

Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

Theory:

AWS Lambda and S3 Integration:

AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:

1. Create an S3 Bucket:

- First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

2. Create the Lambda Function:

- Set up a new Lambda function using AWS Lambda’s console. You can choose a runtime environment like Python, Node.js, or Java.
- Write code that logs a message like “An Image has been added” when triggered.

3. Set Up Permissions:

- Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

4. Configure S3 Trigger:

- Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

5. Test the Setup:

- Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

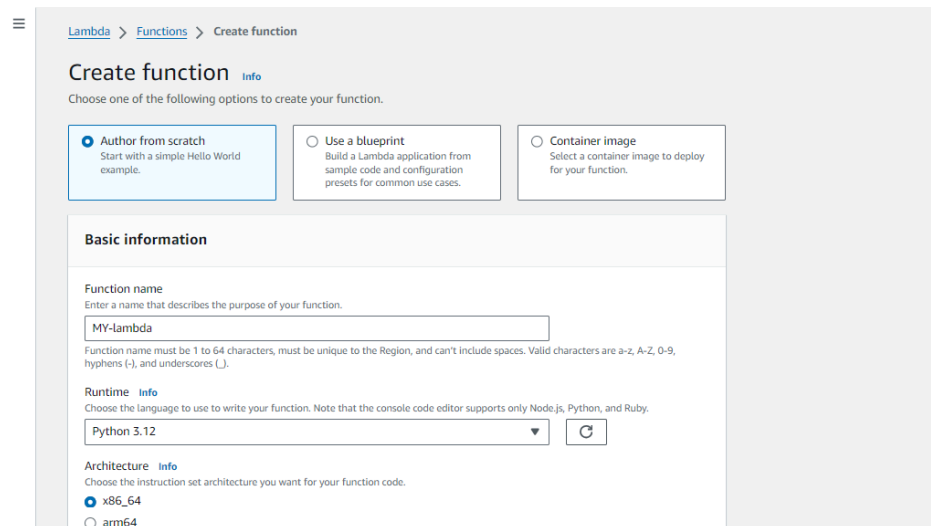
Steps To create the lambda function:

Step 1: Login to your AWS Personal account. Now open S3 from services and click on create S3 bucket.

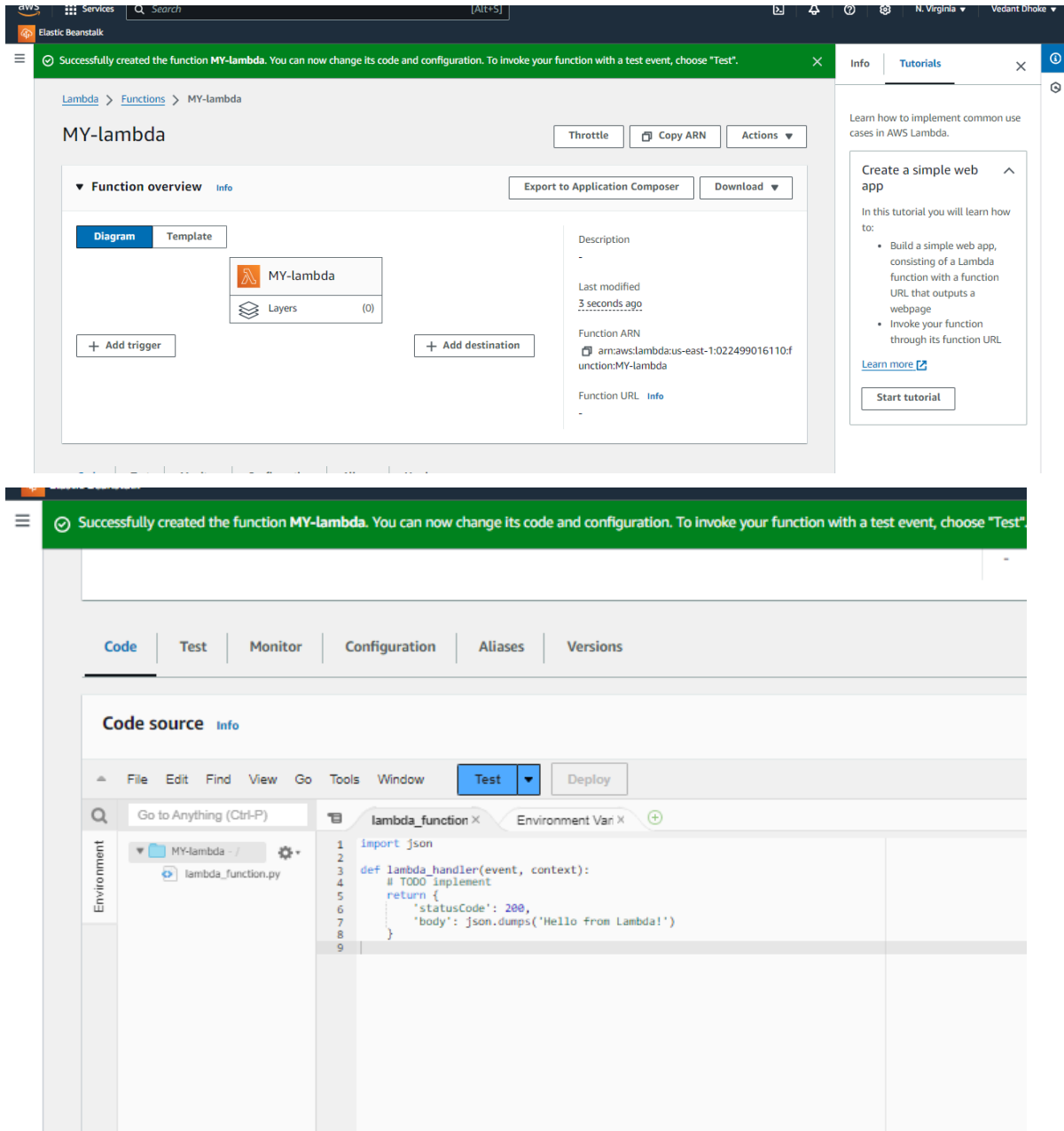
Step 2: I have used already created bucket v2bucket if you dont have then you can create a basic bucket for this experiment .

Step 3: Open lambda console and click on create function button.

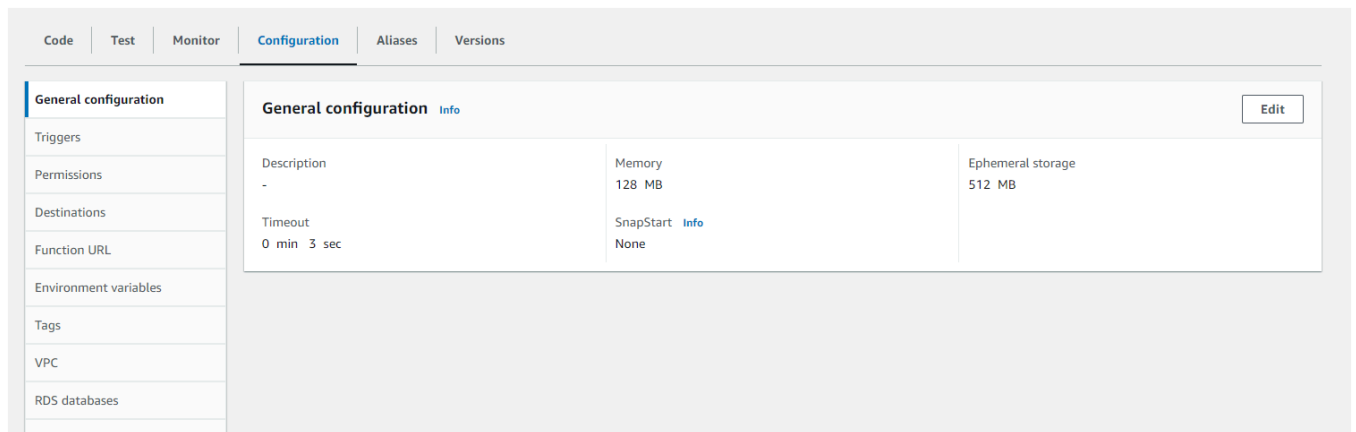
Step 4: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12 , Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.



The screenshot displays the AWS Lambda 'Create function' console. At the top, there's a breadcrumb trail: 'Lambda > Functions > Create function'. The main heading is 'Create function' with an 'Info' link. Below it, a note says 'Choose one of the following options to create your function.' There are three radio button options: 'Author from scratch' (selected, with a sub-note 'Start with a simple Hello World example.'), 'Use a blueprint' (with a sub-note 'Build a Lambda application from sample code and configuration presets for common use cases.'), and 'Container image' (with a sub-note 'Select a container image to deploy for your function.'). Below these is the 'Basic information' section. It contains a 'Function name' field with the value 'MY-lambda' and a note: 'Enter a name that describes the purpose of your function. 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 (_).'. There's also a 'Runtime' dropdown menu set to 'Python 3.12' with a refresh icon, and a note: 'Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.' Finally, there's an 'Architecture' dropdown menu set to 'x86_64' with a note: 'Choose the instruction set architecture you want for your function code.' and a sub-note 'x86_64'.



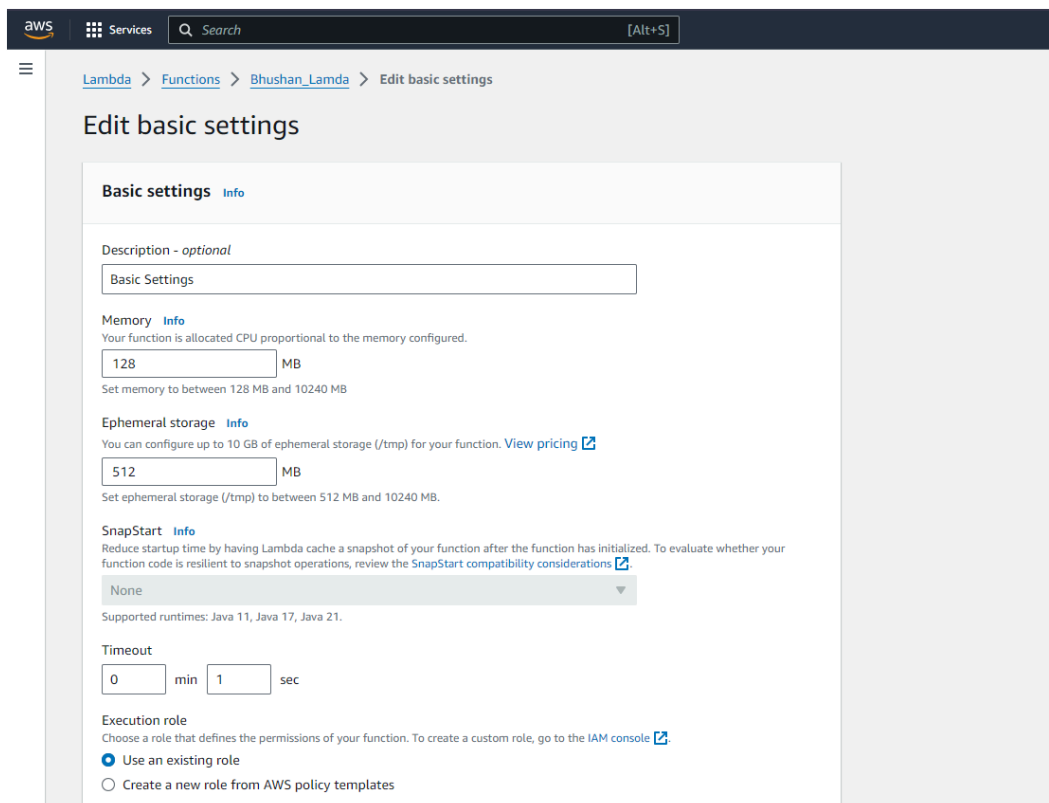
So See or Edit the basic settings go to configuration then click on edit general setting.



The screenshot shows the AWS Lambda console's 'Configuration' tab for a function. On the left is a sidebar with a list of configuration options: General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, and RDS databases. The 'General configuration' option is selected and highlighted. The main area displays the 'General configuration' details. It includes a description field with a hyphen, a memory setting of 128 MB, and an ephemeral storage setting of 512 MB. The timeout is set to 0 minutes and 3 seconds. There is an 'Edit' button in the top right corner of the configuration area.

General configuration		
Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart	
0 min 3 sec	None	

Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.



The screenshot shows the 'Edit basic settings' page in the AWS Lambda console. The breadcrumb trail indicates the path: Lambda > Functions > Bhushan_Lambda > Edit basic settings. The page title is 'Edit basic settings'. The 'Basic settings' section is active. It contains several fields: 'Description - optional' with the value 'Basic Settings'; 'Memory' set to 128 MB; 'Ephemeral storage' set to 512 MB; 'SnapStart' set to 'None'; and 'Timeout' set to 0 minutes and 1 second. The 'Execution role' section has two options: 'Use an existing role' (selected) and 'Create a new role from AWS policy templates'.

Basic settings

Description - optional
Basic Settings

Memory
Your function is allocated CPU proportional to the memory configured.
128 MB
Set memory to between 128 MB and 10240 MB

Ephemeral storage
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)
512 MB
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the [SnapStart compatibility considerations](#).
None

Supported runtimes: Java 11, Java 17, Java 21.

Timeout
0 min 1 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).
☒ Use an existing role
☐ Create a new role from AWS policy templates

Step 5: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select s3 put template.

Test event Info

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

event-exp_12

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

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

Template - optional

s3-put

Event JSON [Format JSON](#)

```
2  "Records": [  
3  {  
4    "eventVersion": "2.0",  
5    "eventSource": "aws:s3",  
6    "awsRegion": "us-east-1",  
7    "eventName": "ObjectCreated:Put",  
8    "userIdentity": {  
9      "principalId": "EXAMPLE"  
10   },  
11   "requestParameters": {  
12     "sourceIPAddress": "127.0.0.1"  
13   },  
14   "responseElements": {  
15     "x-amz-request-id": "EXAMPLE123456789",  
16     "x-amz-id-2": "EXAMPLE123/5678abcdeghijklmnopqrstuvwxyzABCDEFGHIH"  
17   },  
18   "s3": {  
19     "s3SchemaVersion": "1.0",  
20     "configurationId": "testconfigrule",  
21     "bucket": {  
22       "name": "example-bucket",  
23       "ownerIdentity": {  
24         "principalId": "EXAMPLE"  
25       },  
26       "arn": "arn:aws:s3::example-bucket"  
27     },  
28     "object": {  
29       "key": "testK2fkey",  
30       "size": 1024,  
31     }  
32   }  
33   }  
34 ]
```

1:1 JSON Spaces: 2

Tutorials

Learn how to implement common use cases in AWS Lambda.

Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

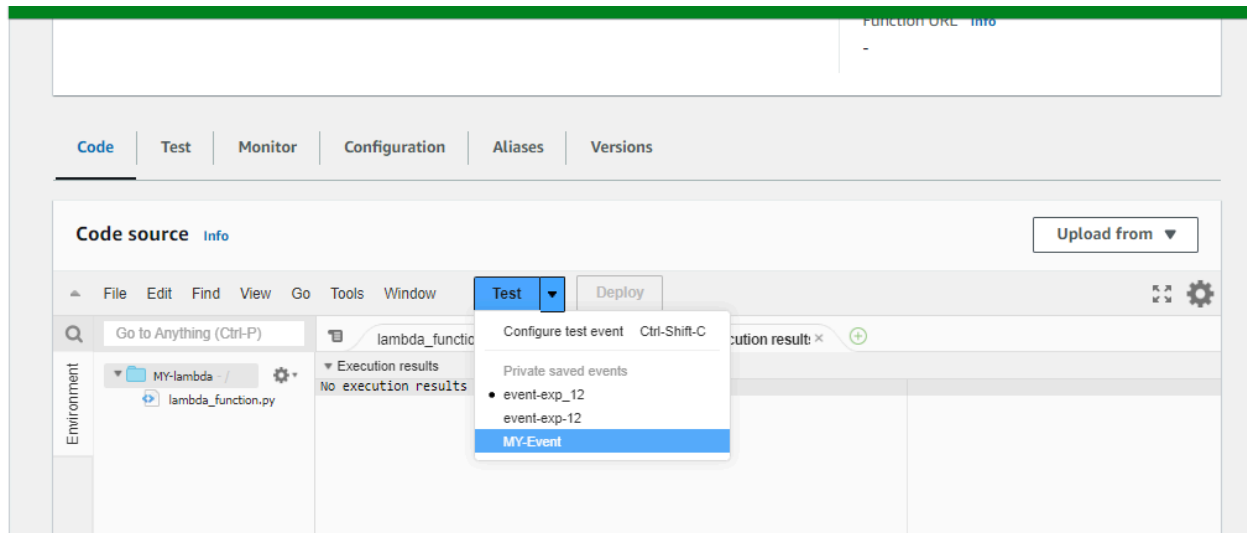
[Learn more](#)

[Start tutorial](#)

CloudShell Feedback

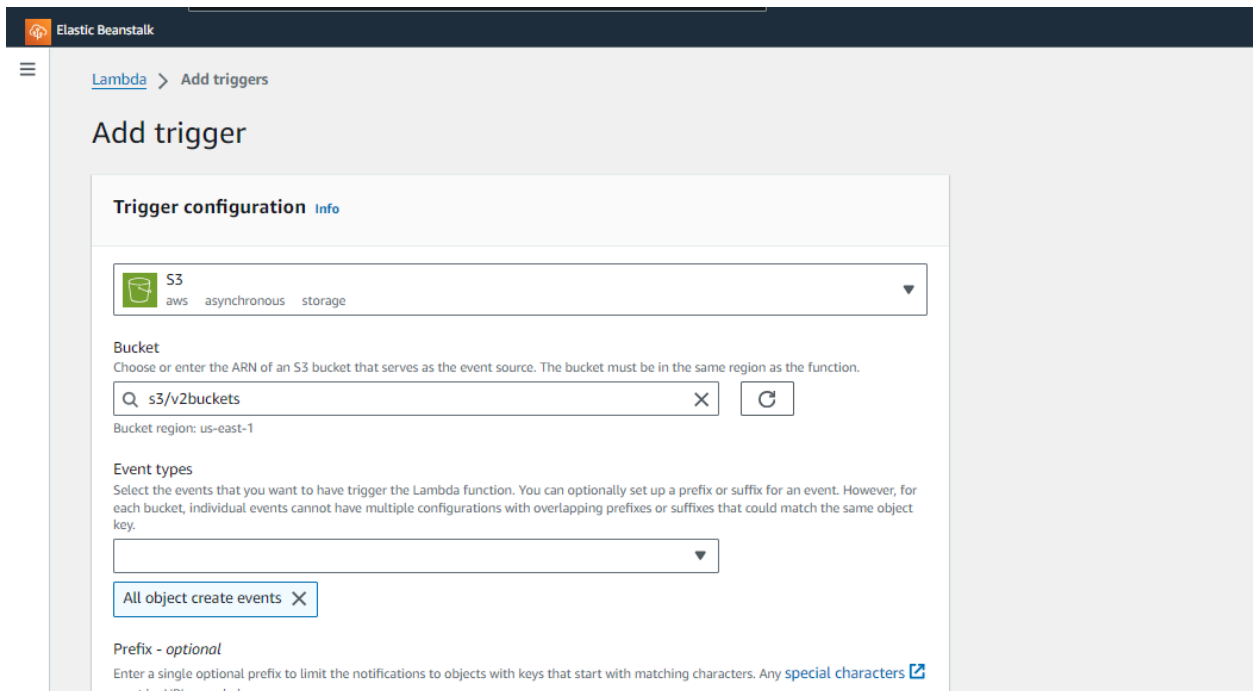
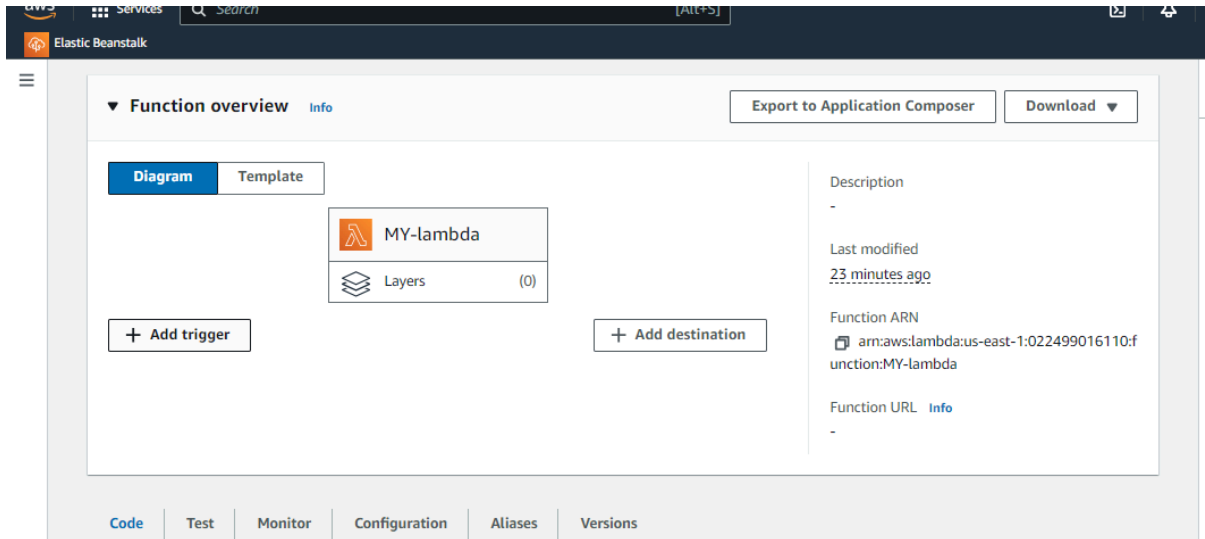
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

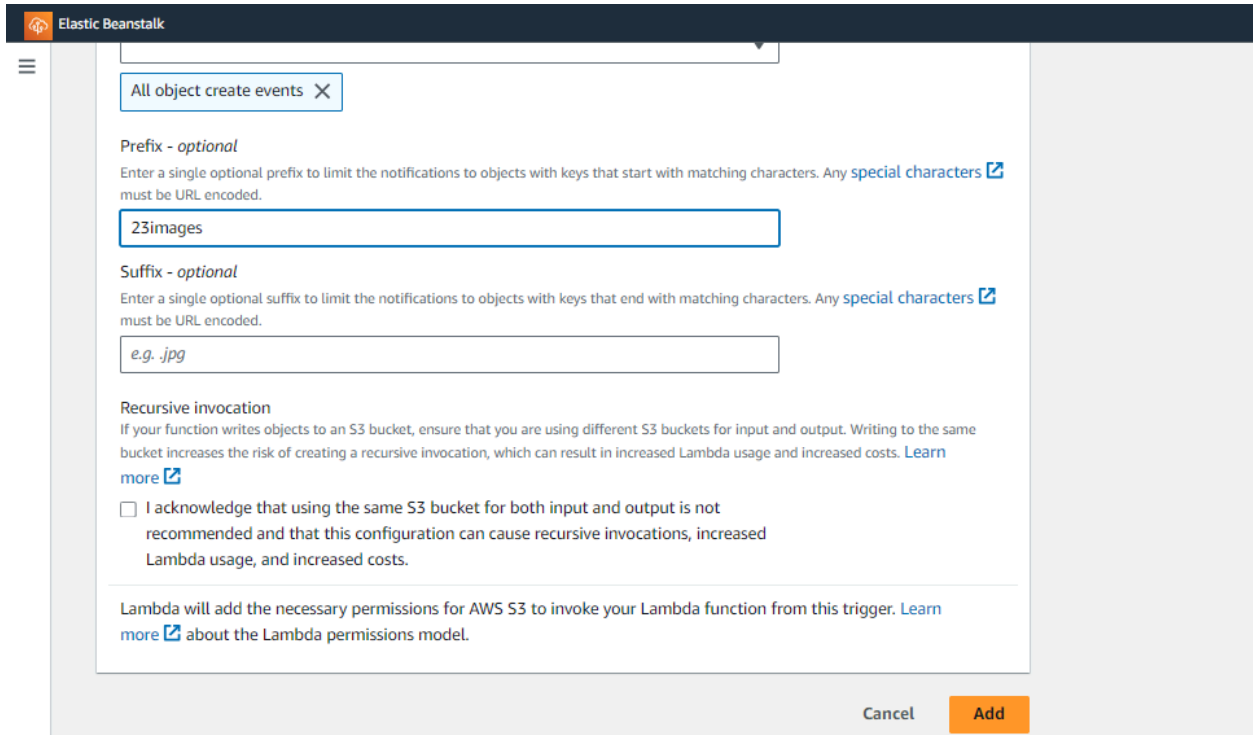
Step 6: Now In Code section select the created event from the dropdown .



Step 7: Now In the Lambda function click on add trigger.

Now select the source as S3 then select the bucket name from the dropdown, keep other things to default and also you can add prefix to





Elastic Beanstalk

All object create events ✕

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any [special characters](#) must be URL encoded.

23images

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any [special characters](#) must be URL encoded.

e.g. .jpg

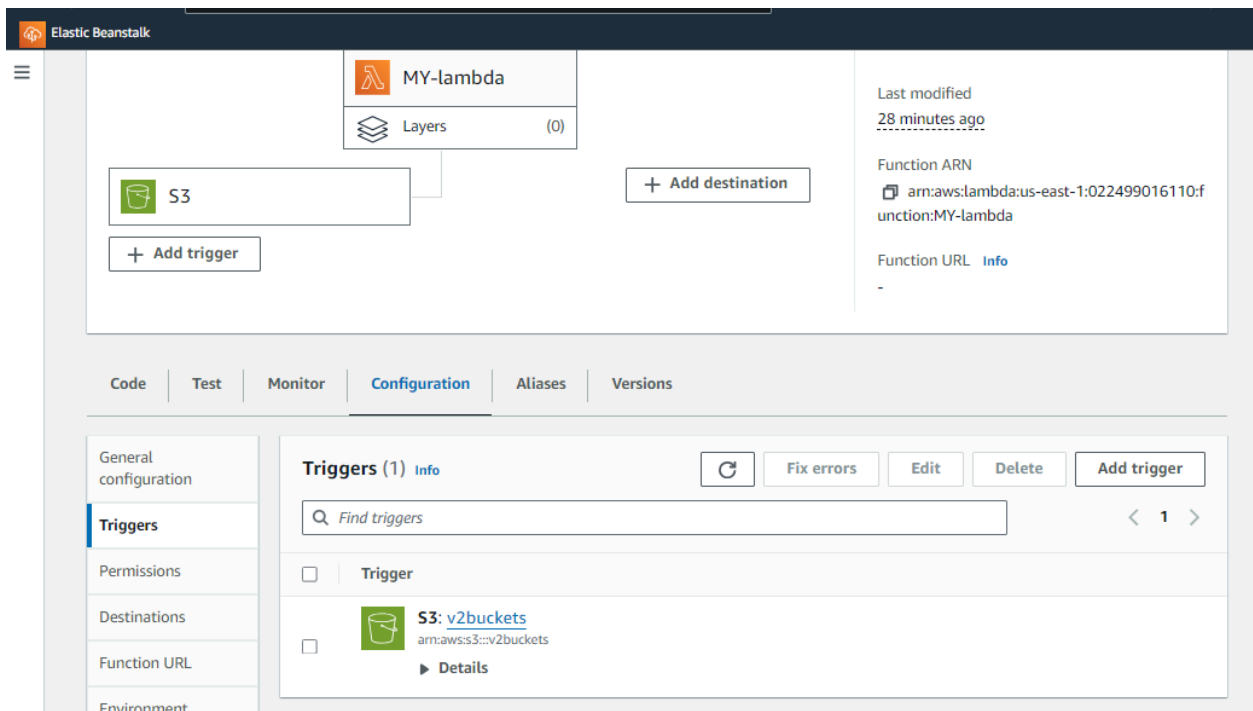
Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

☐ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel Add

Step 8: Now Write code that logs a message like “An Image has been added” when triggered. Save the file and click on deploy.



Elastic Beanstalk

MY-lambda

Layers (0)

S3

+ Add trigger

+ Add destination

Last modified
28 minutes ago

Function ARN
arn:aws:lambda:us-east-1:022499016110:function:MY-lambda

Function URL [Info](#)
-

Code | Test | Monitor | **Configuration** | Aliases | Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment

Triggers (1) Info

Find triggers

Trigger

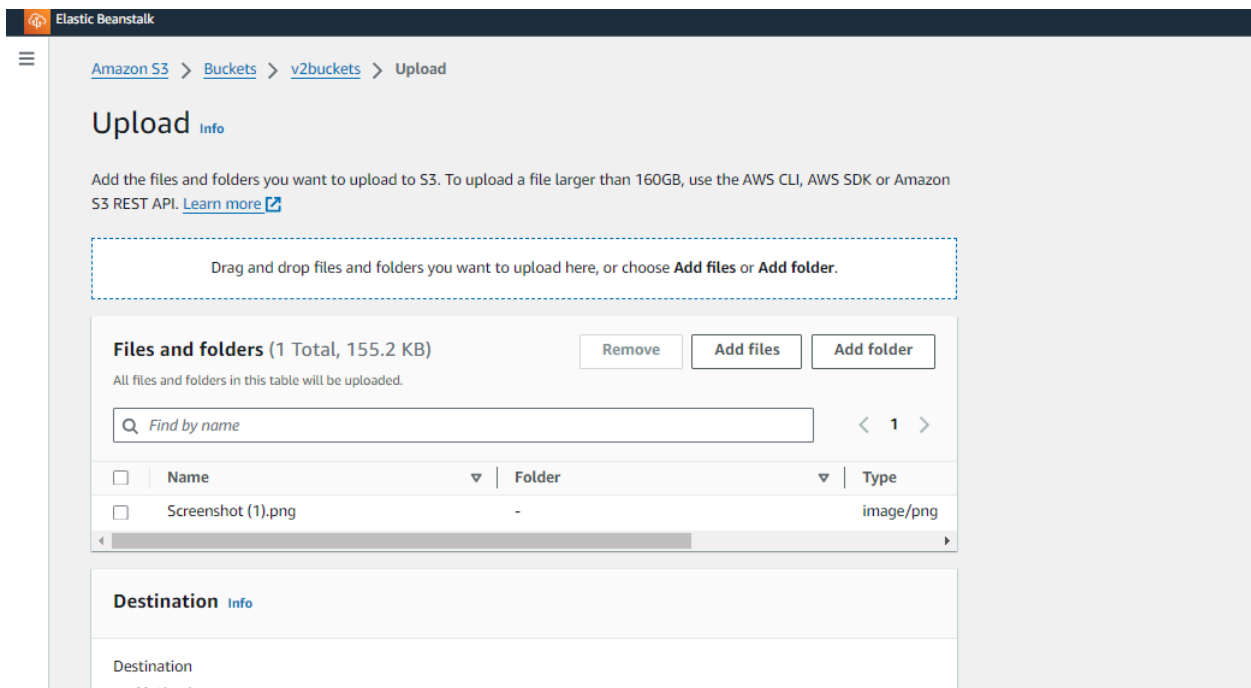
S3: v2buckets
arn:aws:s3::v2buckets

Details

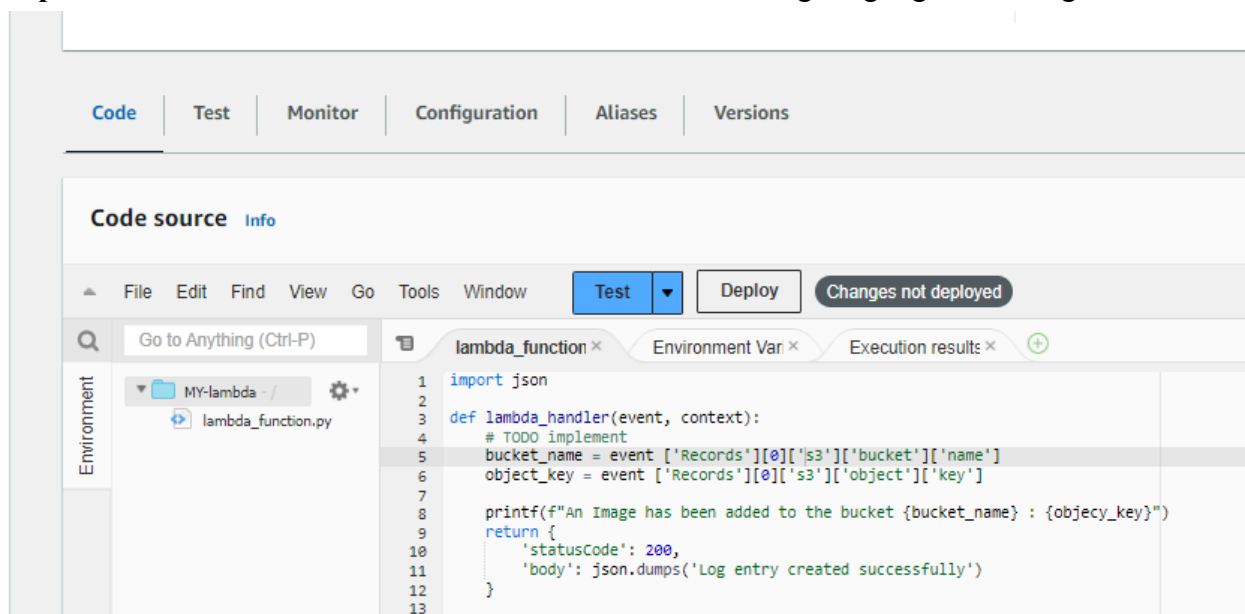
Fix errors Edit Delete Add trigger

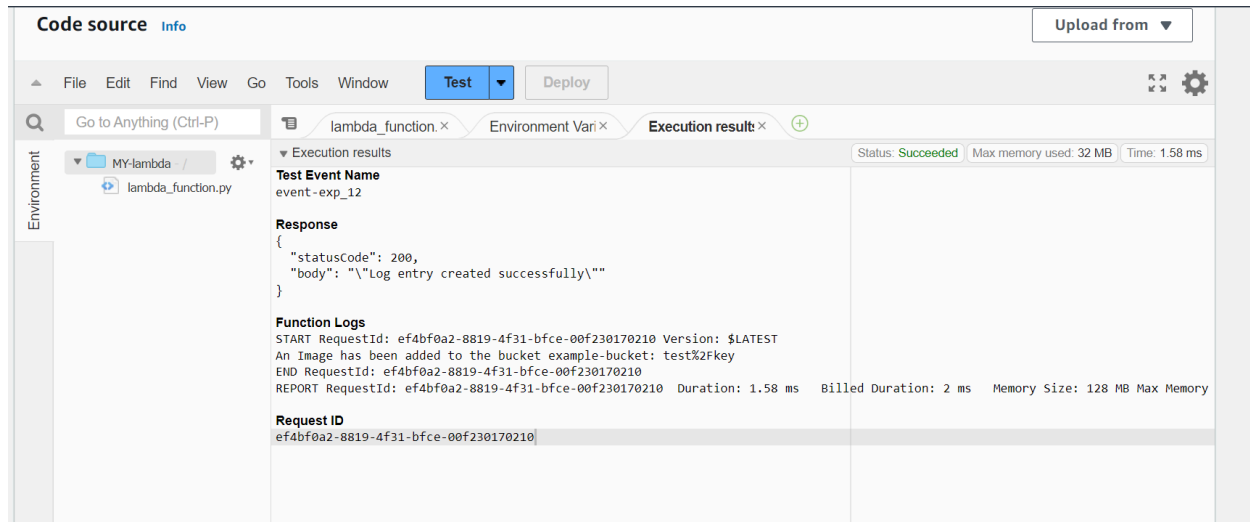
< 1 >

Step 9: Now upload any image to the bucket.

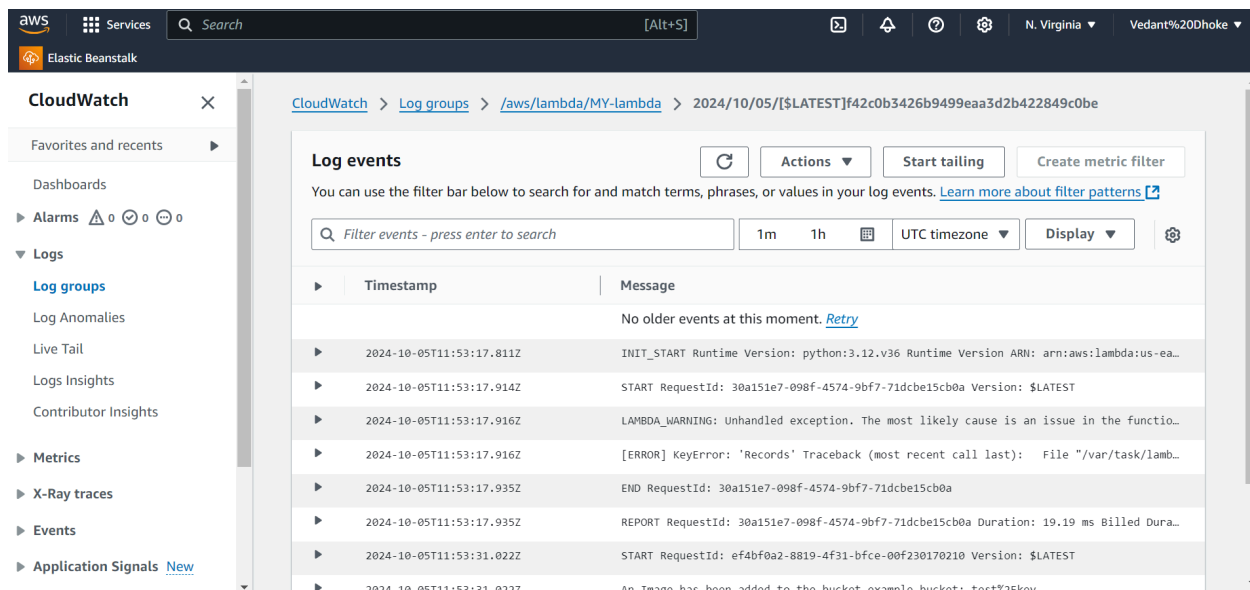


Step 10: Now to click on test in lambda to check whether it is giving log when image is added to S3.





Step 11: Now Lets see the log on Cloud watch.To see it go to monitor section and then click on view cloudwatch logs.



Conclusion: In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. The function was successfully triggered by S3 object uploads, validating the functionality of Lambda's event-driven architecture. This experiment demonstrated how Lambda can efficiently respond to S3 events and how to troubleshoot common issues with event structure.

Errors faced during the experiment :

It is important to configure S3 triggers properly, after changing the code its important to save the code then deploy it and then click on the test to see successful output otherwise we might see an error after clicking on the test.