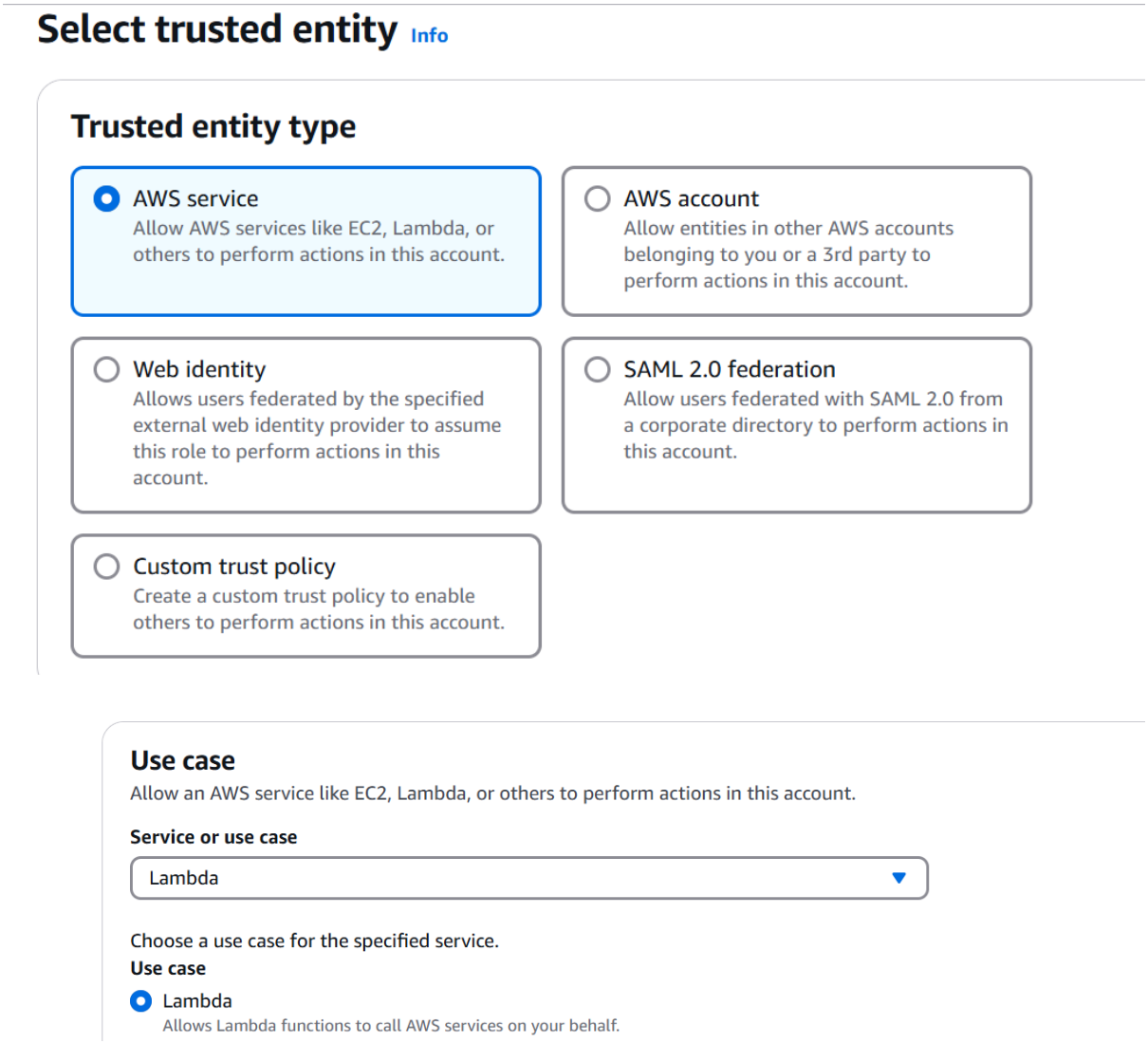
Lab: S3 Lambda

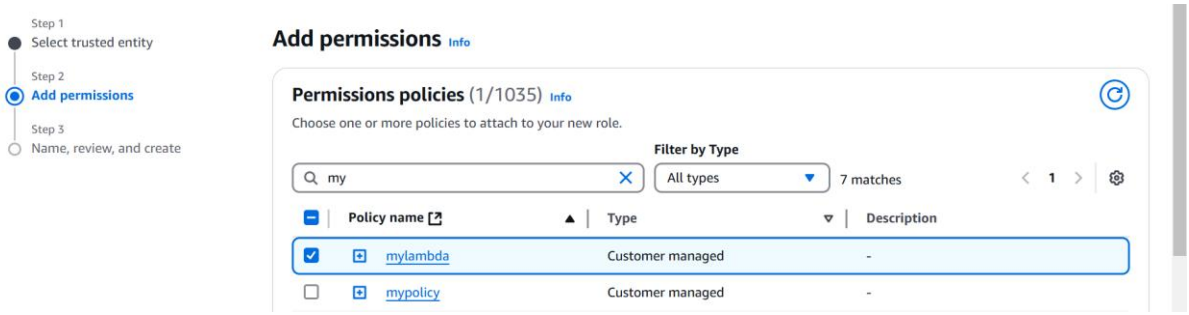
1) Create two buckets with default configuration
    a. Amit23comp
    b. Amit23comp-replica
2) Create a new policy in iam , Change source and destination bucket name using arn

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "putObject",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::awsbucketamit-replica/*"
      ]
    },
    {
      "Sid": "getObject",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::myamit23bucket/*"
      ]
    }
  ]
}
```

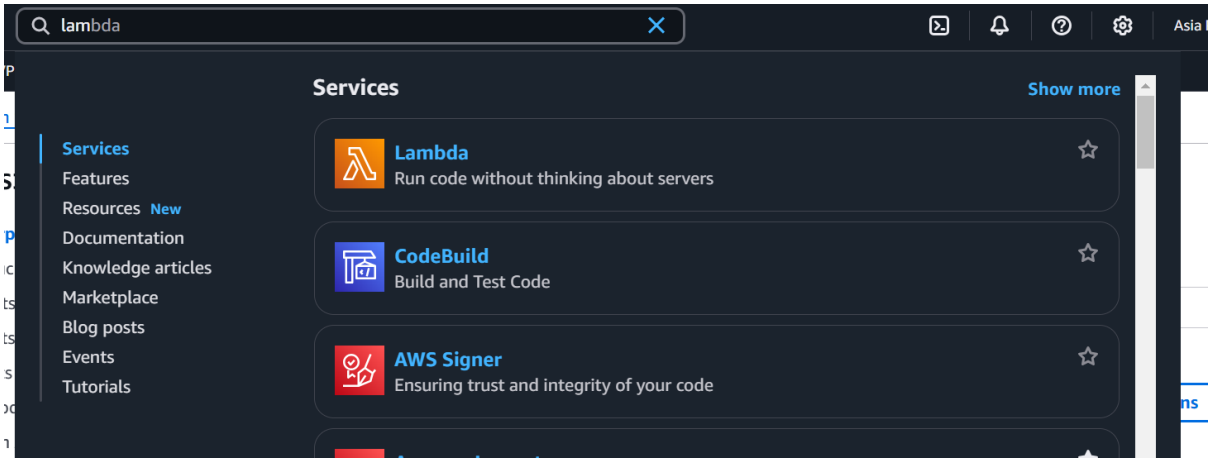
Create a role



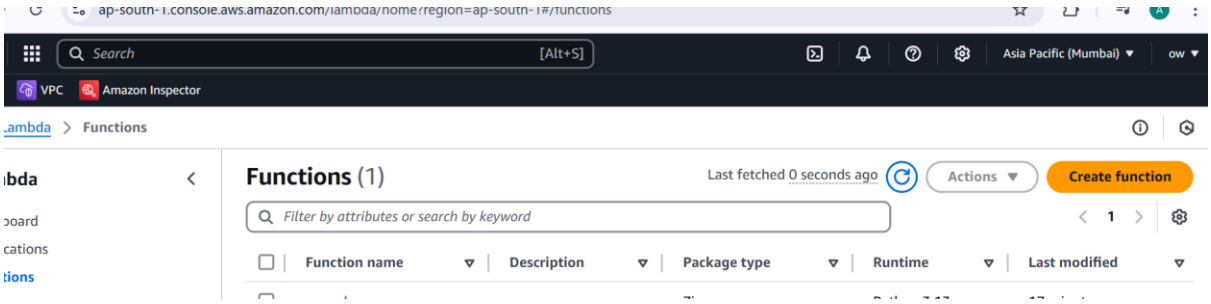
Select your policy



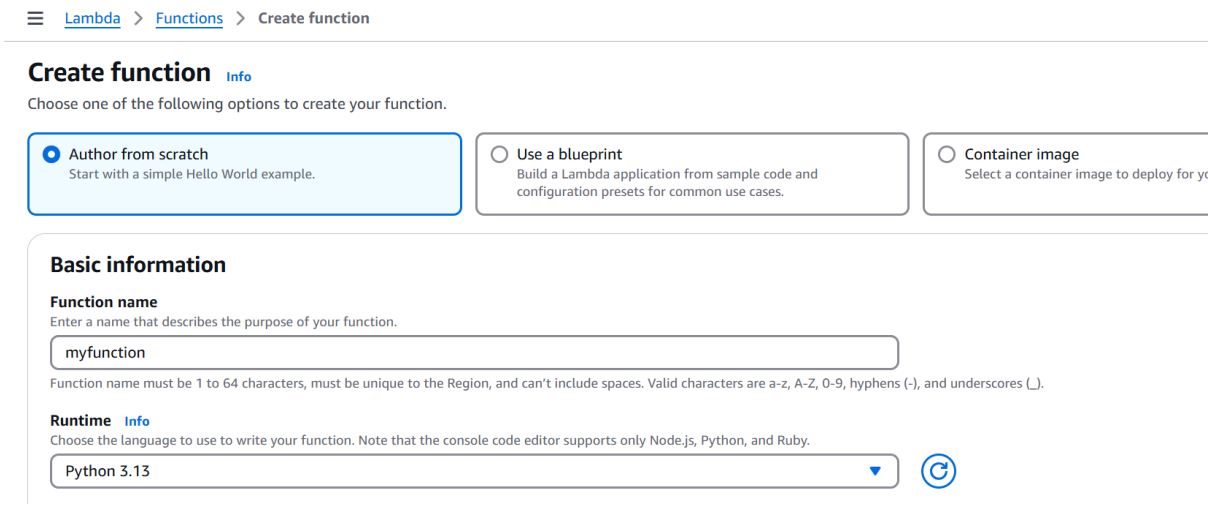
Check for lambda service



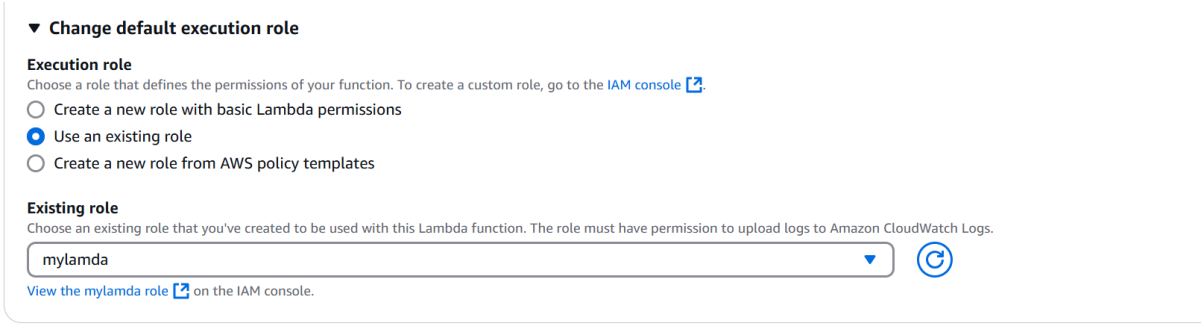
Create a function



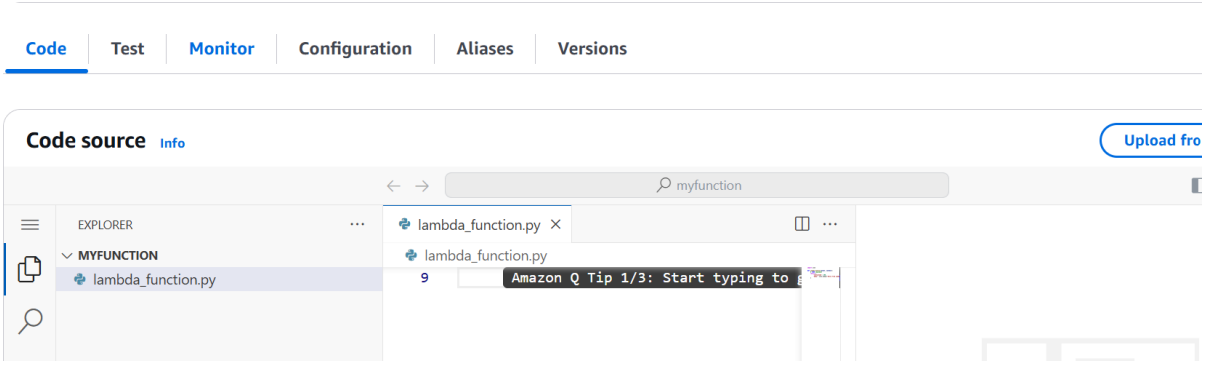
Provide function name and python3.13



Provide the role



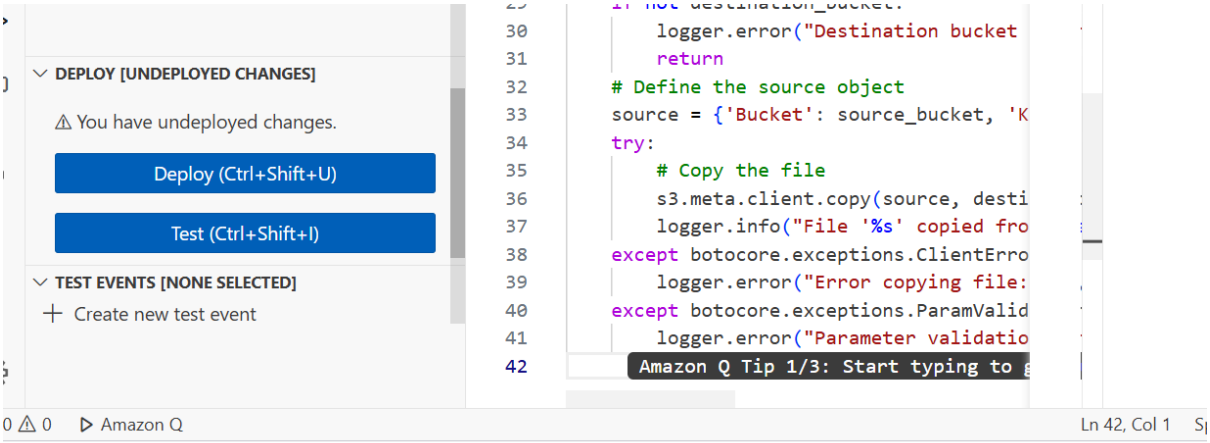
Copy the code



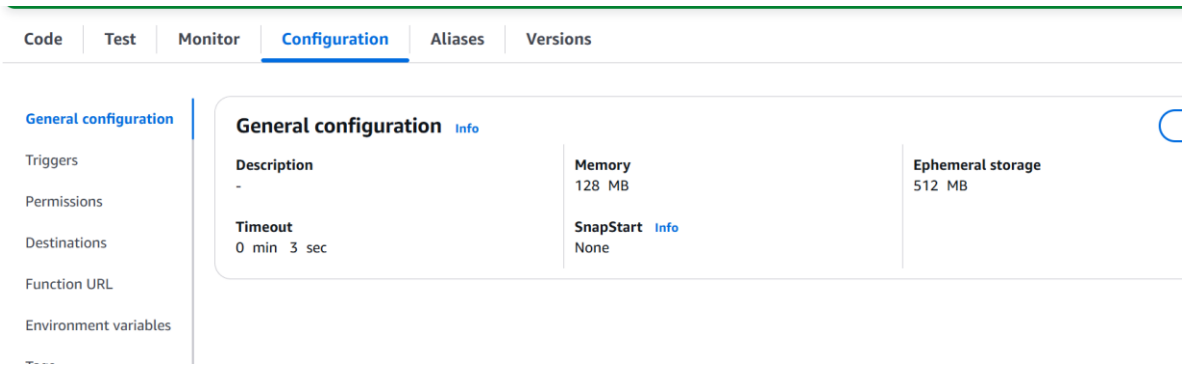
```
import boto3
import botocore
import os
import logging
# Set up logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
# Initialize S3 resource
s3 = boto3.resource('s3')
def lambda_handler(event, context):
    logger.info("New files uploaded to the source bucket.")
    # Retrieve the key and bucket name from the event
    try:
        key = event['Records'][0]['s3']['object']['key']
        source_bucket = event['Records'][0]['s3']['bucket']['name']
    except KeyError as e:
        logger.error("Missing key information in event: %s", e)
        return
    # Get the destination bucket from environment variables
    destination_bucket = os.getenv('destination_bucket')
    if not destination_bucket:
        logger.error("Destination bucket environment variable  is not set.")
        return
    # Define the source object
    source = {'Bucket': source_bucket, 'Key': key}
    try:
        # Copy the file
        s3.meta.client.copy(source, destination_bucket, key)
        logger.info("File '%s' copied from '%s' to '%s' successfully!", key, source_bucket,
destination_bucket)
    except botocore.exceptions.ClientError as error:
```

```
    logger.error("Error copying file: %s", error)
except botocore.exceptions.ParamValidationError as error:
    logger.error("Parameter validation error: %s", error)
```

Click on Deploy



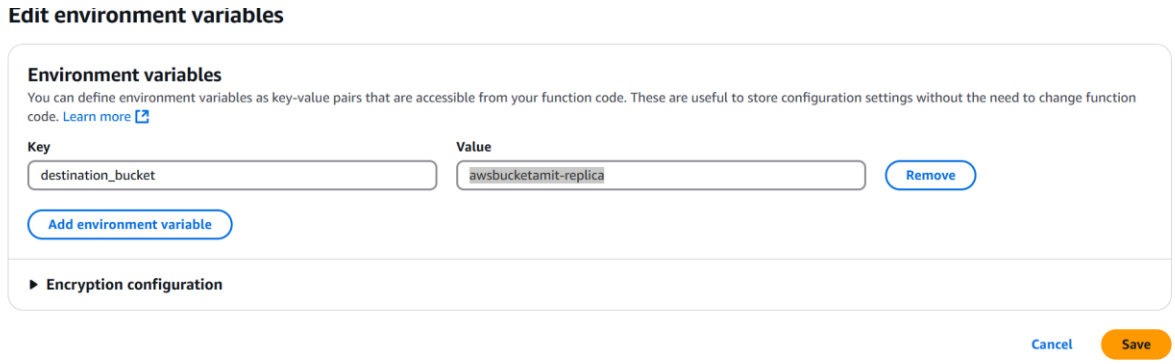
Click on Configuration



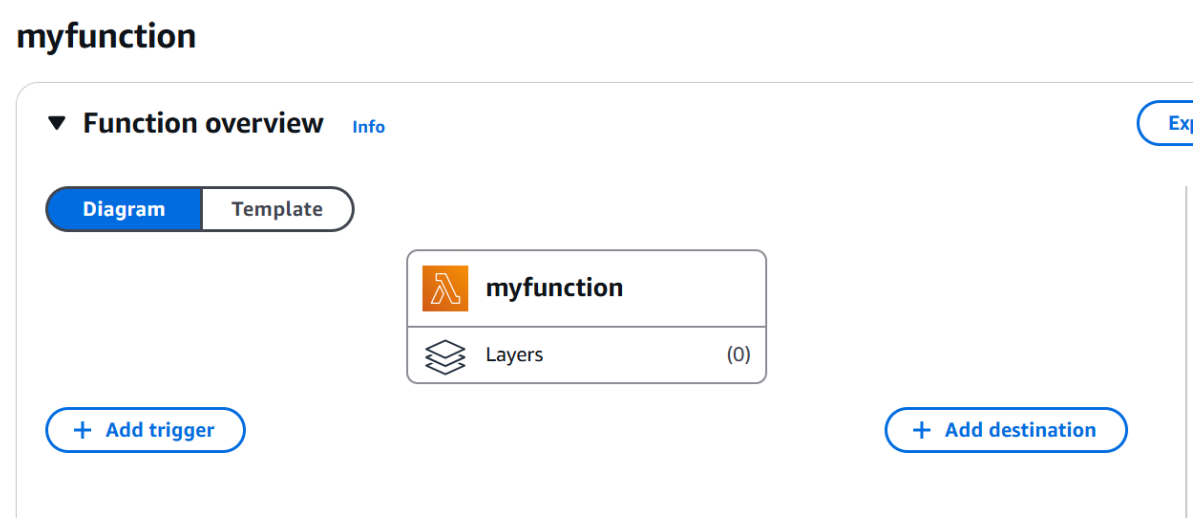
Click on Environment variables

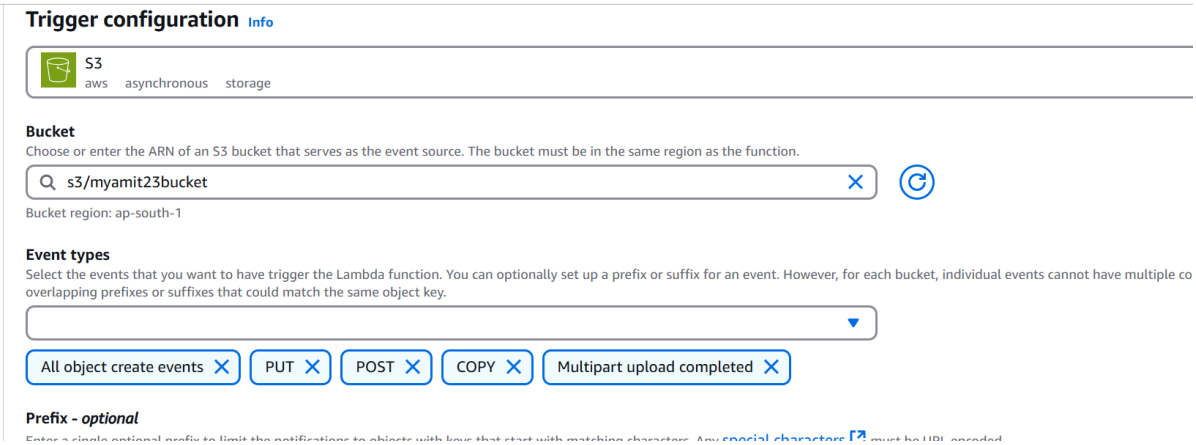
Key :destination_bucket

Value: awsbucketamit-replica



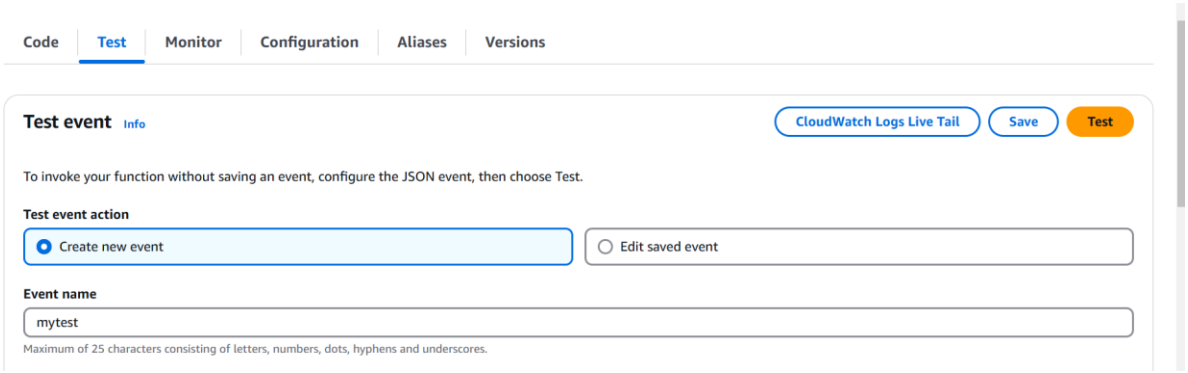
Click on Add triggers





Upload a jpg file in source bucket.

Create a test.



Json

```
{
 "Records": [
  {
   "s3": {
    "bucket": {
     "name": "provide the source bucket name"
    },
    "object": {
     "key": "object name which you uploaded"
    }
   }
  }
 ]
}
```

Save and Test

Updated and Rectified Lab Steps for S3 Lambda

1. Create Two Buckets:
Create two S3 buckets with default configuration:
- Source: myamit23bucket
- Destination: awsbucketamit-replica

2. Create an IAM Policy:
Go to IAM → Policies → Create Policy and paste the JSON (update bucket ARNs accordingly).

3. Create an IAM Role:
Go to IAM → Roles → Create Role. Choose Lambda as trusted entity, attach the policy created above.

4. Create a Lambda Function:
Name it s3CopyFunction, select Python 3.13, and attach the IAM role.

5. Paste the Lambda Code:
Use the provided Python code that uses boto3 to copy from source to destination bucket.

6. Deploy the Function:
Click Deploy to apply the code changes.

7. Add Environment Variable:
Go to Configuration → Environment variables:
- Key: destination_bucket
- Value: awsbucketamit-replica

8. Add S3 Trigger:
Attach a trigger for PUT events from source bucket.

9. Upload Test File:
Upload a file to the source bucket to trigger the function.

10. Create Test Event:
Manually test with a JSON simulating an S3 PUT event using the file key.

11. Rectification Steps:
- Verify logs in CloudWatch
- Check destination bucket for copied file
- Validate error handling for missing env or bad bucket name