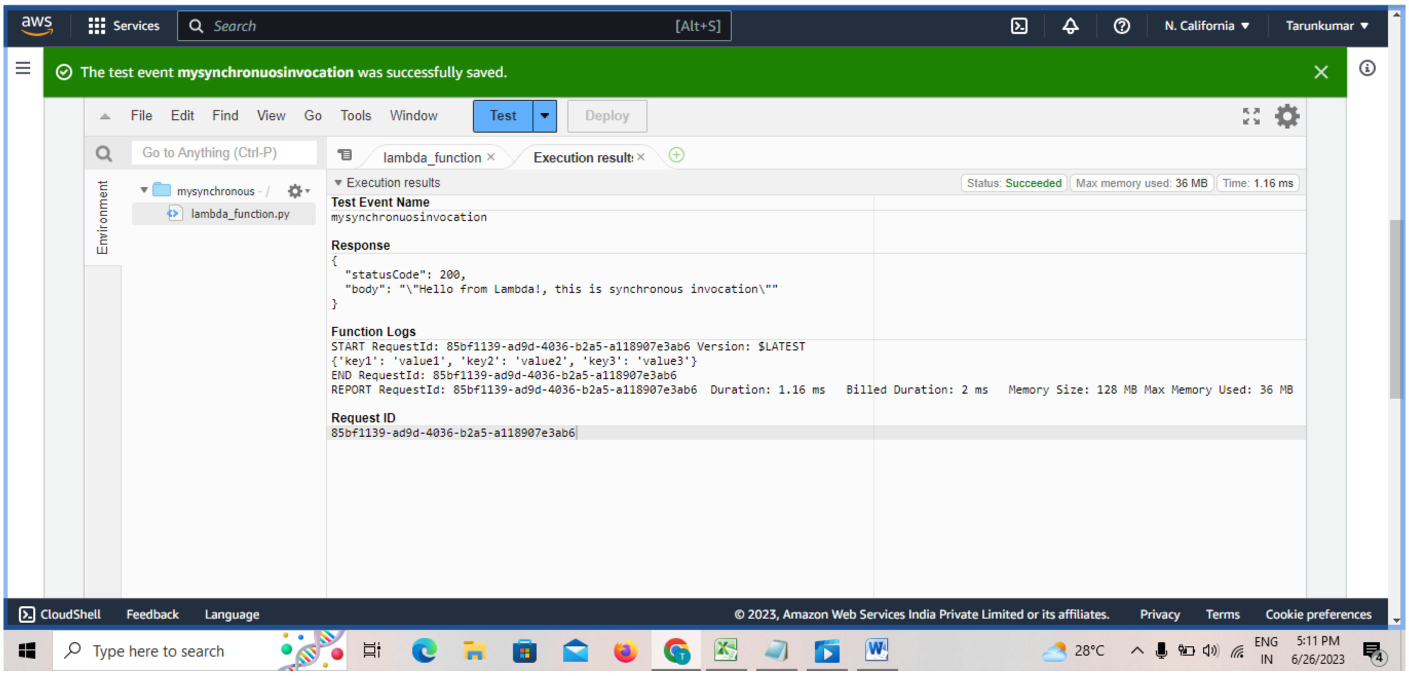
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more

Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more

Template - *optional*

Test is successful.



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

Execution results   Status: Succeeded   Max memory used: 36 MB   Time: 1.16 ms

Test Event Name

```
mysynchronuosinvocation
```

Response

```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!, this is synchronous invocation\""
}
```

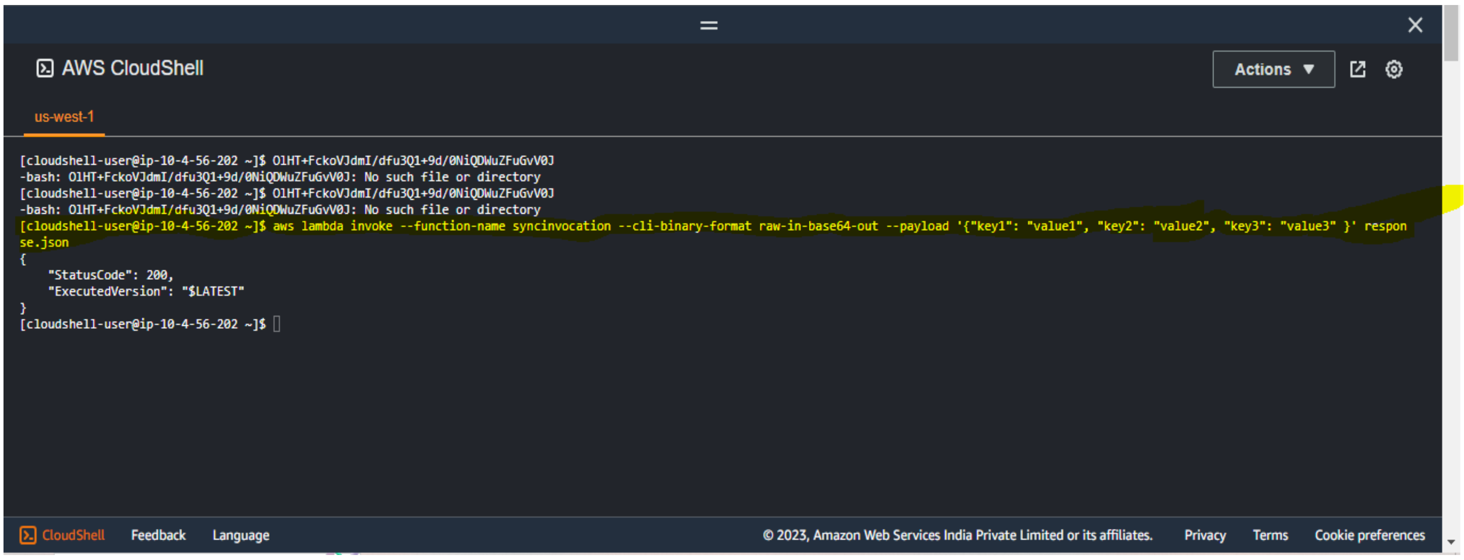
Function Logs

```
START RequestId: 85bf1139-ad9d-4036-b2a5-a118907e3ab6 Version: $LATEST
{'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
END RequestId: 85bf1139-ad9d-4036-b2a5-a118907e3ab6
REPORT RequestId: 85bf1139-ad9d-4036-b2a5-a118907e3ab6  Duration: 1.16 ms   Billed Duration: 2 ms   Memory Size: 128 MB Max Memory Used: 36 MB
```

Request ID

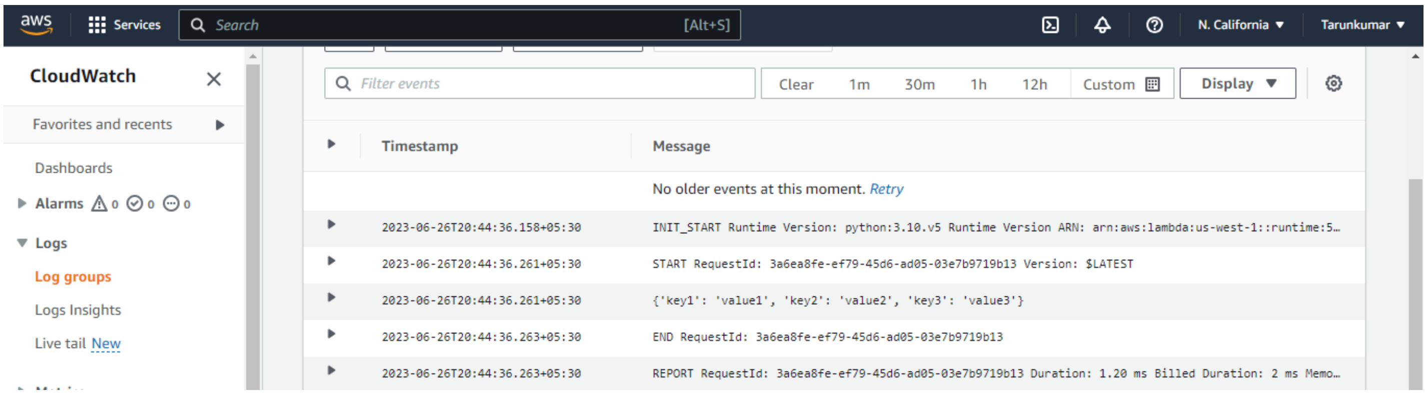
```
85bf1139-ad9d-4036-b2a5-a118907e3ab6
```

Open cloudshell and trigger the lambda. The below highlighted code is used to trigger the lambda function.



```
[cloudshell-user@ip-10-4-56-202 ~]$ OlHT+FckoVJdmI/dfu3Q1+9d/0NiQDWuZFuGvV0J
-bash: OlHT+FckoVJdmI/dfu3Q1+9d/0NiQDWuZFuGvV0J: No such file or directory
[cloudshell-user@ip-10-4-56-202 ~]$ OlHT+FckoVJdmI/dfu3Q1+9d/0NiQDWuZFuGvV0J
-bash: OlHT+FckoVJdmI/dfu3Q1+9d/0NiQDWuZFuGvV0J: No such file or directory
[cloudshell-user@ip-10-4-56-202 ~]$ aws lambda invoke --function-name syncinvocation --cli-binary-format raw-in-base64-out --payload '{"key1": "value1", "key2": "value2", "key3": "value3" }' respon
se.json
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
[cloudshell-user@ip-10-4-56-202 ~]$
```

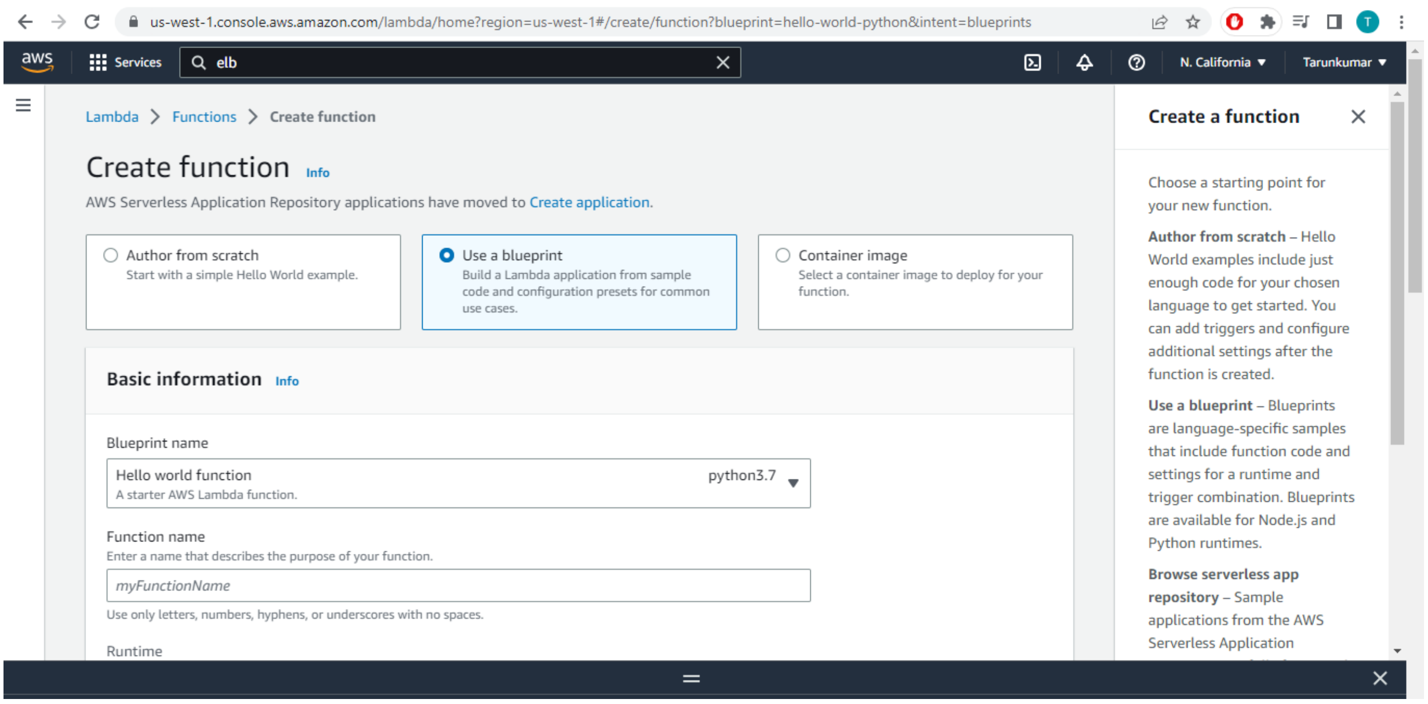
Now the lambda has started running. The trigger logs can be monitored using cloudwatch



```
Timestamp                        Message
                                 No older events at this moment. Retry
2023-06-26T20:44:36.158+05:30    INIT_START Runtime Version: python:3.10.v5 Runtime Version ARN: arn:aws:lambda:us-west-1::runtime:5…
2023-06-26T20:44:36.261+05:30    START RequestId: 3a6ea8fe-ef79-45d6-ad05-03e7b9719b13 Version: $LATEST
2023-06-26T20:44:36.261+05:30    {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
2023-06-26T20:44:36.263+05:30    END RequestId: 3a6ea8fe-ef79-45d6-ad05-03e7b9719b13
2023-06-26T20:44:36.263+05:30    REPORT RequestId: 3a6ea8fe-ef79-45d6-ad05-03e7b9719b13 Duration: 1.20 ms Billed Duration: 2 ms Memo…
```

ASYNCHRONOUS INVOCATION

Create a lambda function. Use a sample code



us-west-1.console.aws.amazon.com/lambda/home?region=us-west-1#/create/function?blueprint=hello-world-python&intent=blueprints

Lambda > Functions > Create function

Create function Info

AWS Serverless Application Repository applications have moved to Create application.

- 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 Info

Blueprint name: Hello world function — A starter AWS Lambda function. — python3.7

Function name: Enter a name that describes the purpose of your function. myFunctionName

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime

Create a function

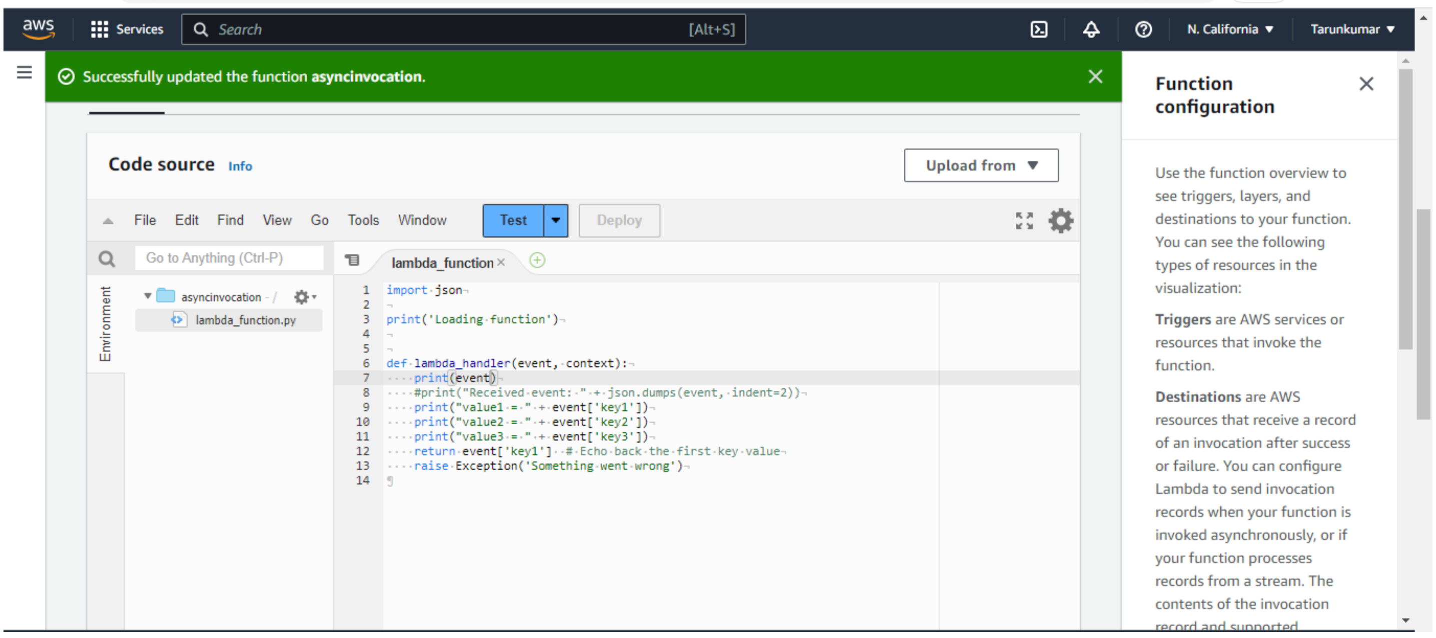
Choose a starting point for your new function.

Author from scratch – Hello World examples include just enough code for your chosen language to get started. You can add triggers and configure additional settings after the function is created.

Use a blueprint – Blueprints are language-specific samples that include function code and settings for a runtime and trigger combination. Blueprints are available for Node.js and Python runtimes.

Browse serverless app repository – Sample applications from the AWS Serverless Application

Deploy and test the following code



Successfully updated the function **asyncinvocation**.

Code source Info

Environment: asyncinvocation / lambda_function.py

```python
import json

print('Loading function')


def lambda_handler(event, context):
    print(event)
    #print("Received event: " + json.dumps(event, indent=2))
    print("value1 = " + event['key1'])
    print("value2 = " + event['key2'])
    print("value3 = " + event['key3'])
    return event['key1']  # Echo back the first key value
    raise Exception('Something went wrong')
```

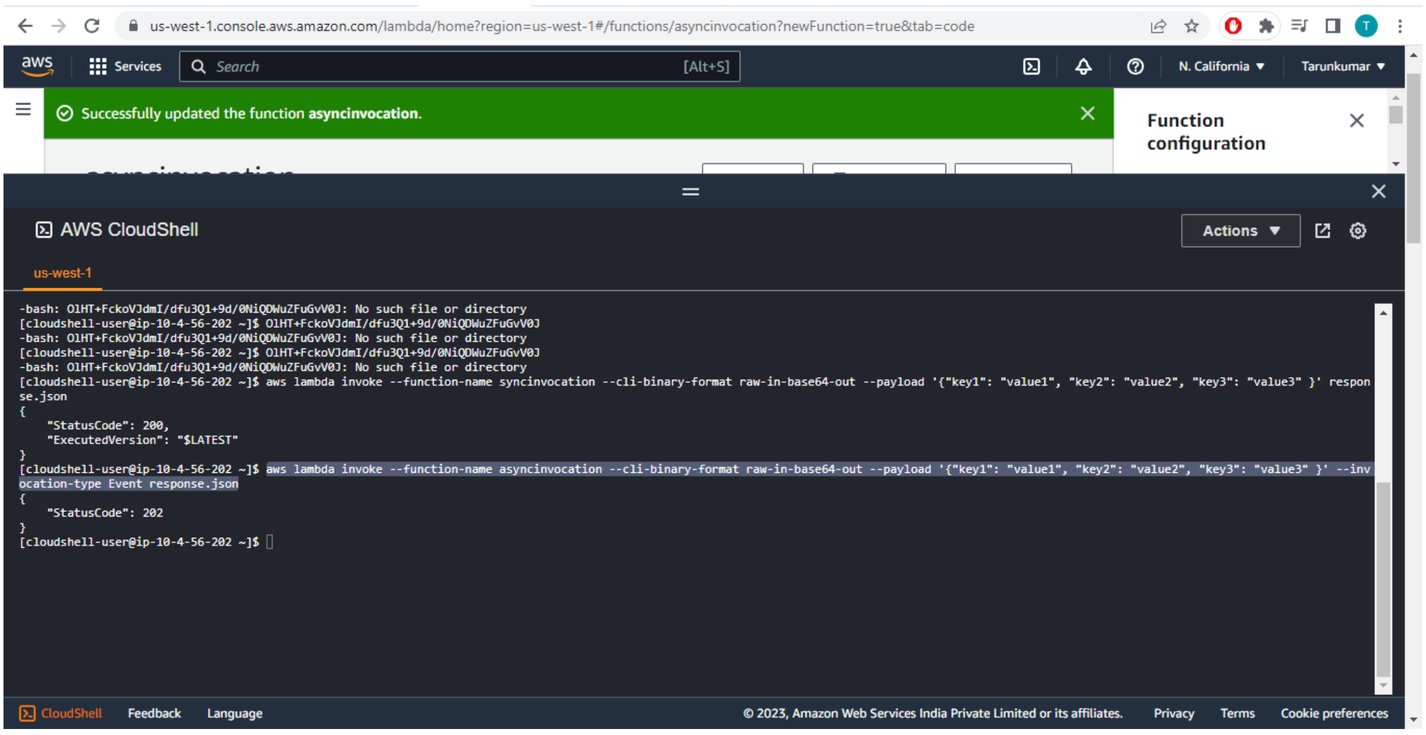
Function configuration

Use the function overview to see triggers, layers, and destinations to your function. You can see the following types of resources in the visualization:

Triggers are AWS services or resources that invoke the function.

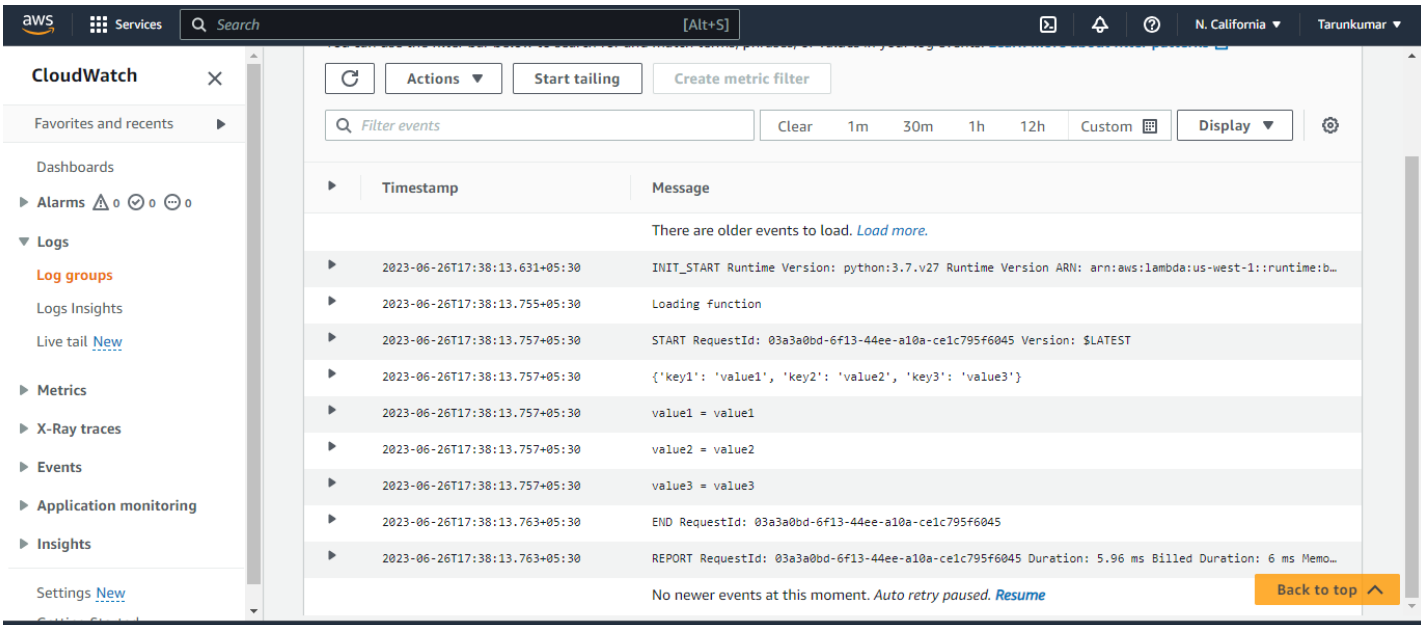
Destinations are AWS resources that receive a record of an invocation after success or failure. You can configure Lambda to send invocation records when your function is invoked asynchronously, or if your function processes records from a stream. The contents of the invocation record and supported

Invoke the lambda function through cloudshell with the help of the following command- (aws lambda invoke --function-name asyncinvocation --cli-binary-format raw-in-base64-out --payload '{"key1": "value1", "key2": "value2", "key3": "value3" }' --invocation-type Event response.json)


us-west-1.console.aws.amazon.com/lambda/home?region=us-west-1#/functions/asyncinvocation?newFunction=true&tab=code
Services
Search
[Alt+S]
N. California
Tarunkumar
Successfully updated the function asyncinvocation.
Function configuration
AWS CloudShell
Actions
us-west-1
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

```
-bash: OlHT+FckoVJdmI/dfu3Q1+9d/0NiQDWuZFuGvV0J: No such file or directory
[cloudshell-user@ip-10-4-56-202 ~]$ OlHT+FckoVJdmI/dfu3Q1+9d/0NiQDWuZFuGvV0J
-bash: OlHT+FckoVJdmI/dfu3Q1+9d/0NiQDWuZFuGvV0J: No such file or directory
[cloudshell-user@ip-10-4-56-202 ~]$ OlHT+FckoVJdmI/dfu3Q1+9d/0NiQDWuZFuGvV0J
-bash: OlHT+FckoVJdmI/dfu3Q1+9d/0NiQDWuZFuGvV0J: No such file or directory
[cloudshell-user@ip-10-4-56-202 ~]$ aws lambda invoke --function-name syncinvocation --cli-binary-format raw-in-base64-out --payload '{"key1": "value1", "key2": "value2", "key3": "value3" }' respon
se.json
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
[cloudshell-user@ip-10-4-56-202 ~]$ aws lambda invoke --function-name asyncinvocation --cli-binary-format raw-in-base64-out --payload '{"key1": "value1", "key2": "value2", "key3": "value3" }' --inv
ocation-type Event response.json
{
    "StatusCode": 202
}
[cloudshell-user@ip-10-4-56-202 ~]$ 
```

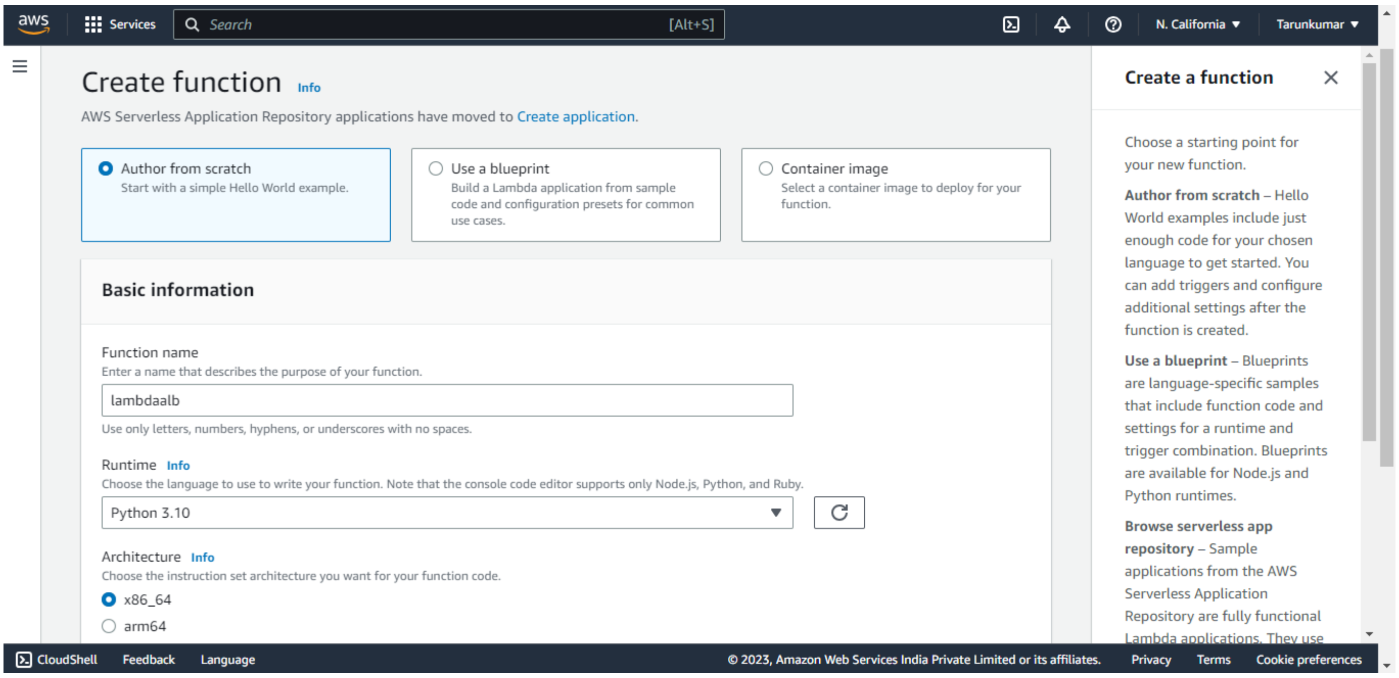
The logs are recorded in cloudwatch logs.



```
Timestamp                       Message
                                There are older events to load. Load more.
2023-06-26T17:38:13.631+05:30   INIT_START Runtime Version: python:3.7.v27 Runtime Version ARN: arn:aws:lambda:us-west-1::runtime:b…
2023-06-26T17:38:13.755+05:30   Loading function
2023-06-26T17:38:13.757+05:30   START RequestId: 03a3a0bd-6f13-44ee-a10a-ce1c795f6045 Version: $LATEST
2023-06-26T17:38:13.757+05:30   {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
2023-06-26T17:38:13.757+05:30   value1 = value1
2023-06-26T17:38:13.757+05:30   value2 = value2
2023-06-26T17:38:13.757+05:30   value3 = value3
2023-06-26T17:38:13.763+05:30   END RequestId: 03a3a0bd-6f13-44ee-a10a-ce1c795f6045
2023-06-26T17:38:13.763+05:30   REPORT RequestId: 03a3a0bd-6f13-44ee-a10a-ce1c795f6045 Duration: 5.96 ms Billed Duration: 6 ms Memo…
                                No newer events at this moment. Auto retry paused. Resume
```

LAMBDA WITH ALB

Create a lambda function



Create function Info

AWS Serverless Application Repository applications have moved to Create application.

[✓ 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.
lambdaalb
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 3.10

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64] arm64

Create a function

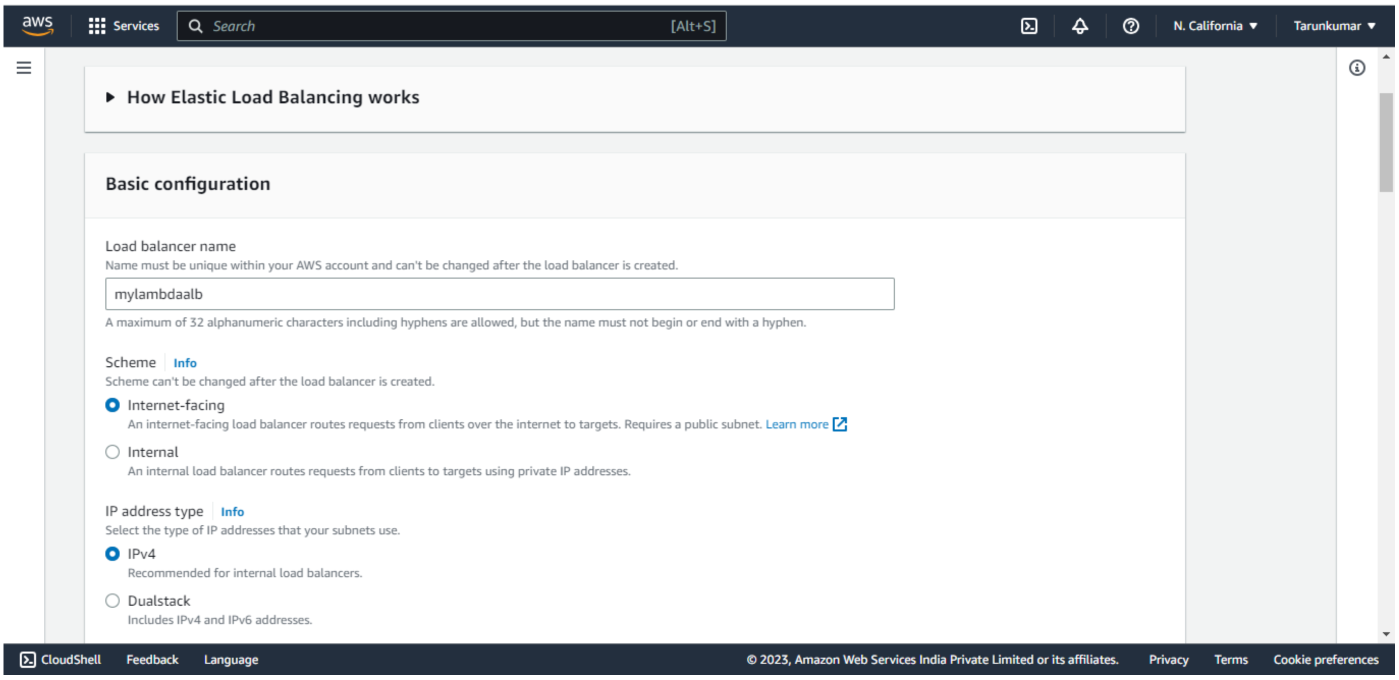
Choose a starting point for your new function.

Author from scratch – Hello World examples include just enough code for your chosen language to get started. You can add triggers and configure additional settings after the function is created.

Use a blueprint – Blueprints are language-specific samples that include function code and settings for a runtime and trigger combination. Blueprints are available for Node.js and Python runtimes.

Browse serverless app repository – Sample applications from the AWS Serverless Application Repository are fully functional Lambda applications. They use

Create an application load balancer.



How Elastic Load Balancing works

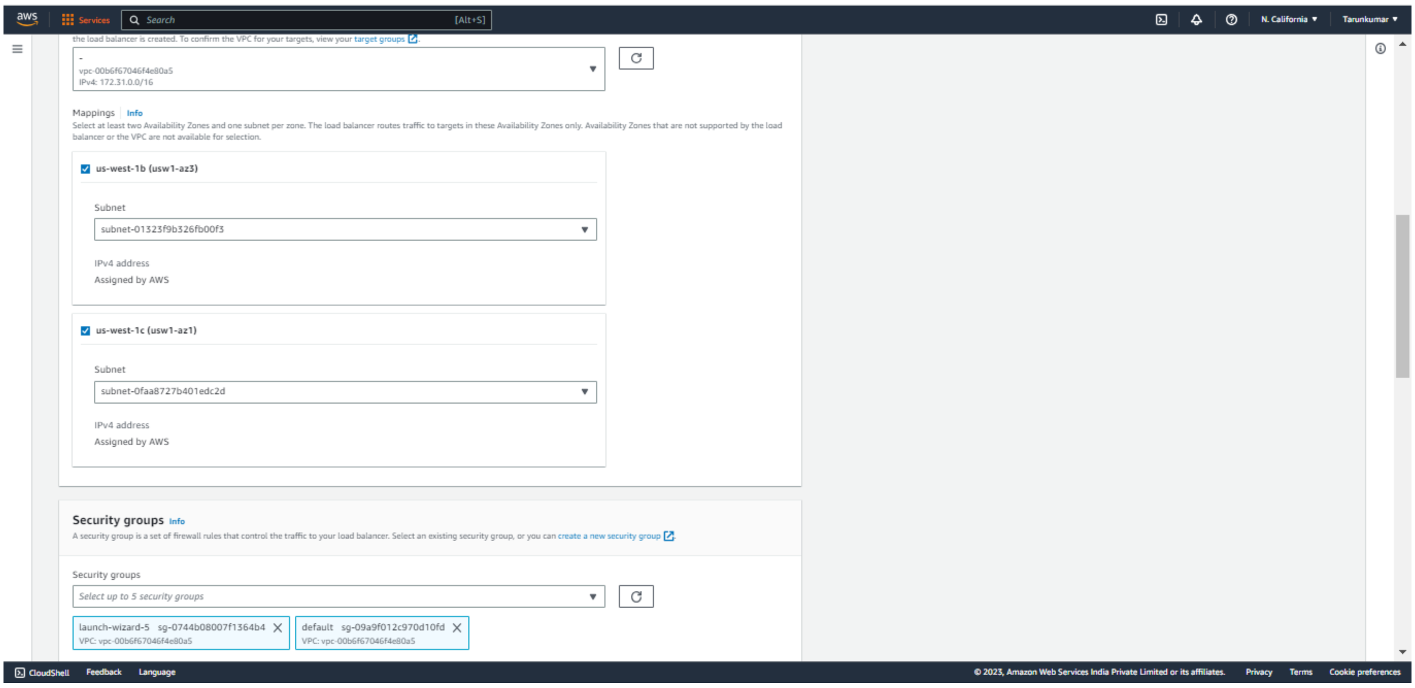
Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.
mylambdaalb
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme Info
Scheme can't be changed after the load balancer is created.
[✓ Internet-facing] An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more
Internal — An internal load balancer routes requests from clients using private IP addresses.

IP address type Info
Select the type of IP addresses that your subnets use.
[✓ IPv4] Recommended for internal load balancers.
Dualstack — Includes IPv4 and IPv6 addresses.

Make sure that the subnet is public and security group has inbound traffic open for HTTP.



the load balancer is created. To confirm the VPC for your targets, view your target groups.

-
vpc-00b6f67046f4e80a5
IPv4: 172.31.0.0/16

Mappings Info
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

✓ us-west-1b (usw1-az3)

Subnet
subnet-01323f9b326fb00f3

IPv4 address
Assigned by AWS

✓ us-west-1c (usw1-az1)

Subnet
subnet-0faa8727b401edc2d

IPv4 address
Assigned by AWS

Security groups Info
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group.

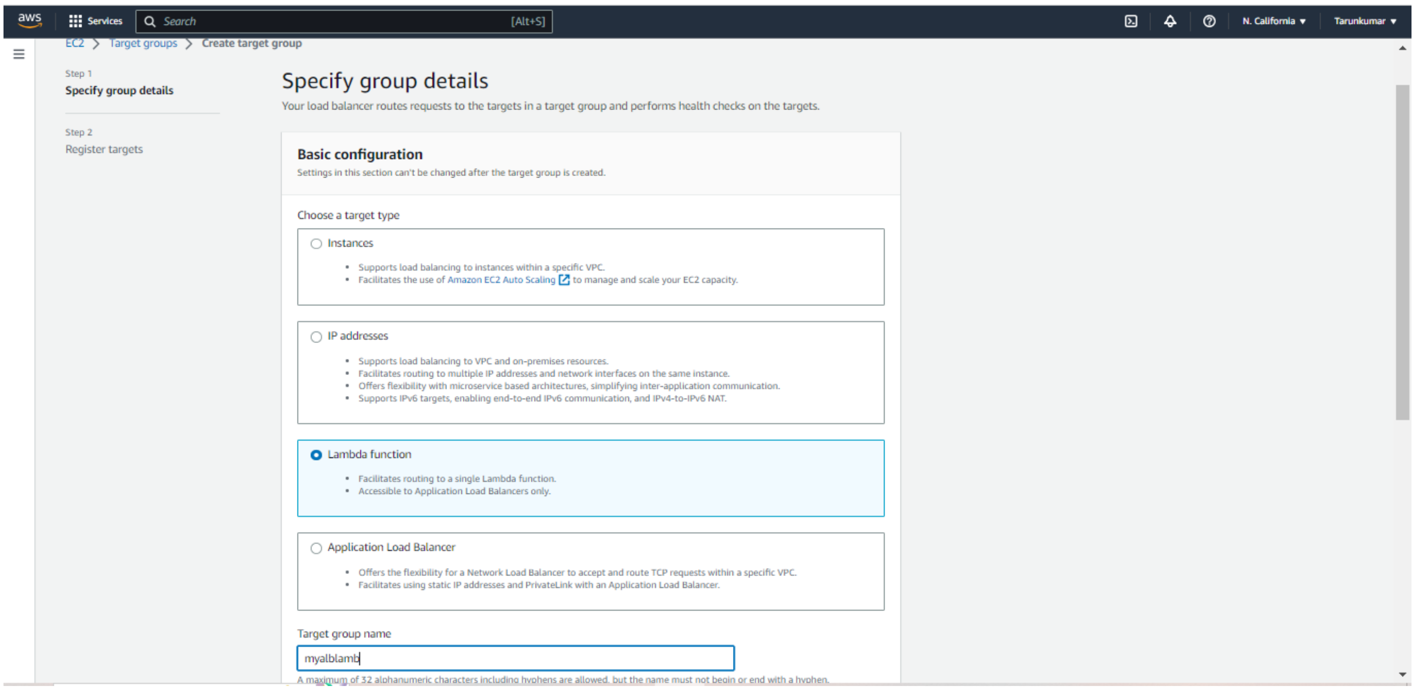
Security groups
Select up to 5 security groups

launch-wizard-5 sg-0744b08007f1364b4
VPC: vpc-00b6f67046f4e80a5

default sg-09a9f012c970d10fd
VPC: vpc-00b6f67046f4e80a5

CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

Create a target group for the application load balancer and add lambda function which was created in the first step (with HTTP(Port-80) protocol) as target.



EC2 > Target groups > Create target group

Step 1
Specify group details

Step 2
Register targets

Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

Basic configuration
Settings in this section can't be changed after the target group is created.

Choose a target type

○ Instances
- Supports load balancing to instances within a specific VPC.
- Facilitates the use of Amazon EC2 Auto Scaling to manage and scale your EC2 capacity.

○ IP addresses
- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

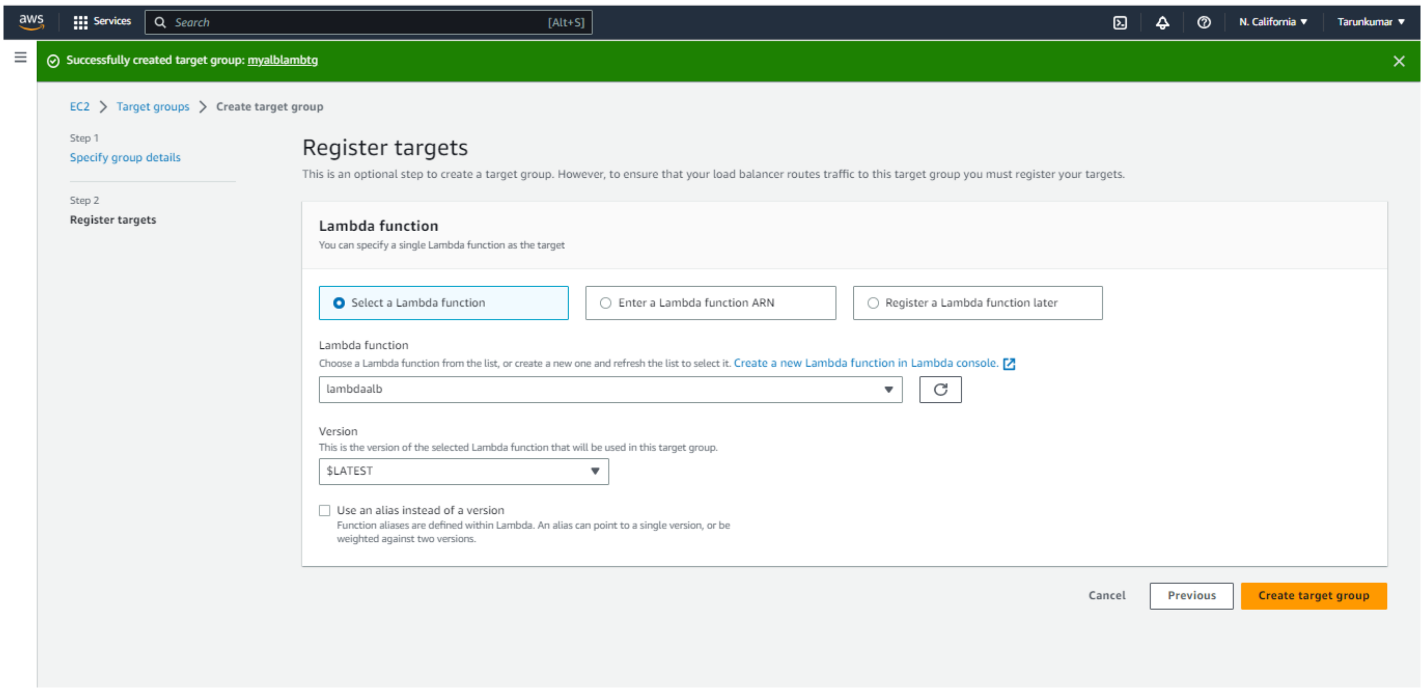
● Lambda function
- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

○ Application Load Balancer
- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Target group name
myalblamb

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Select the lambda function created for ALB


Successfully created target group: myalblambtg

EC2 > Target groups > Create target group

Step 1
Specify group details

Step 2
Register targets

Register targets

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Lambda function

You can specify a single Lambda function as the target

● Select a Lambda function ○ Enter a Lambda function ARN ○ Register a Lambda function later

Lambda function

Choose a Lambda function from the list, or create a new one and refresh the list to select it. Create a new Lambda function in Lambda console.

lambdaalb

Version

This is the version of the selected Lambda function that will be used in this target group.

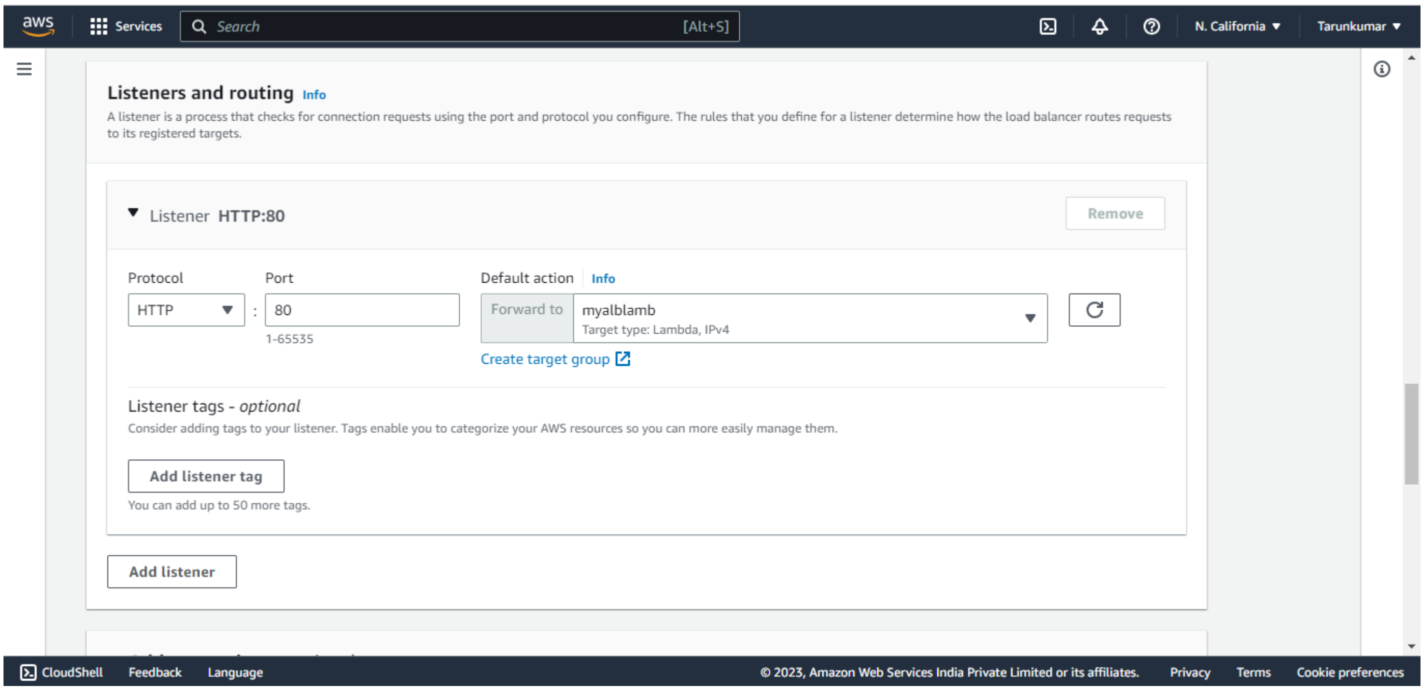
$LATEST

☐ Use an alias instead of a version

Function aliases are defined within Lambda. An alias can point to a single version, or be weighted against two versions.

Cancel | Previous | Create target group

Attach the target group and create the load balancer,


Listeners and routing Info

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

Listener HTTP:80

Protocol: HTTP

Port: 80 (1-65535)

Default action Info

Forward to myalblamb
Target type: Lambda, IPv4

Create target group

Listener tags - *optional*

Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

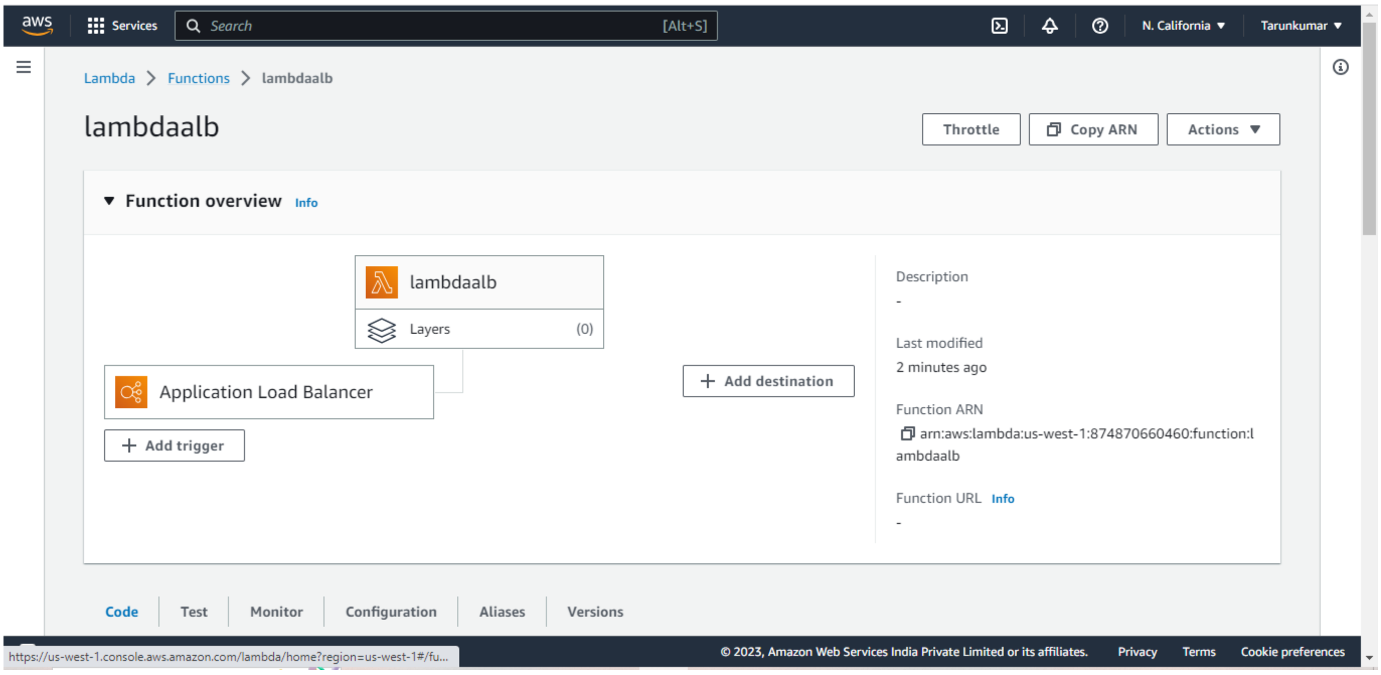
Add listener tag

You can add up to 50 more tags.

Add listener

© 2023, Amazon Web Services India Private Limited or its affiliates.

Now Application load balancer is added as the trigger to the lambda function


Lambda > Functions > lambdaalb

lambdaalb

Function overview

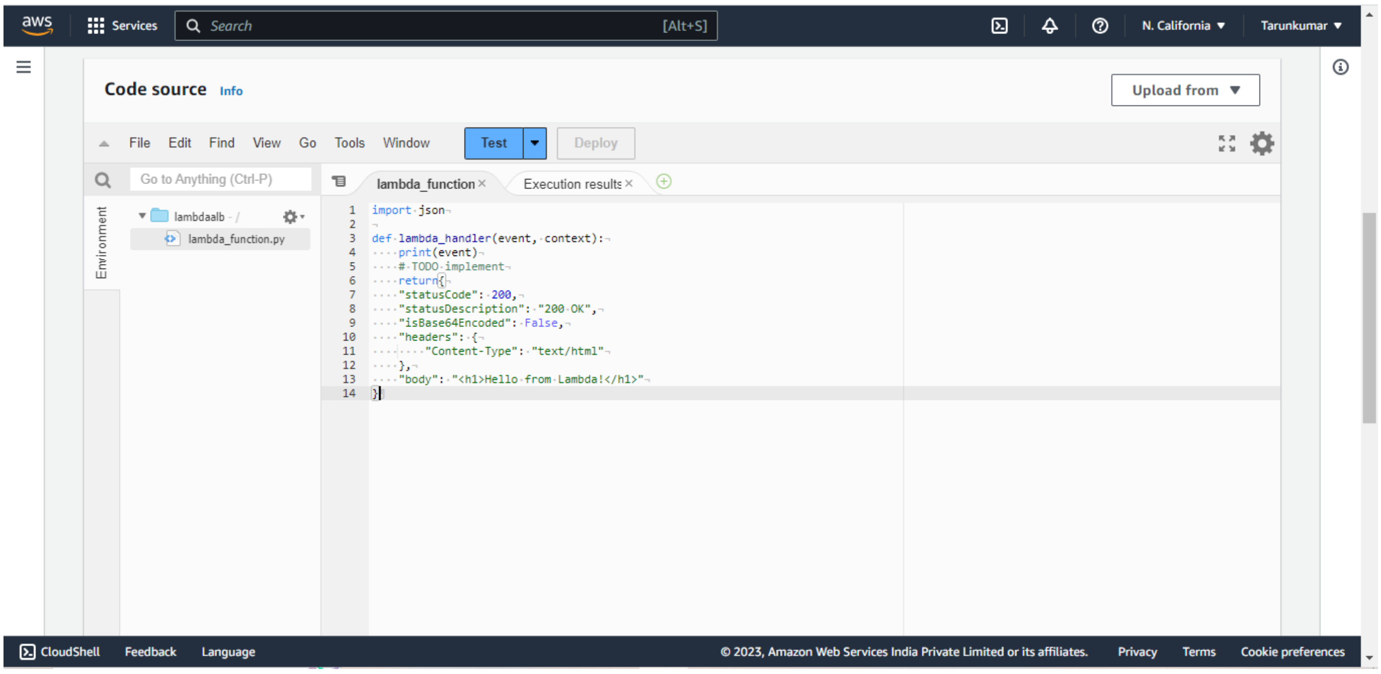
Description
-

Last modified
2 minutes ago

Function ARN
arn:aws:lambda:us-west-1:874870660460:function:lambdaalb

Function URL
-

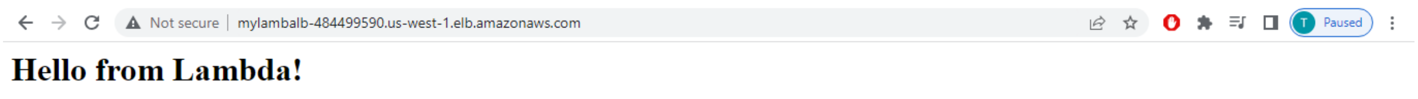
Edit the code of lambda function to view in HTML format when the DNS of ALB is clicked.


Code source

```
import json

def lambda_handler(event, context):
    print(event)
    # TODO implement
    return{
    "statusCode": 200,
    "statusDescription": "200 OK",
    "isBase64Encoded": False,
    "headers": {
        "Content-Type": "text/html"
    },
    "body": "<h1>Hello from Lambda!</h1>"
}
```

The application load balancer triggered the code written in lambda to run when the DNS of the alb was clicked.


Not secure | mylambalb-484499590.us-west-1.elb.amazonaws.com

Hello from Lambda!