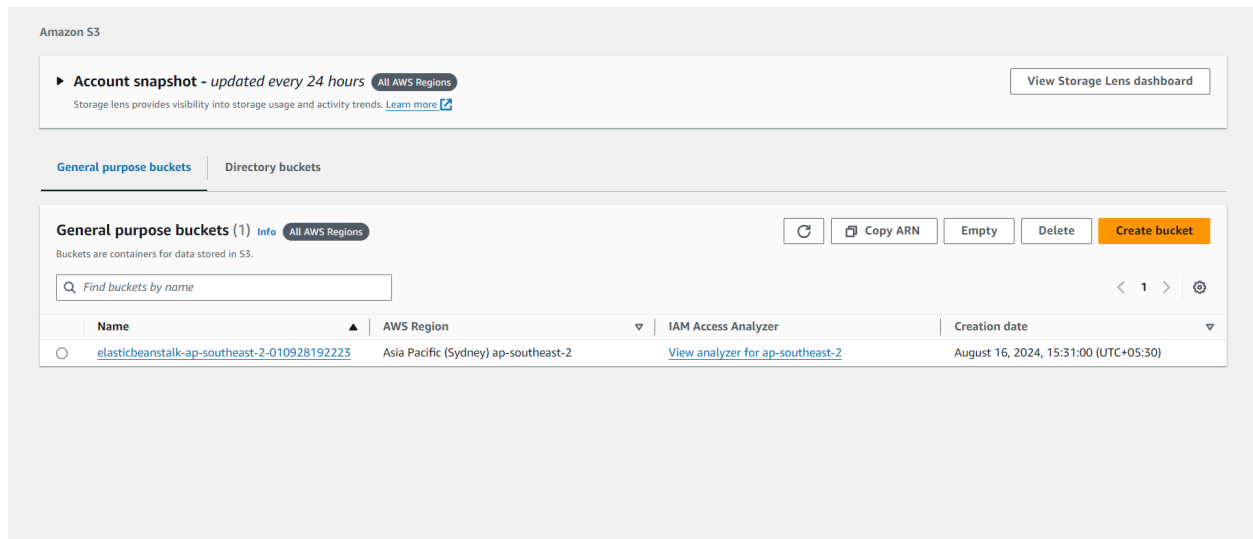


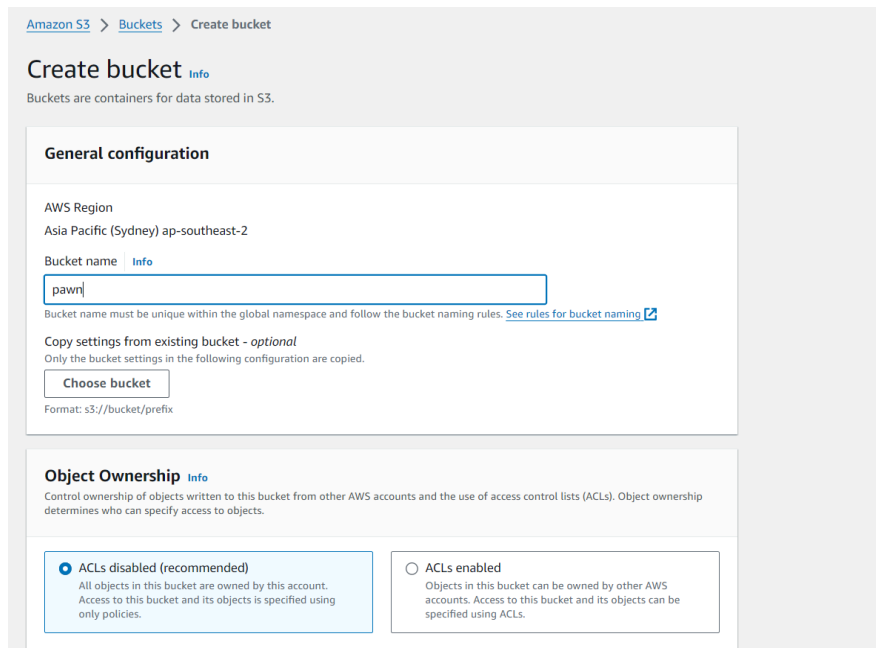
EXPERIMENT NO. 12

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.

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



2. Now Give a name to the Bucket, select general purpose project and deselect the Block public access and keep other things to default.



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☐ **Block all public access**

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**

S3 will ignore all ACLs that grant public access to buckets and objects.

☐ **Block public access to buckets and objects granted through *new* public bucket or access point policies**

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.



Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☒ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

General purpose buckets (2) [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	elasticbeanstalk-ap-southeast-2-010928192223	Asia Pacific (Sydney) ap-southeast-2	View analyzer for ap-southeast-2	August 16, 2024, 15:31:00 (UTC+05:30)
<input type="radio"/>	pawn	Asia Pacific (Sydney) ap-southeast-2	View analyzer for ap-southeast-2	October 6, 2024, 03:13:53 (UTC+05:30)

Thus, we have created a bucket named *pawn*.

3. Open lambda console and click on the create function button.

AWS Lambda
lets you run code without thinking about servers.

You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

Get started
Author a Lambda function from scratch, or choose from one of many preconfigured examples.
[Create a function](#)

How it works [Run](#) Next: Lambda responds to events

.NET Java **Node.js** Python Ruby Custom runtime

```

1 * exports.handler = async (event) => {
2   console.log(event);
3   return 'Hello from Lambda!';
4 }
5

```

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

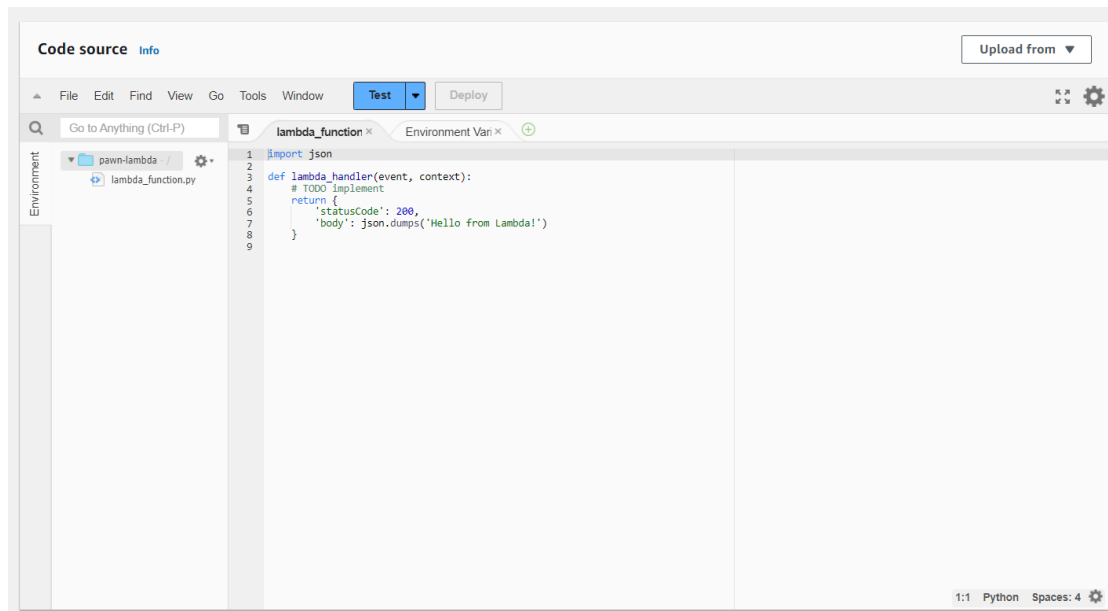
- Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12, Architecture as x86 and Execution role to Create a new role with basic Lambda permissions

The screenshot shows the 'Create function' page in the AWS Lambda console. The breadcrumb navigation is 'Lambda > Functions > Create function'. The page title is 'Create function' with an 'Info' link. Below the title, it says 'Choose one of the following options to create your function.' There are three radio button options: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. The 'Basic information' section contains: 'Function name' with the value 'pawn-lambda'; 'Runtime' set to 'Python 3.12'; 'Architecture' set to 'x86_64'; and 'Permissions' with a link to 'Change default execution role'. At the bottom are 'Cancel' and 'Create function' buttons.

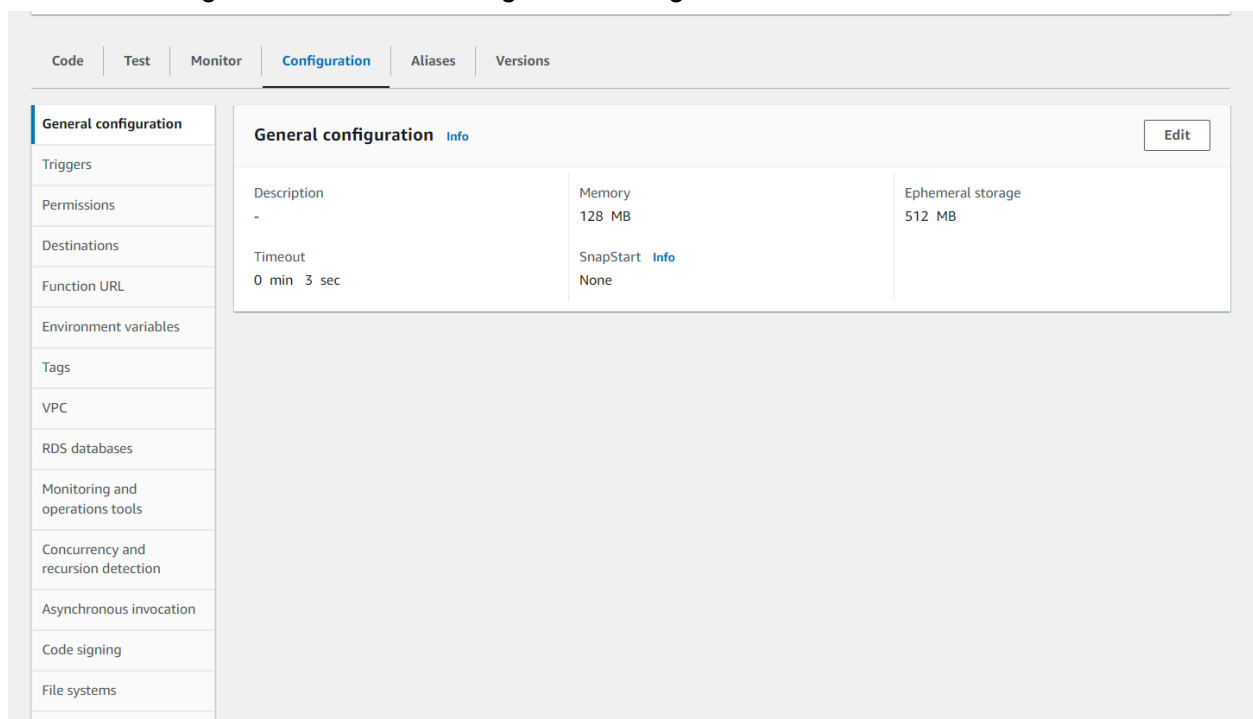
Thus, we have successfully created a lambda function named *pawn-lambda*.

The screenshot shows the 'pawn-lambda' function details page in the AWS Lambda console. The breadcrumb navigation is 'Lambda > Functions > pawn-lambda'. The page title is 'pawn-lambda'. There are buttons for 'Throttle', 'Copy ARN', and 'Actions'. The 'Function overview' section shows a diagram of the function with layers and a button to 'Add trigger'. The 'Code source' section shows the function's code editor with a 'Test' button. The right sidebar contains a 'Tutorials' section with a link to 'Create a simple web app'.

This is how the function is displayed.



Go to the Configuration tab to see the general configuration of our function.



We want to edit the timeout time and the rest can be kept the default.

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

To edit, we go to the Edit button seen in the above screenshot.

Edit basic settings

Basic settings [Info](#)

Description - *optional*

Memory [Info](#)
Your function is allocated CPU proportional to the memory configured.

MB

Set memory to between 128 MB and 10240 MB

Ephemeral storage [Info](#)
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)

MB

Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart [Info](#)
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](#).

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

Timeout

min 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

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](#) Save Test

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

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

Event JSON Format JSON

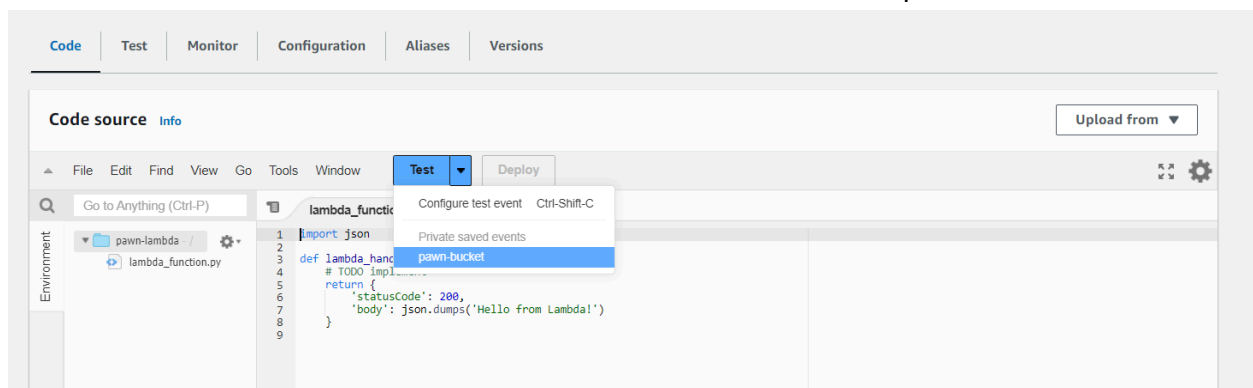
Event JSON

Format JSON

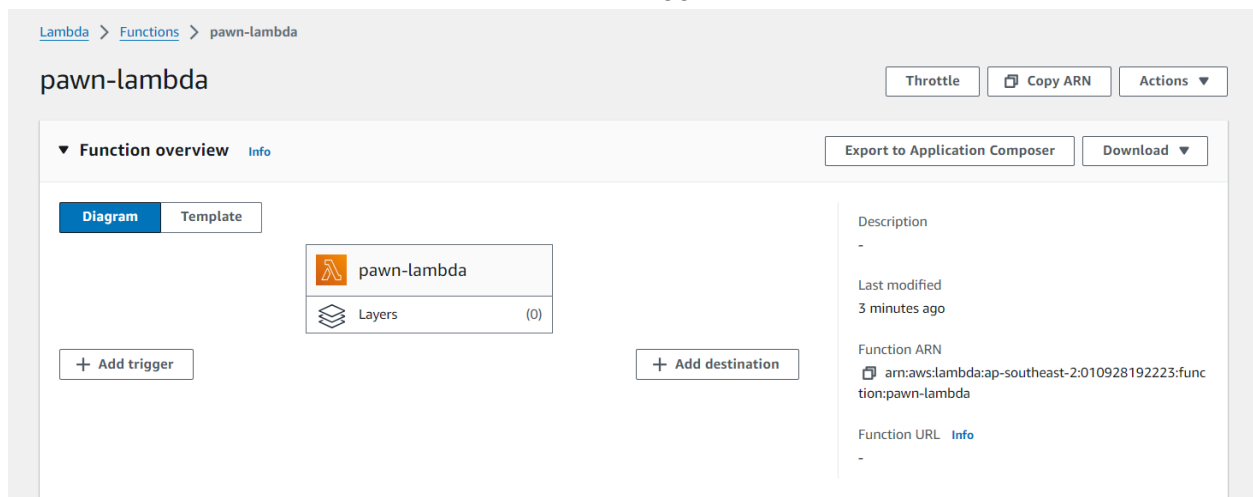
```
1 {
2   "Records": [
3     {
4       "eventVersion": "2.0",
5       "eventSource": "aws:s3",
6       "awsRegion": "us-east-1",
7       "eventTime": "1970-01-01T00:00:00.000Z",
8       "eventName": "ObjectCreated:Put",
9       "userIdentity": {
10        "principalId": "EXAMPLE"
11      },
12      "requestParameters": {
13        "sourceIPAddress": "127.0.0.1"
14      },
15      "responseElements": {
16        "x-amz-request-id": "EXAMPLE123456789",
17        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklmbdaaisawsome/mnopqrstuvwxyzABCDEFGH"
18      },
19      "s3": {
20        "s3SchemaVersion": "1.0",
21        "configurationId": "testConfigRule",
22        "bucket": {
23          "name": "example-bucket",
24          "ownerIdentity": {
25            "principalId": "EXAMPLE"
26          },
27          "arn": "arn:aws:s3:::example-bucket"
28        },
29        "object": {
30          "key": "test%2Fkey",
```

1:1 JSON Spaces: 2

6. Now in the Code section select the created event from the dropdown.



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 image

[Lambda](#) > [Add triggers](#)

Add trigger

Trigger configuration [info](#)

 **S3**
aws asynchronous storage

Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.

Bucket region: ap-southeast-2

Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

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.

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.

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 invocation, increased

Info **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](#)

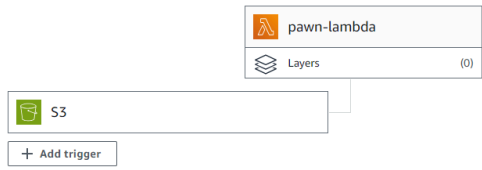
[Lambda](#) > [Functions](#) > [pawn-lambda](#)

pawn-lambda

☒ The trigger pawn was successfully added to function pawn-lambda. The function is now receiving events from the trigger.


Function overview [info](#)

Diagram **Template**



Description
-

Last modified
5 minutes ago

Function ARN
 am:aws:lambda:ap-southeast-2:010928192223:function:pawn-lambda

Function URL [info](#)
-


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

General configuration

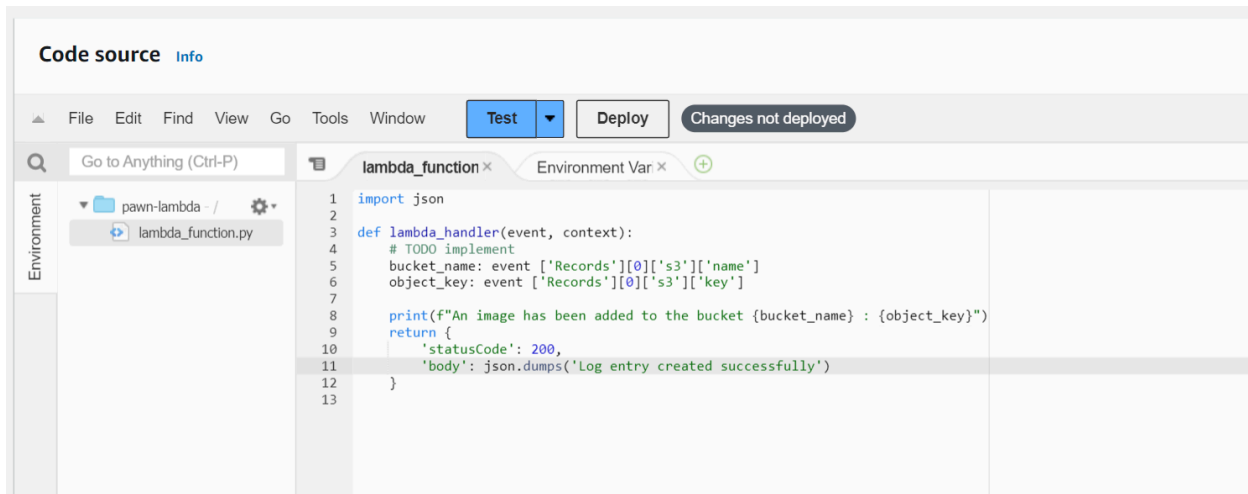
Triggers

Triggers (1) [info](#)

☐ **Trigger**

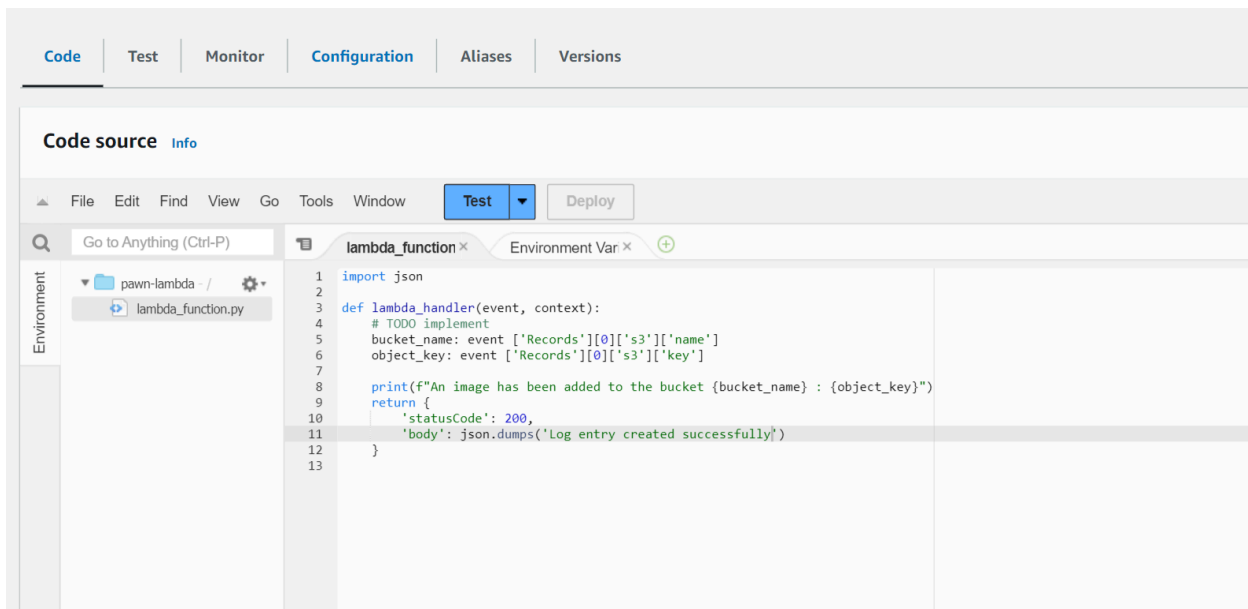
☒  **S3: pawn**
arn:aws:s3:::pawn
[Details](#)

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



The screenshot shows the AWS Lambda console's code editor interface. At the top, there's a 'Code source' link and an 'Info' icon. Below this is a menu bar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', and 'Window'. To the right of the menu are 'Test' and 'Deploy' buttons, and a 'Changes not deployed' status indicator. The left sidebar shows the 'Environment' tab with a file explorer displaying 'pawm-lambda - /' and 'lambda_function.py'. The main editor area shows the code for 'lambda_function.py' with line numbers 1 through 13. The code imports 'json' and defines a 'lambda_handler' function that extracts 'name' and 'key' from an event, prints a message, and returns a JSON response with status 200.

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     bucket_name: event ['Records'][0]['s3']['name']
6     object_key: event ['Records'][0]['s3']['key']
7
8     print(f"An image has been added to the bucket {bucket_name} : {object_key}")
9     return {
10         'statusCode': 200,
11         'body': json.dumps('Log entry created successfully')}
12
13
```



This screenshot is identical to the one above, showing the same AWS Lambda console interface and code for 'lambda_function.py'. It displays the 'Code source' link, menu bar, environment sidebar, and the Python code for the lambda handler function.

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     bucket_name: event ['Records'][0]['s3']['name']
6     object_key: event ['Records'][0]['s3']['key']
7
8     print(f"An image has been added to the bucket {bucket_name} : {object_key}")
9     return {
10         'statusCode': 200,
11         'body': json.dumps('Log entry created successfully')}
12
13
```

- Now upload any image to the bucket.

Amazon S3 > Buckets > pawn > Upload

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (0) [Remove](#) [Add files](#) [Add folder](#)

All files and folders in this table will be uploaded.

<input type="checkbox"/>	Name	Folder
No files or folders		
You have not chosen any files or folders to upload.		

Destination [Info](#)

Destination

[s3://pawn](#)

► **Destination details**
Bucket settings that impact new objects stored in the specified destination.

► **Permissions**

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (1 Total, 77.3 KB) [Remove](#) [Add files](#) [Add folder](#)

All files and folders in this table will be uploaded.

<input type="checkbox"/>	Name	Folder
<input type="checkbox"/>	Screenshot 2024-09-10 220423.png	-

Destination [Info](#)

Destination

[s3://pawn](#)

► **Destination details**
Bucket settings that impact new objects stored in the specified destination.

► **Permissions**
Grant public access and access to other AWS accounts.

Upload: status Close

The information below will no longer be available after you navigate away from this page.

Summary

Destination s3://pawm	Succeeded 1 file, 77.3 KB (100.00%)	Failed 0 files, 0 B (0%)
--------------------------	--	-----------------------------

Files and folders (1 Total, 77.3 KB)

Find by name

Name	Folder	Type	Size	Status	Error
Screenshot 2...	-	image/png	77.3 KB	Succeeded	-

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

Code source Info Upload from

File Edit Find View Go Tools Window **Test** Deploy

Go to Anything (Ctrl-P)

Environment: pawm-lambda - / lambda_function.py

Execution results Status: Succeeded Max memory used: 32 MB Time: 1.67 ms

Test Event Name
pawm-bucket

Response

```
{
  "statusCode": 200,
  "body": "\\Log entry created successfully\\"
}
```

Function Logs

```
START RequestId: 69a05696-0896-495f-906f-cfa14acf7bb4 Version: $LATEST
An image has been added to the bucket example-bucket : test%2Fkey
END RequestId: 69a05696-0896-495f-906f-cfa14acf7bb4
REPORT RequestId: 69a05696-0896-495f-906f-cfa14acf7bb4 Duration: 1.67 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB Init D
```

Request ID
69a05696-0896-495f-906f-cfa14acf7bb4

11. Now let's see the log on CloudWatch. To see it go to monitor section and then click on view cloudwatch logs.

The screenshot displays the AWS CloudWatch console. The left sidebar shows the navigation menu with 'Logs' selected. The main content area shows the 'Log groups' page for the log group `/aws/lambda/pawn-lambda`. Below the log group list, the 'Log events' page is shown for the same log group. The log events table displays several log entries with timestamps and messages.

Timestamp	Message
2024-10-05T22:03:38.376Z	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:ap-southeast-2::runtime:188d9ca2e2714ff5637bd2bbe06ceb81ec3bc408a0f27...
2024-10-05T22:03:38.445Z	START RequestId: 69a05696-0896-495f-906f-cfa14acf7bb4 Version: \$LATEST
2024-10-05T22:03:38.446Z	An image has been added to the bucket example-bucket : test12fkey
2024-10-05T22:03:38.447Z	END RequestId: 69a05696-0896-495f-906f-cfa14acf7bb4
2024-10-05T22:03:38.447Z	REPORT RequestId: 69a05696-0896-495f-906f-cfa14acf7bb4 Duration: 1.67 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 32 MB Init Durati...

Conclusion: In this experiment, we successfully created an AWS Lambda function that logs a message when an image is uploaded to an S3 bucket. It is important to note that we have to select the S3-put template in the event otherwise code will give an error. 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.