Efficient Test Data Prediction Using Amazon S3 Triggers, Lambda Functions, and Sagemaker Endpoint


Owned by Ganesamanian Kolappan •••
Yesterday at 11:25 AM

This document provides a comprehensive guide on using a serverless AWS Lambda function designed to seamlessly integrate with Amazon SageMaker, a machine learning service. Its main purpose is to make predictions on data stored in a CSV file located in an Amazon S3 bucket. This Lambda function downloads the input data, sends it to a pre-configured SageMaker endpoint, and receives the model's predictions. It then manually calculates the accuracy of these predictions using the labels from the same CSV file, providing a detailed evaluation of the model's performance. Finally, the Lambda function saves the predictions and accuracy results in a JSON file and uploads it to another specified Amazon S3 bucket. This approach ensures efficient and automated machine learning model evaluation while leveraging AWS serverless computing for enhanced scalability.

🟦 Steps Involved

To implement this solution, complete the following high-level steps:

1. Creating S3 buckets
2. Creating Endpoint
3. Creating read-and-write policy
4. Creating lambda-specific roles and attaching the policy
5. Creating lambda function and setting trigger

👀 Prerequisite

Not expertise level is required at least first-hand experience so that the following steps are ease to understand.

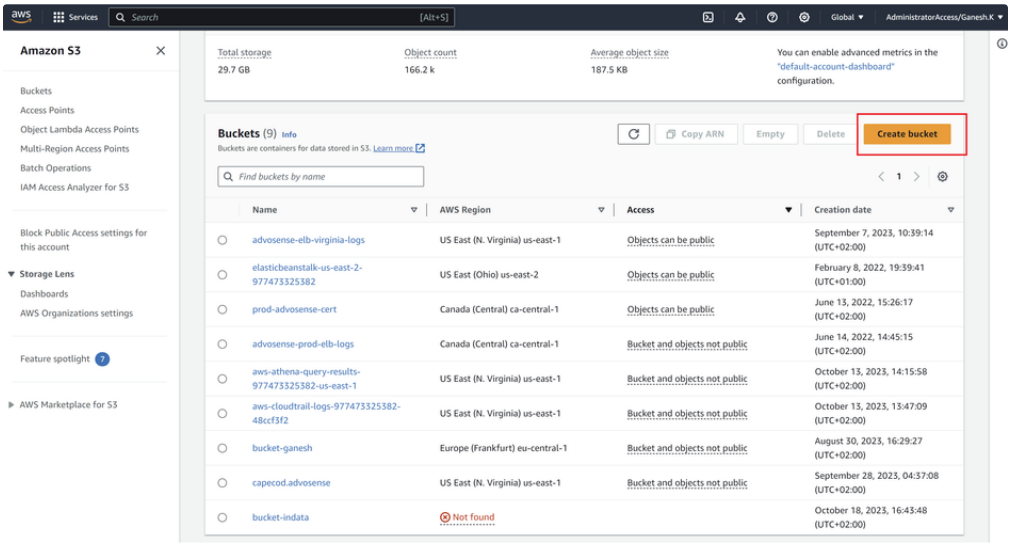
1. Python & pandas library
2. AWS
3. Sagemaker and ML knowledge

💻 Code

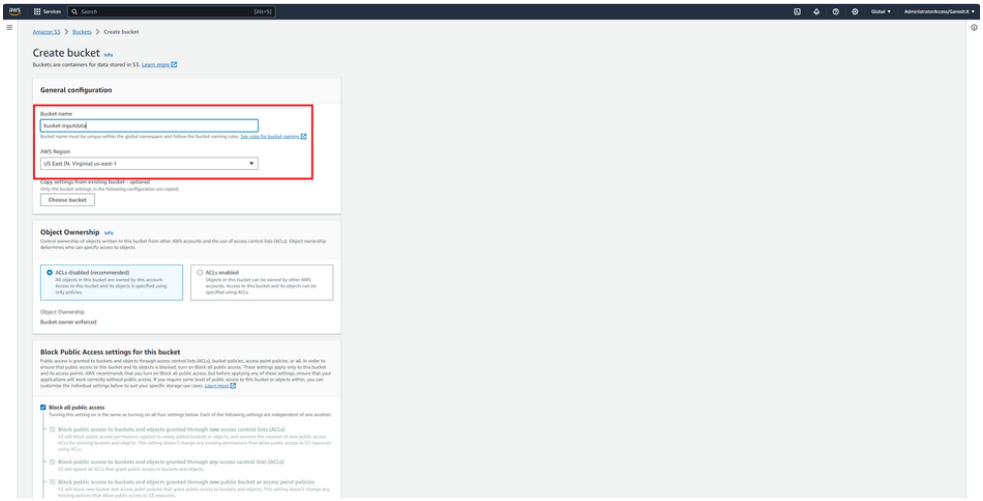
The code files and the current PDF can be found in the Bitbucket repository here

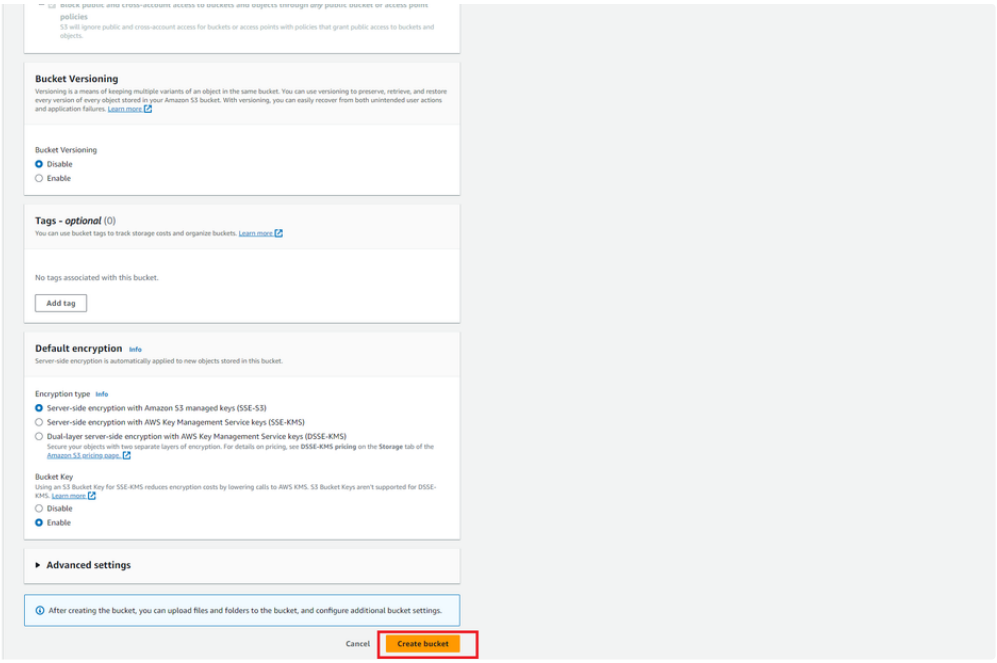
Step 1: Creating S3 buckets

1. Open the Amazon S3 console and select the **Buckets** page

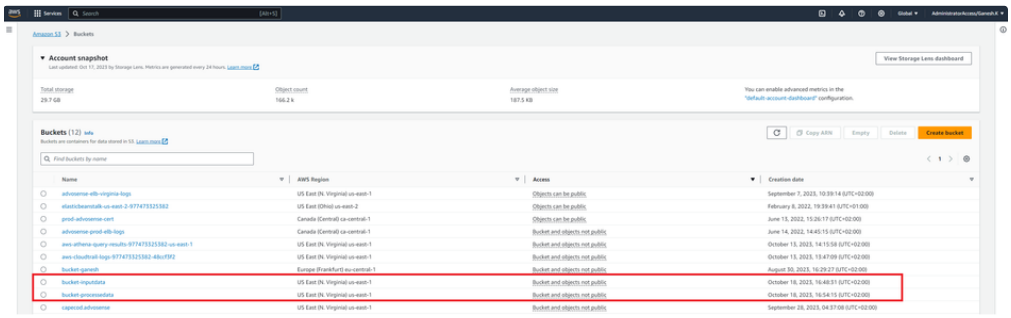


2. Choose **Create bucket** to load the input data.
3. Under **General configuration**, do the following:
   a. For the **Bucket name**, enter a globally unique name that meets the Amazon S3 Bucket naming rules. Bucket names can contain only lowercase letters, numbers, dots (.), and hyphens (-).
   b. For **AWS Region**, choose a Region. Later in the tutorial, you must create your Lambda function in the same Region.



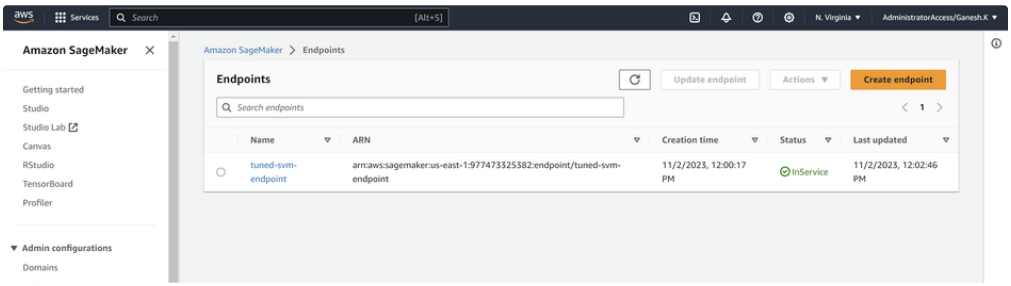


4. Leave all other options set to their default values and choose **Create bucket**.
5. Create **another bucket** to store the processed data as per the above procedure
6. Should have **two buckets** to store one for input data and one for processed data



Step 2: Creating Endpoint

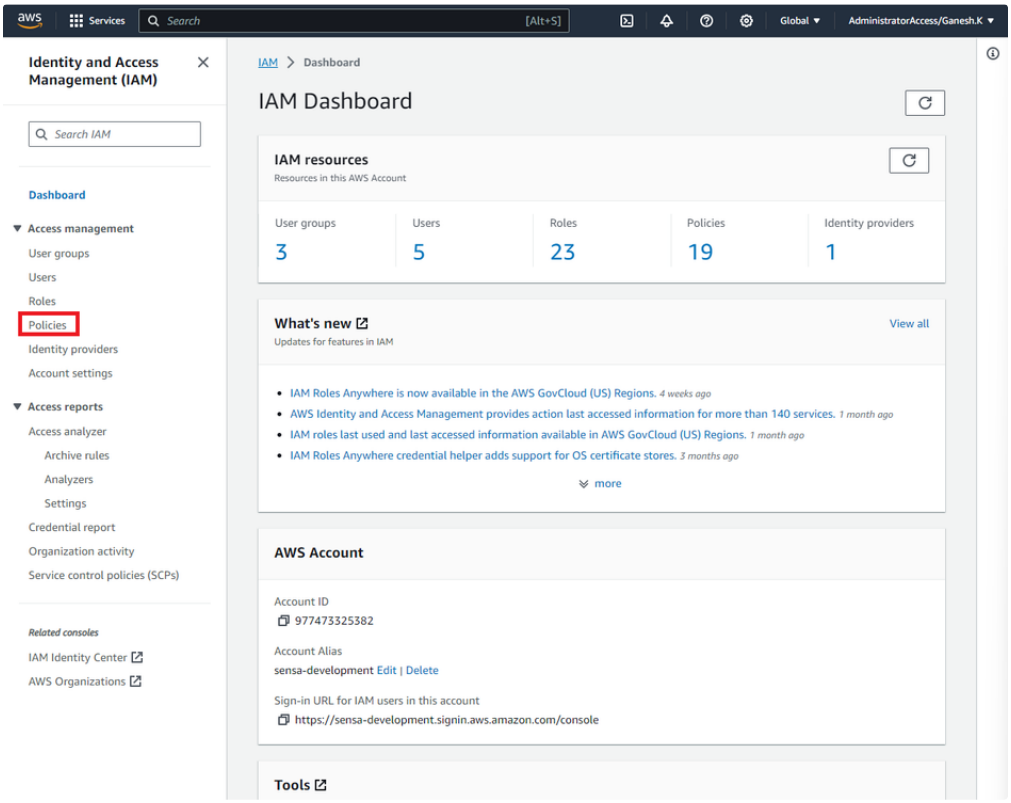
1. To do this one should go through my sagemaker file and implement it till the end to get the endpoint



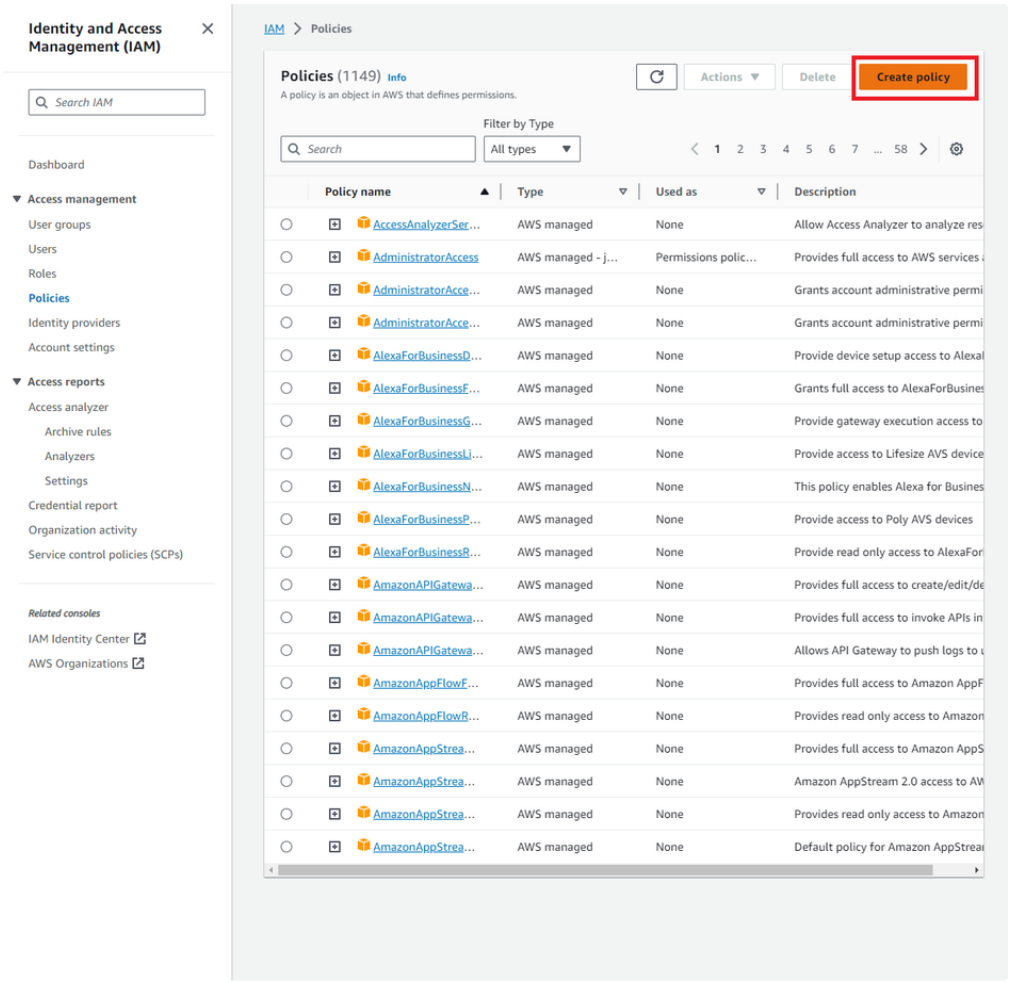
2. Note down the ARN value of the endpoint, which should be added to the policy for access in the next step.

Step 3: Creating a read-and-write policy

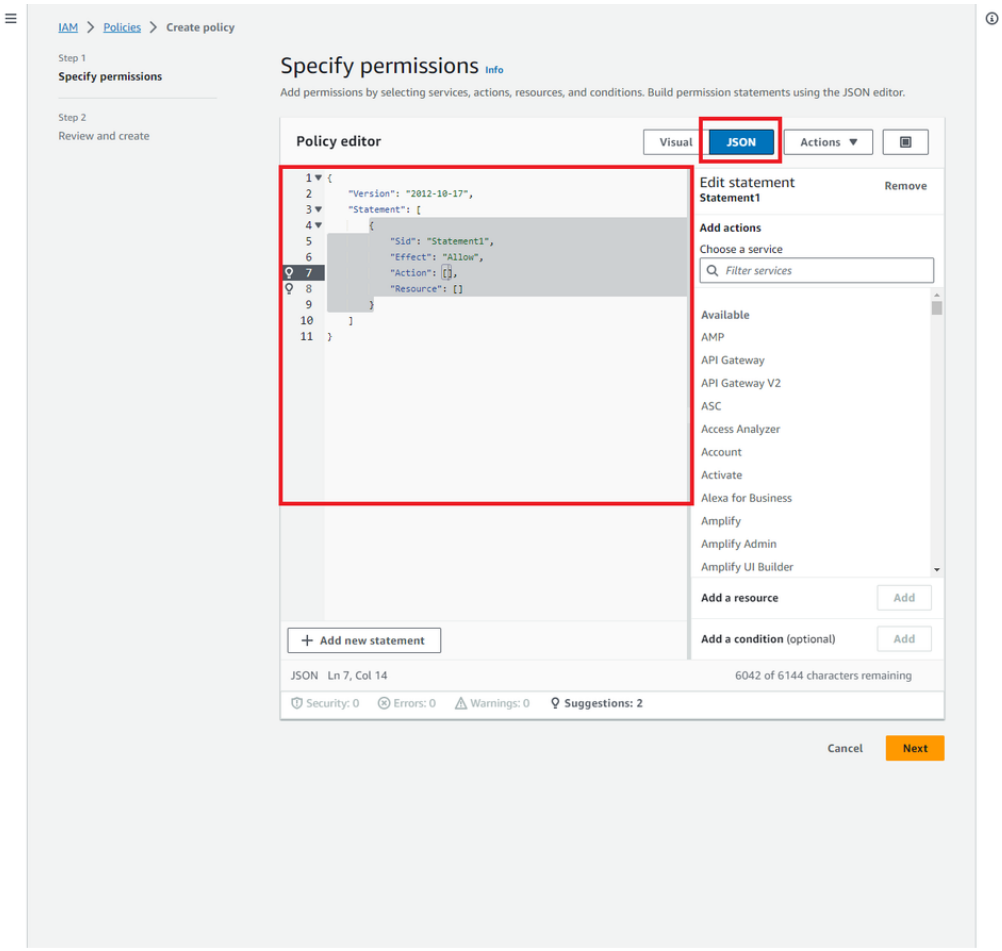
1. Open the **Management Console** and select the **IAM** page
2. On the **IAM** page select **policies**



3. Choose to **create a policy**



4. Select **JSON**

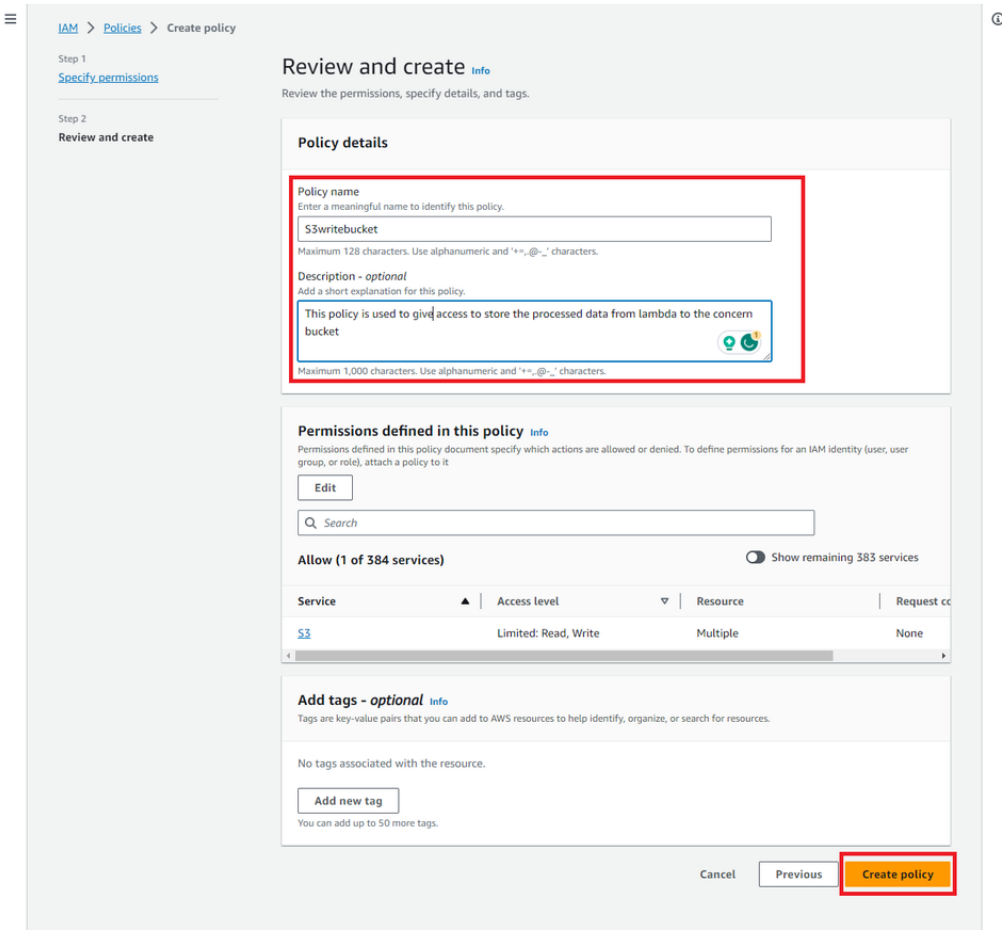


5. Paste the following code in the **policy editor field** giving access to S3 buckets and Sagemaker endpoint

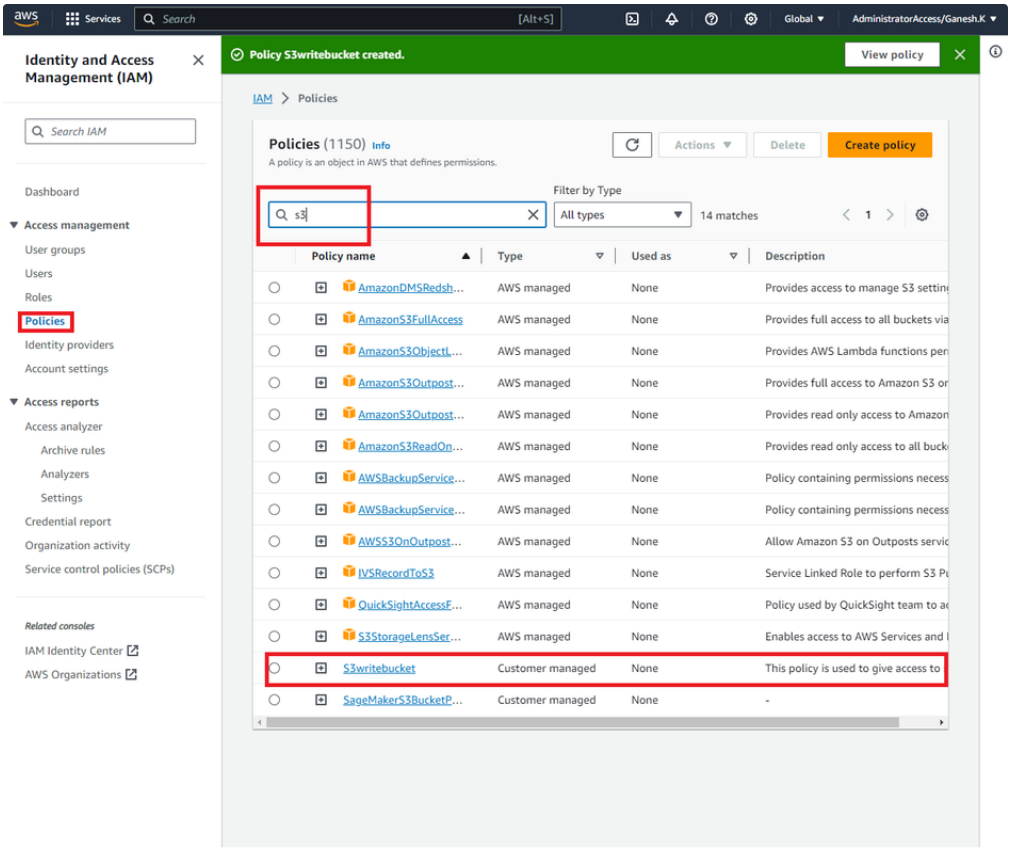
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "putObject",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-processedata/*"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

```
            "Action": [
                "sagemaker:InvokeEndpoint"
            ],
            "Resource": "arn:aws:sagemaker:us-east-1:977473325382:endpoint/tuned-svm-endpoint"
        }
    ]
}
```

6. Select **Next**
7. In the next window, give the **policy name and description**, then select **Create policy.** In our case the name of the policy is **S3writebucket**

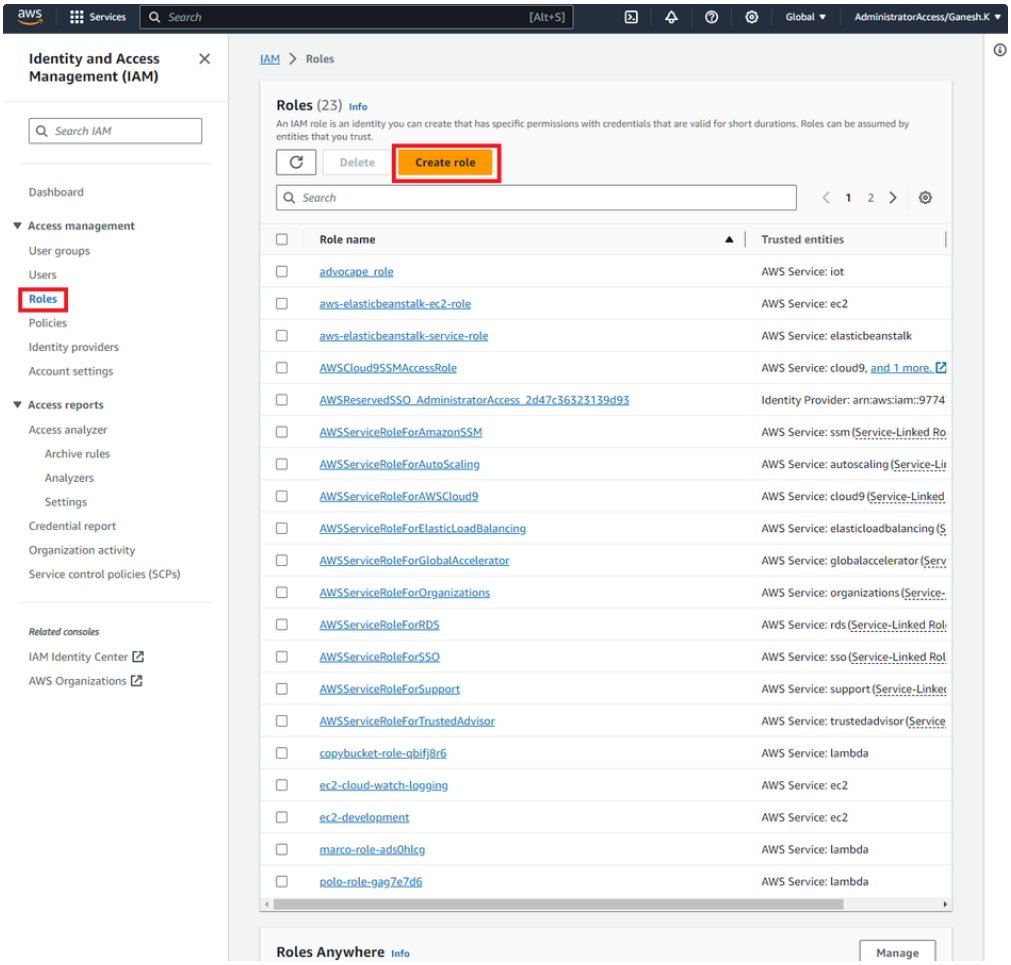


8. Once created, verify by searching the name in the field as shown below

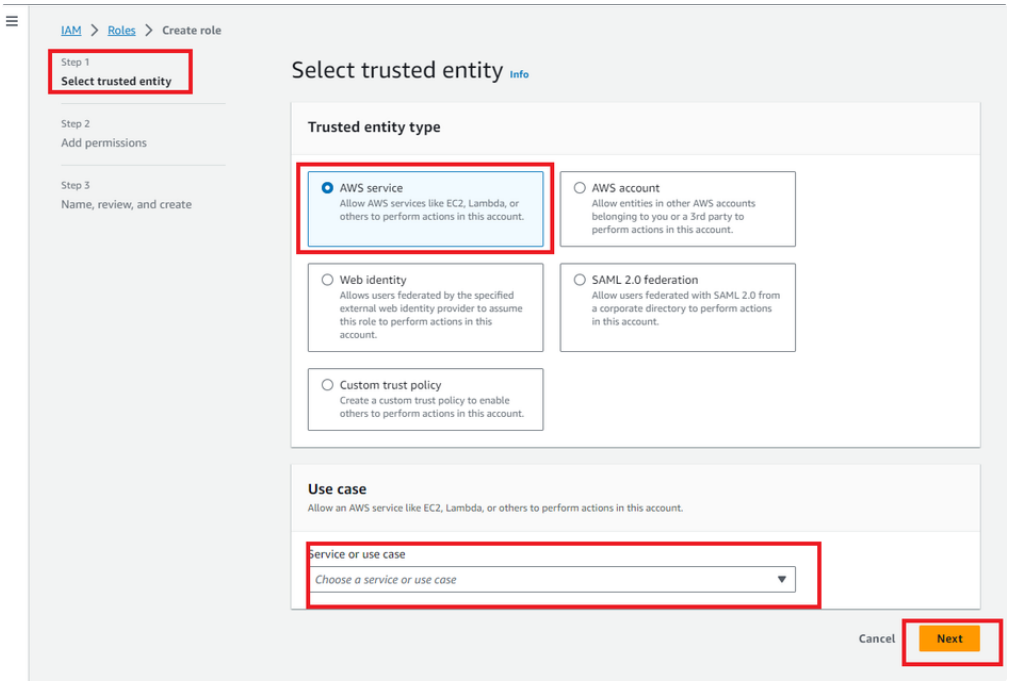


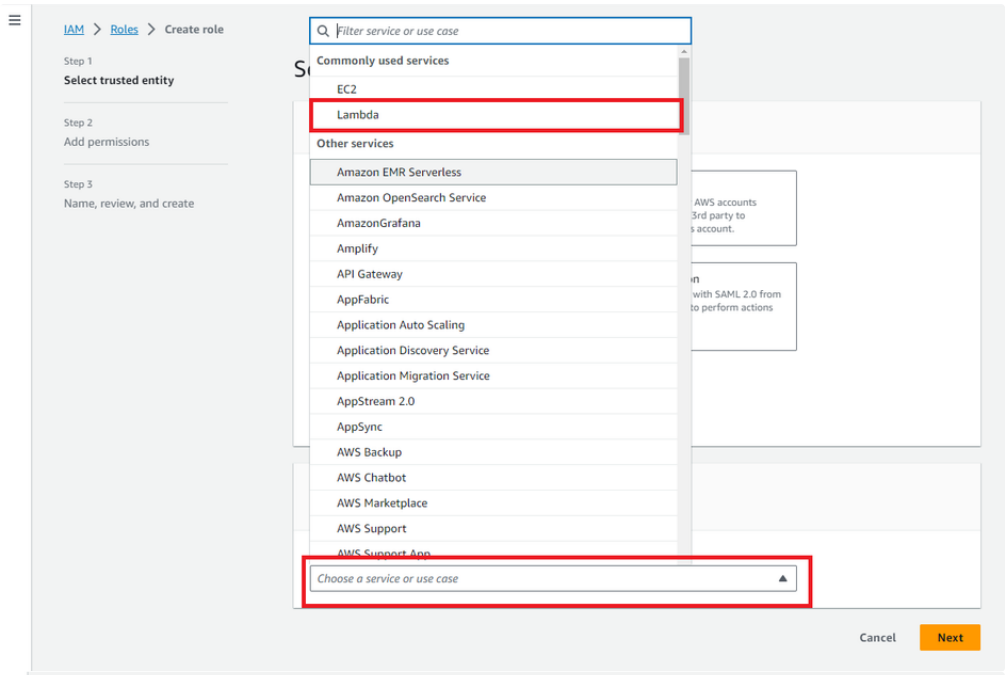
Step 4: Creating lambda-specific roles and attaching the policy

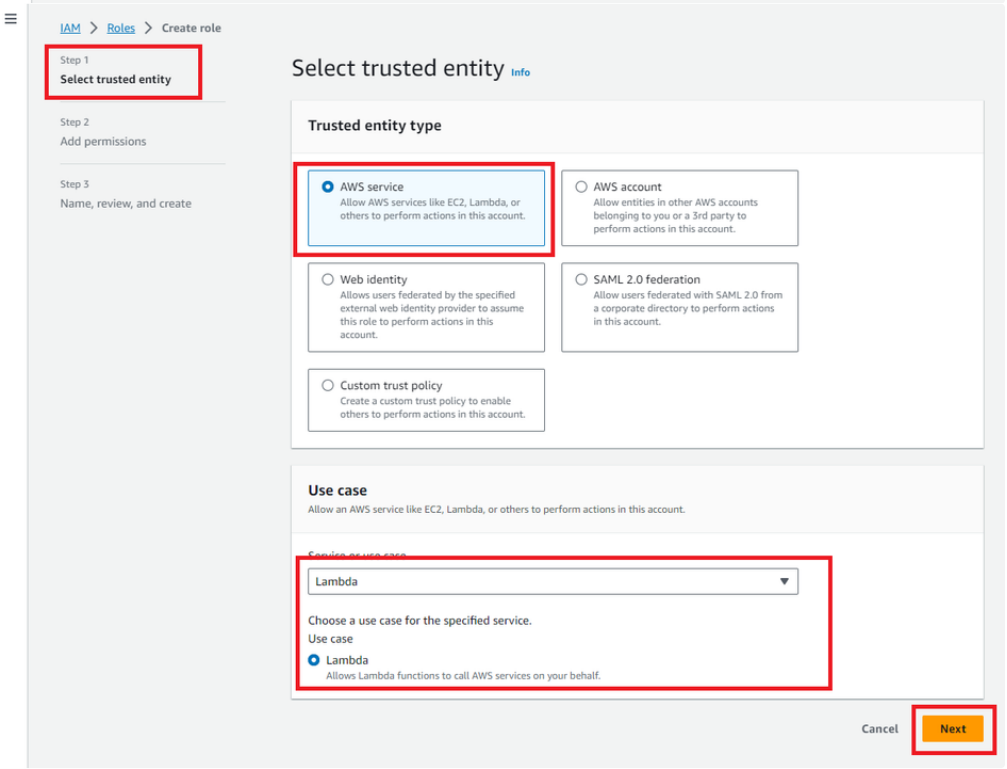
1. On the same **IAM** page, select **Roles**
2. Select **Create Role**



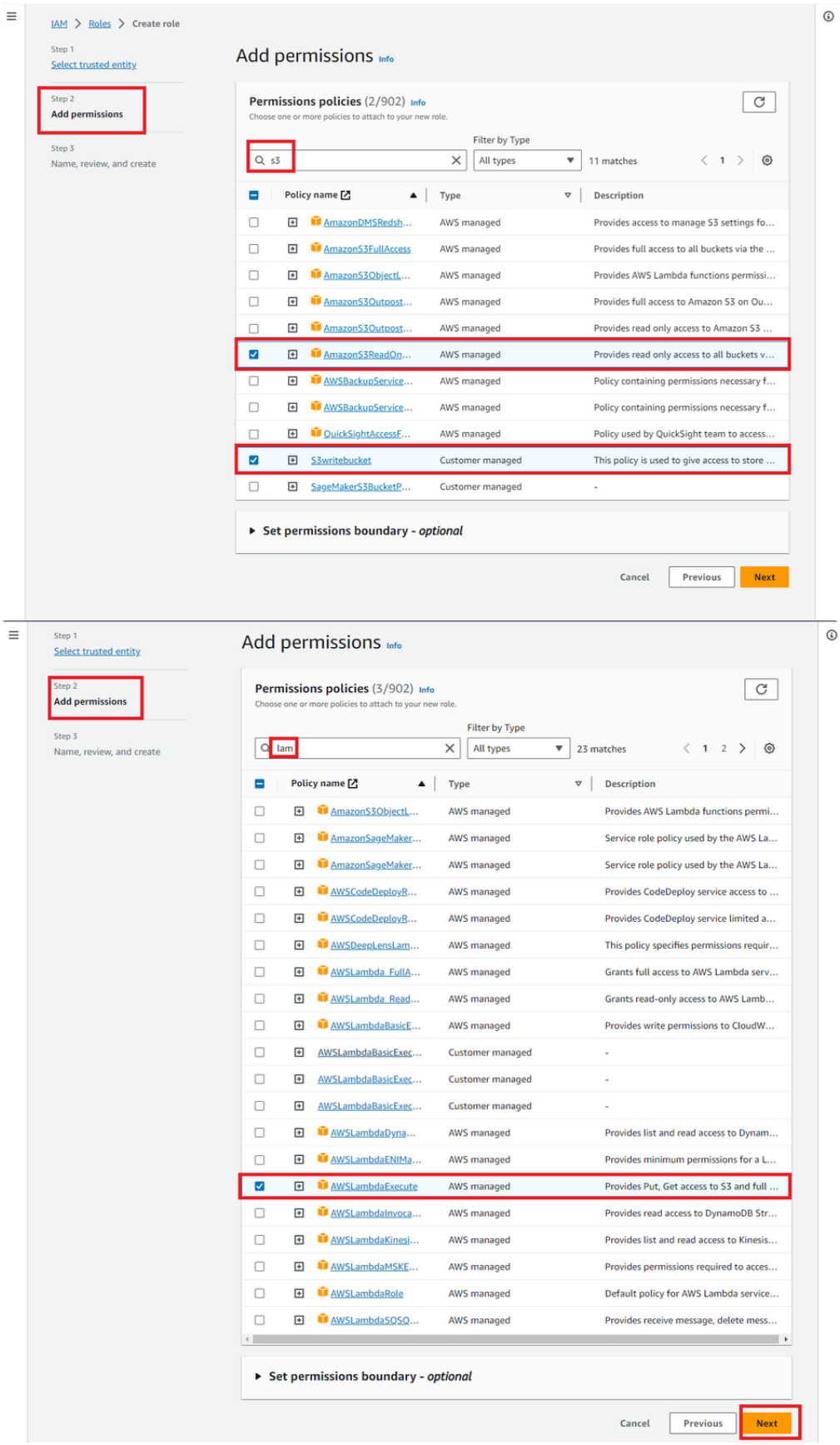
3. In the create role page, select **AWS service** and **choose service or use case as lambda,** then select **next**



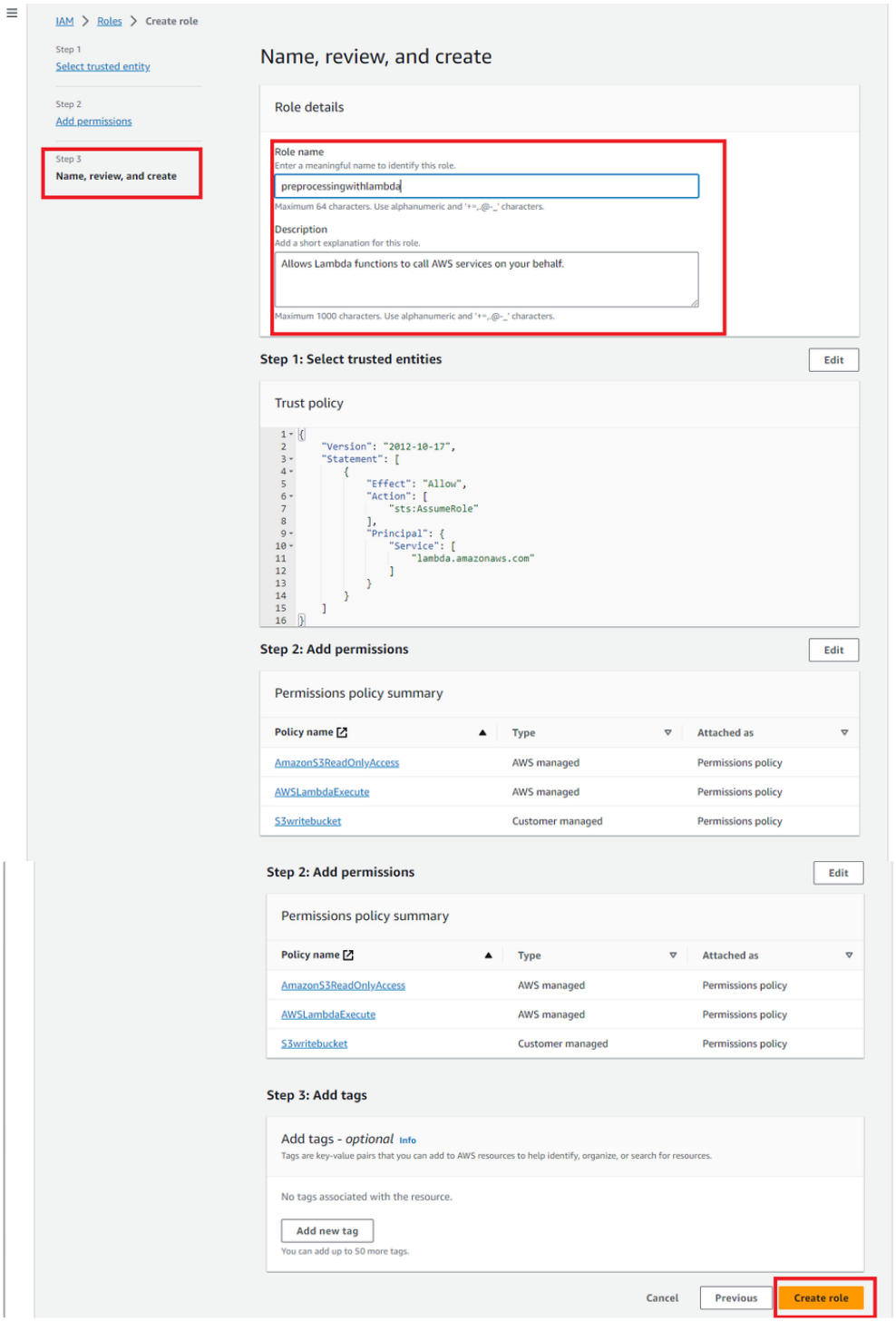




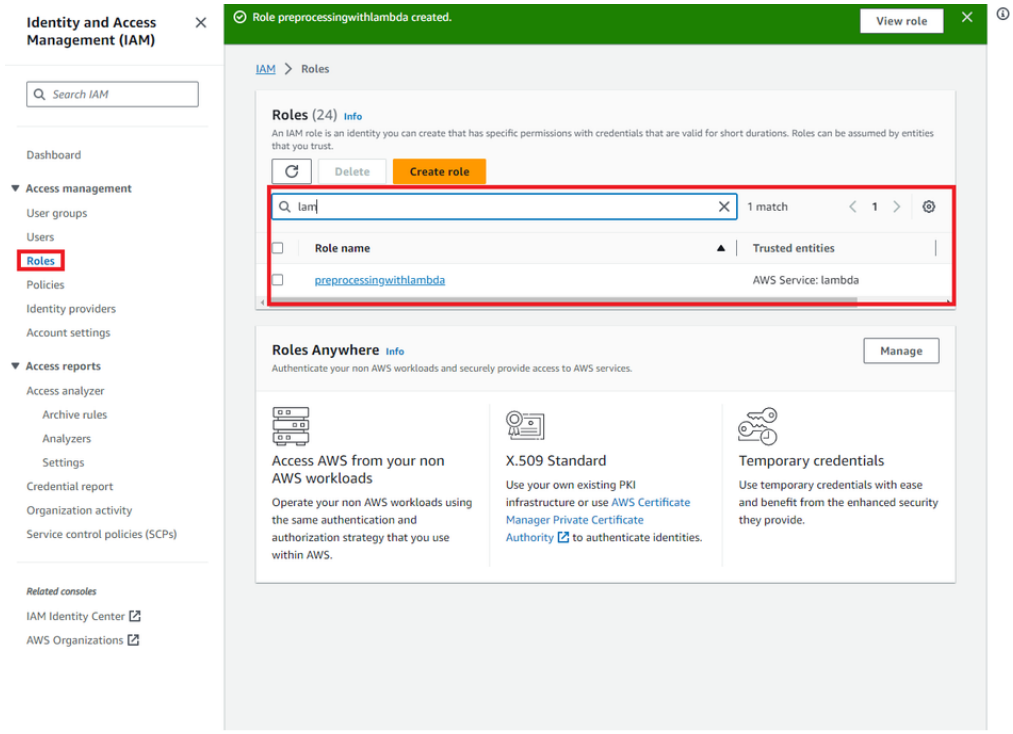
4. In the next add permission page, select three policies namely **S3ReadonlyAcess, S3writebucket, and AWSLambdaExecute,** then click **Next**



5. On the next page, give the role name, and description, then select **Create role**

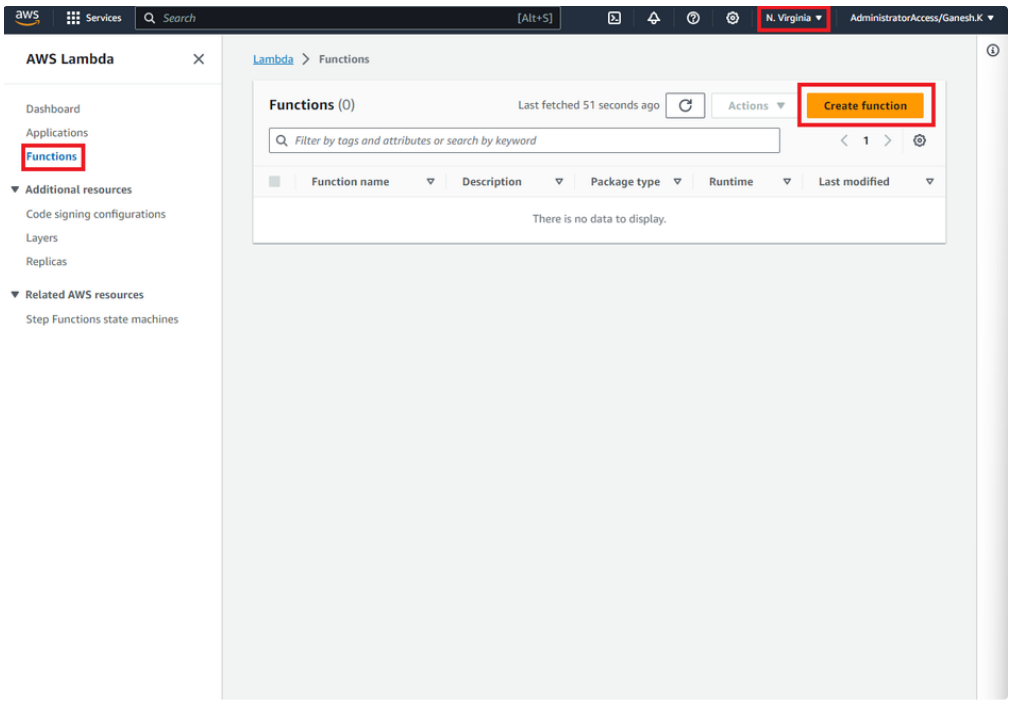


6. Once created, verify by searching the name in the field as shown below

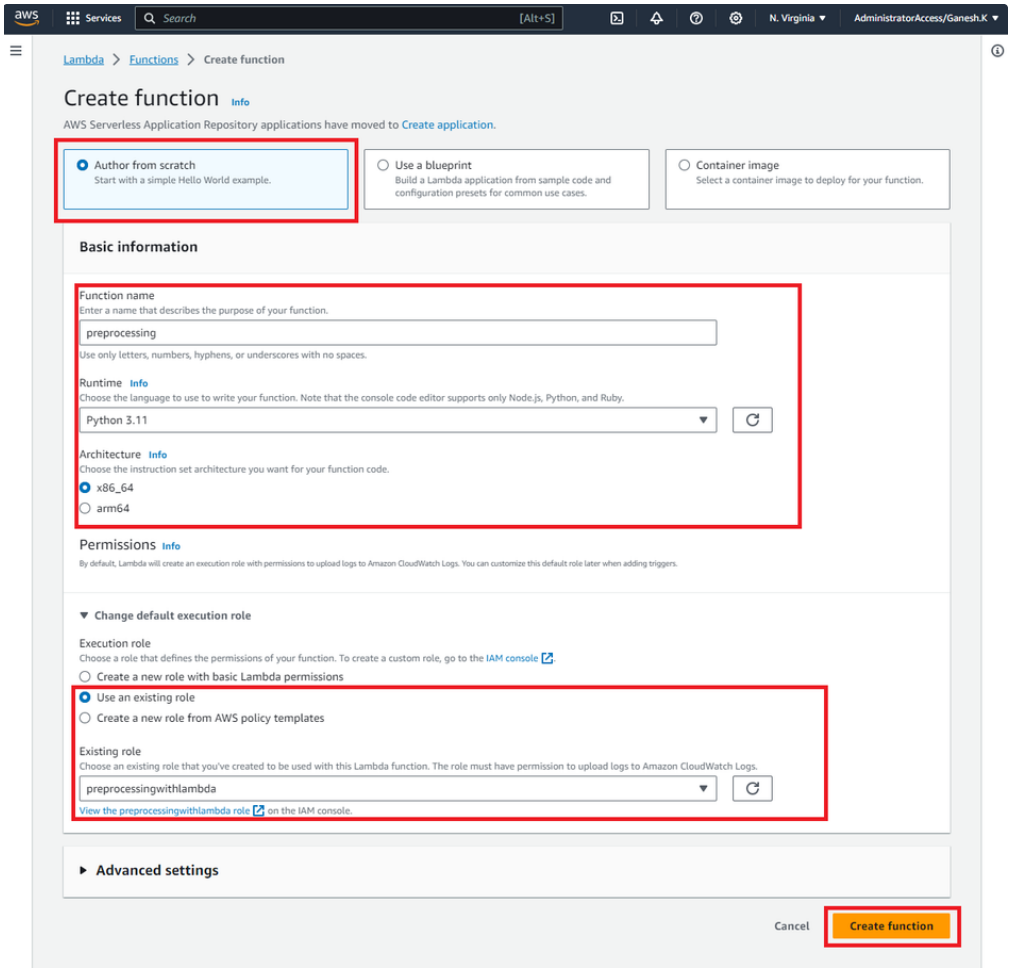


Step 5: Creating lambda function and setting trigger

1. Open the **Management Console** and select the **lambda** page
2. Select **Create Function**



3. In the creation function, choose an **author from scratch**, give a **function name**, select **runtime as python**, **architecture x86_64**, choose a role as **use an existing role** as created, then select **create function**

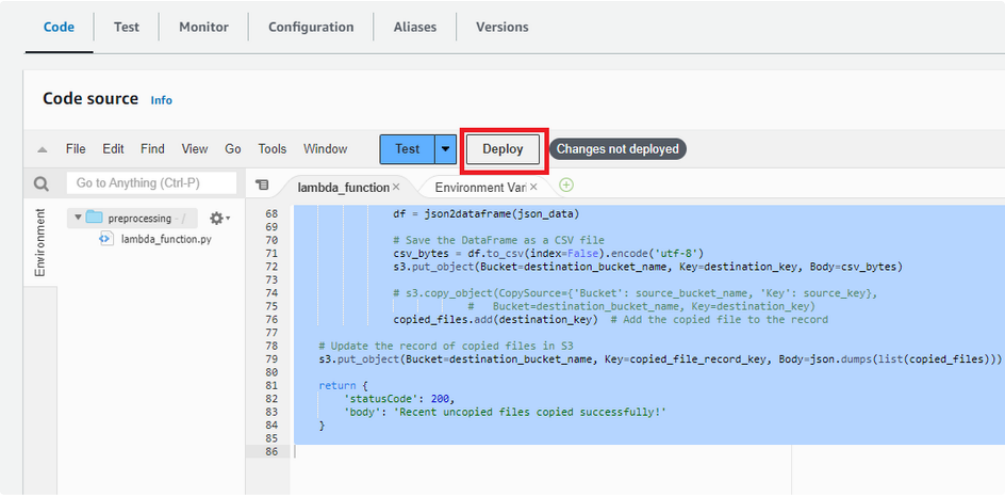


4. Paste the below code in the **code source on the next page**

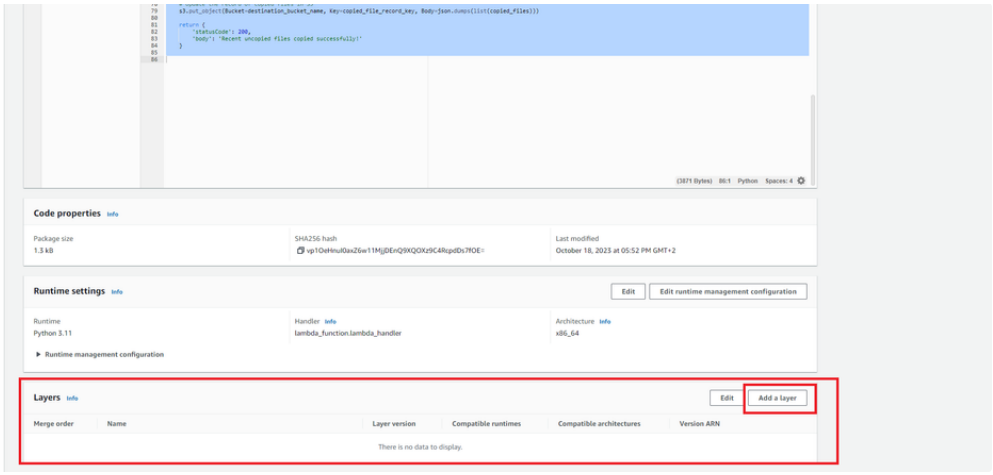
```python
import json
import boto3
import csv
import tempfile
import os
import pandas as pd

def calculate_accuracy(predictions, labels):
    correct_predictions = 0
    total_predictions = len(predictions)

    for pred, label in zip(predictions, labels):
        if pred == label:
            correct_predictions += 1

    return correct_predictions / total_predictions

def lambda_handler(event, context):
    # Initialize SageMaker runtime and S3 client
    sm_runtime = boto3.client('sagemaker-runtime')
    s3_client = boto3.client('s3')

    # Specify the SageMaker endpoint name
    endpoint_name = 'tuned-svm-endpoint'
```

```
25
26      # Define S3 bucket and file paths
27      input_bucket = 'bucket-inputdata'
28      input_file_key = 'test.csv'
29      output_bucket = 'bucket-processedata'
30      output_file_key = 'predictions.json'
31
32      # Download the input CSV file from S3 to a temporary directory
33      temp_dir = tempfile.mkdtemp()
34      input_file_path = os.path.join(temp_dir, 'input.csv')
35      s3_client.download_file(input_bucket, input_file_key, input_file_path)
36
37      # Read the CSV file into a pandas DataFrame
38      df = pd.read_csv(input_file_path)
39
40      # Extract the label column for accuracy calculation
41      labels = df['label']  # Assuming 'label' is the name of your label column
42
43      # Remove the label column before making predictions
44      data = df.drop(columns=['label'])
45
46      # Make a prediction request to the SageMaker endpoint
47      response = sm_runtime.invoke_endpoint(
48          EndpointName=endpoint_name,
49          ContentType='text/csv',
50          Body=data.to_csv(index=False, header=False)
51      )
52
53      # Extract the prediction result
54      prediction_result = json.loads(response['Body'].read())
55
56      # Calculate accuracy manually
57      accuracy = calculate_accuracy(prediction_result, labels)
58
59      # Save the predictions and accuracy to a JSON file
60      output_data = {
61          "predictions": prediction_result,
62          "accuracy": accuracy
63      }
64      output_json = json.dumps(output_data)
65
66      # Upload the JSON file to the output S3 bucket
67      s3_client.put_object(
68          Bucket=output_bucket,
69          Key=output_file_key,
70          Body=output_json,
71          ContentType='application/json'
72      )
73
74      return {
75          'statusCode': 200,
76          'body': json.dumps("Predictions and accuracy saved to S3")
77      }
78
79
```

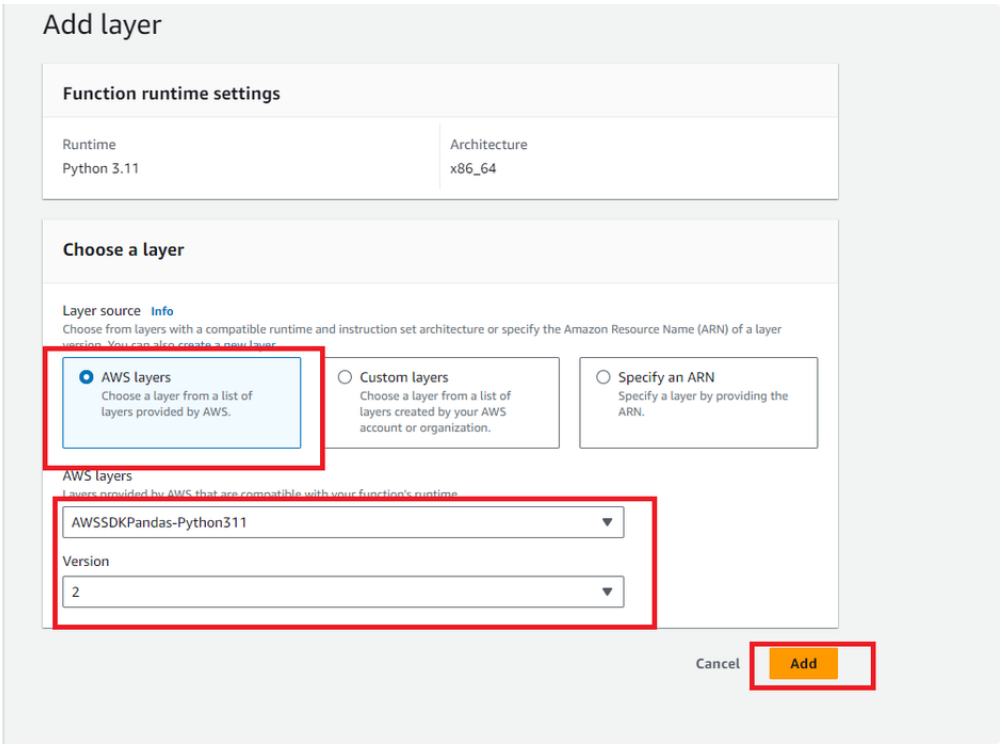
5. Once the code is pasted, click **Deploy**



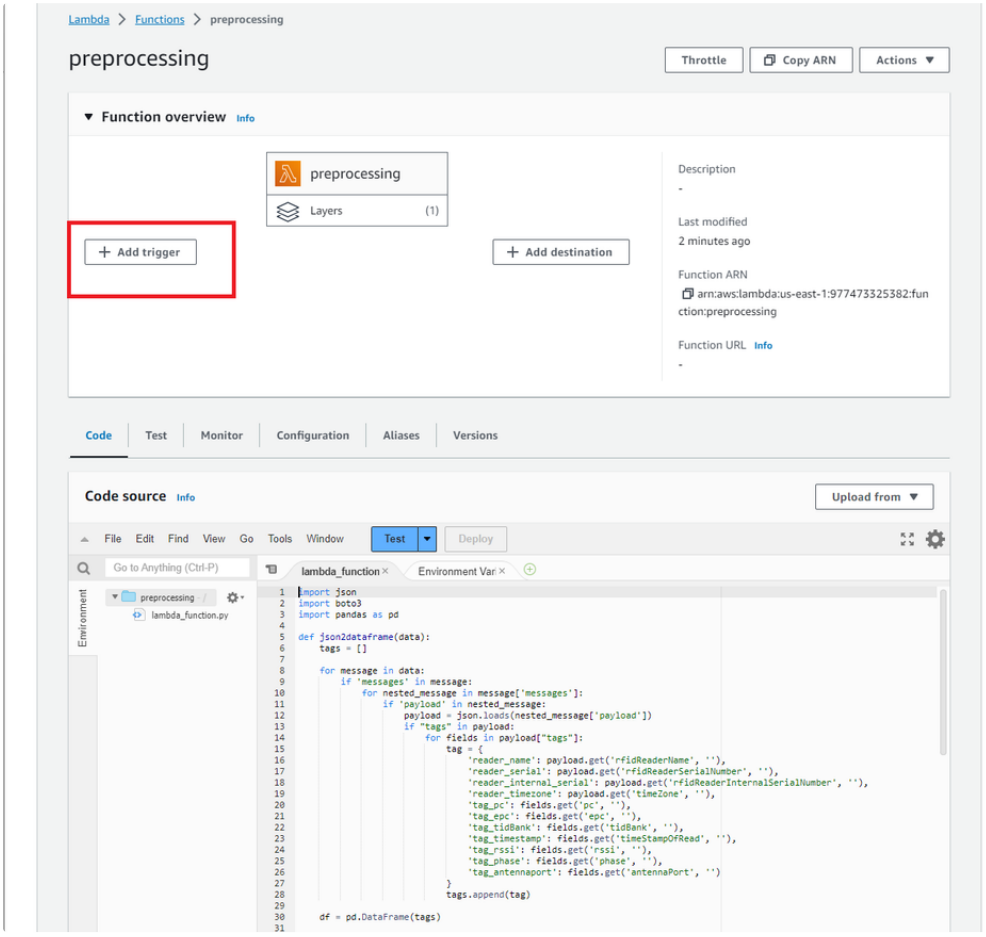
6. Now scroll down to the layers, click **add layer**



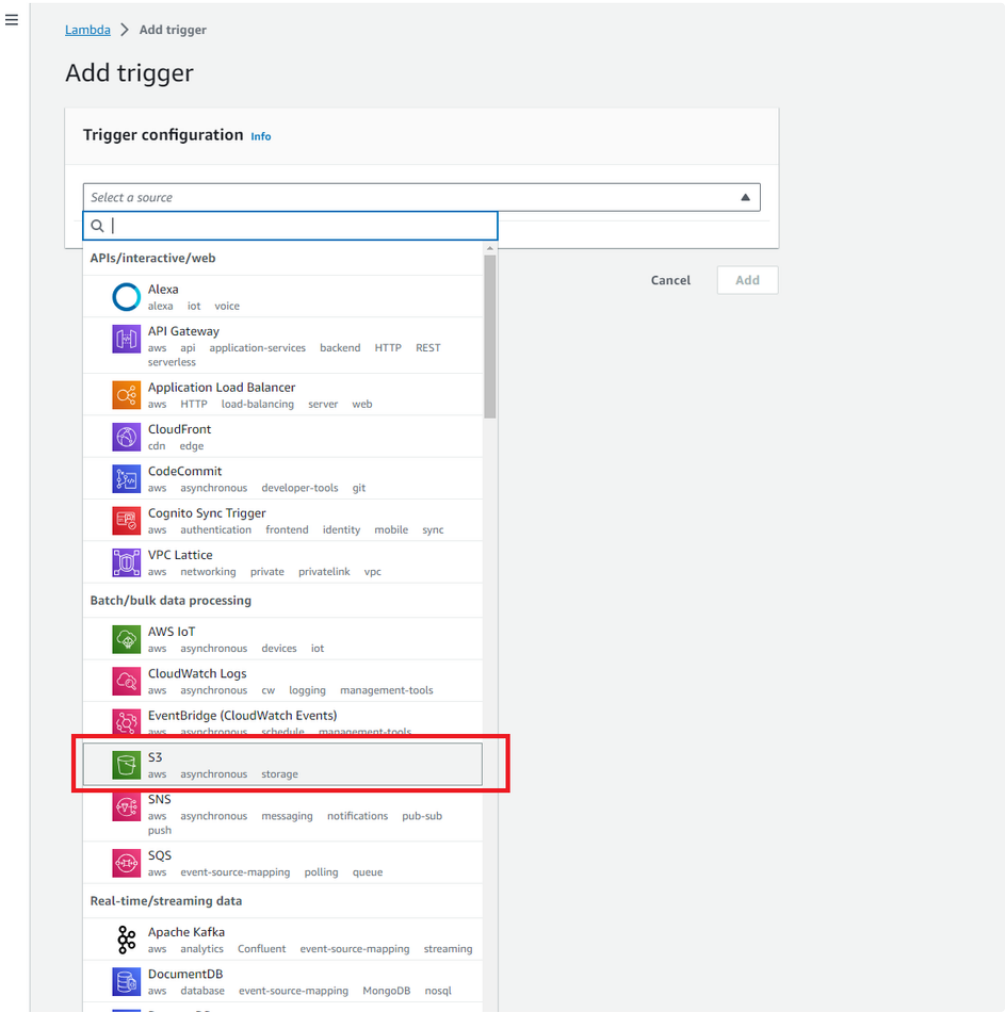
7. In the layer page, select **pandas** and version, then select **add**



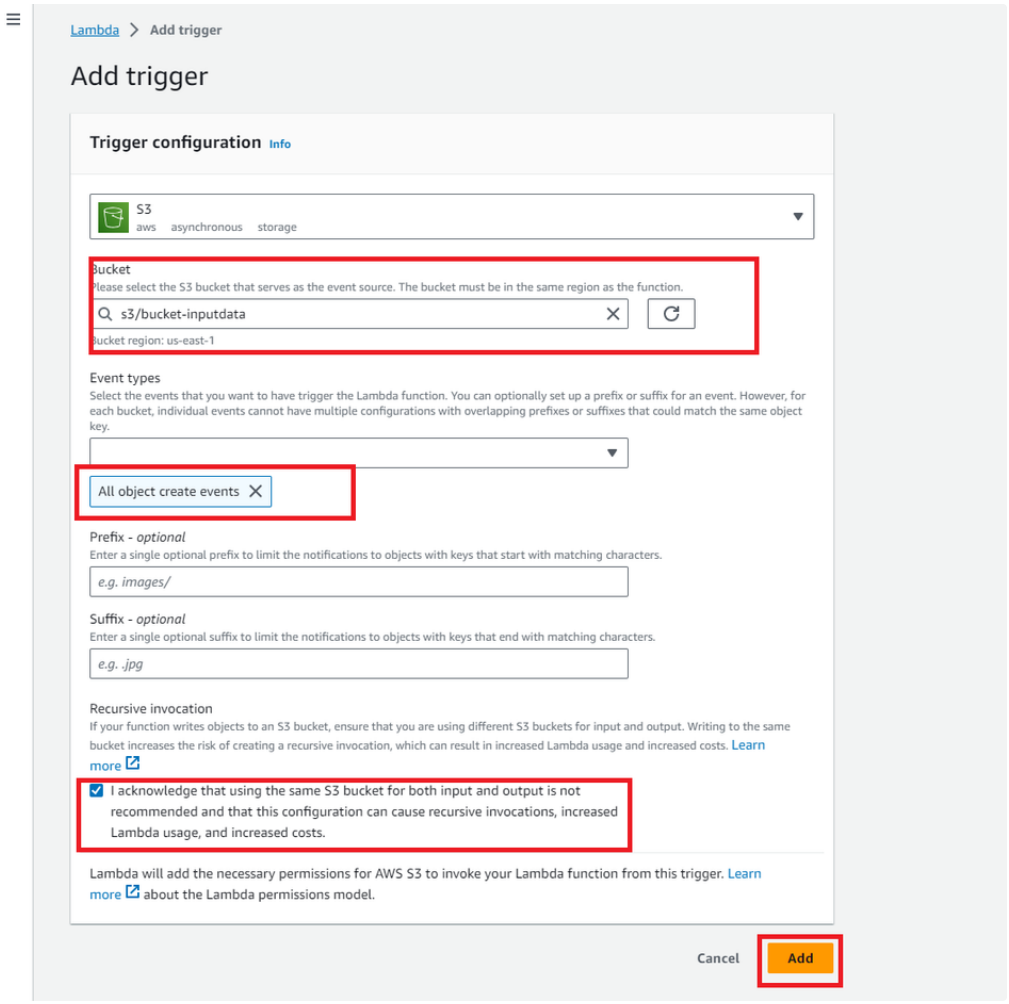
8. Now in the **function overview** click **Add trigger**



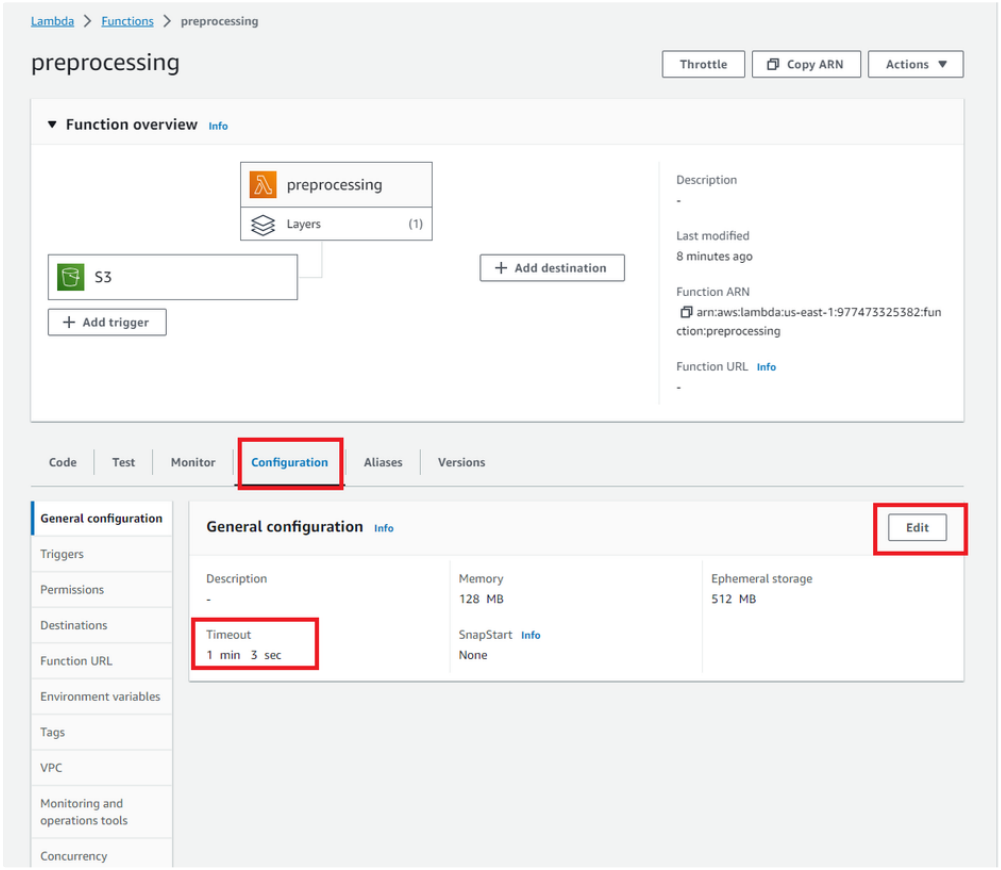
9. Select the source **S3** on the next page then click **next**

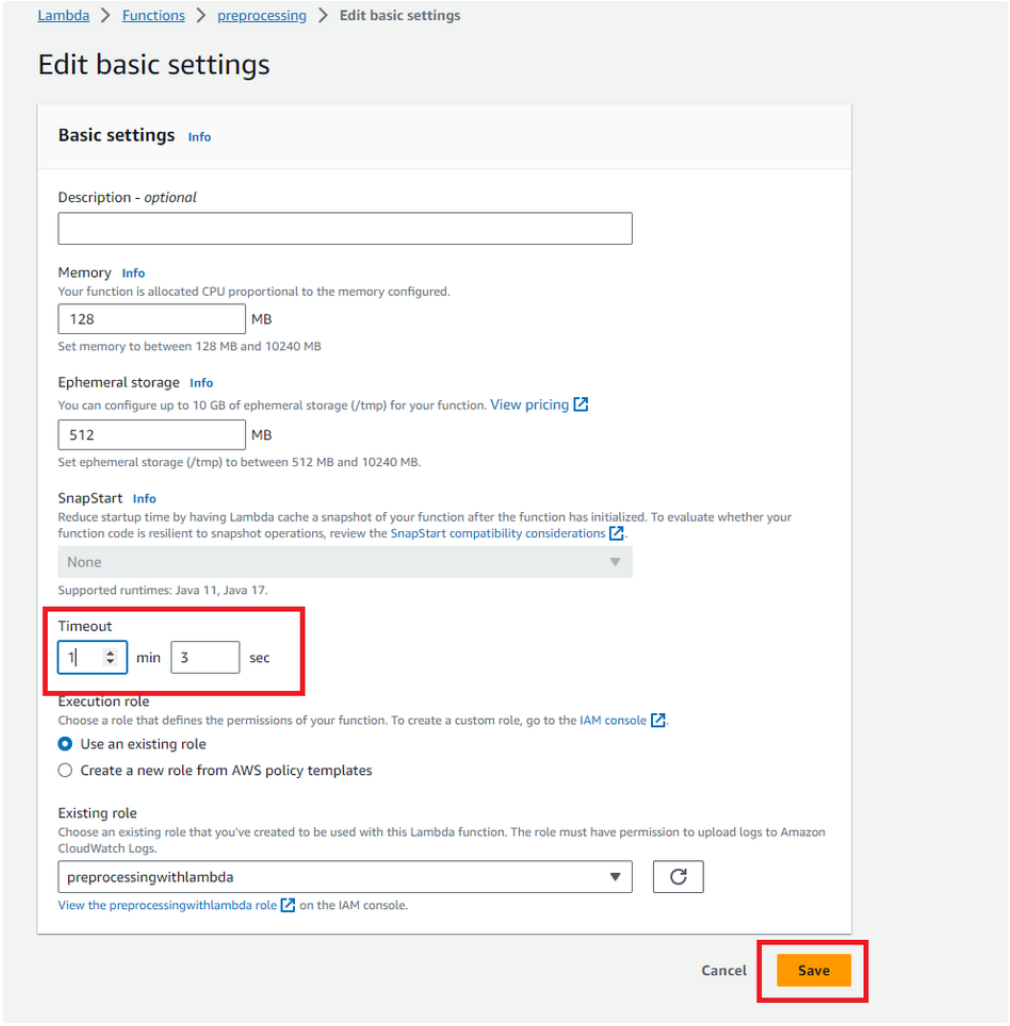


10. On the next page, select the **input bucket**, check the **acknowledge** then select **Add**



11. Goto **configuration tab on the lambda function** page and choose **edit in General configuration** as shown below



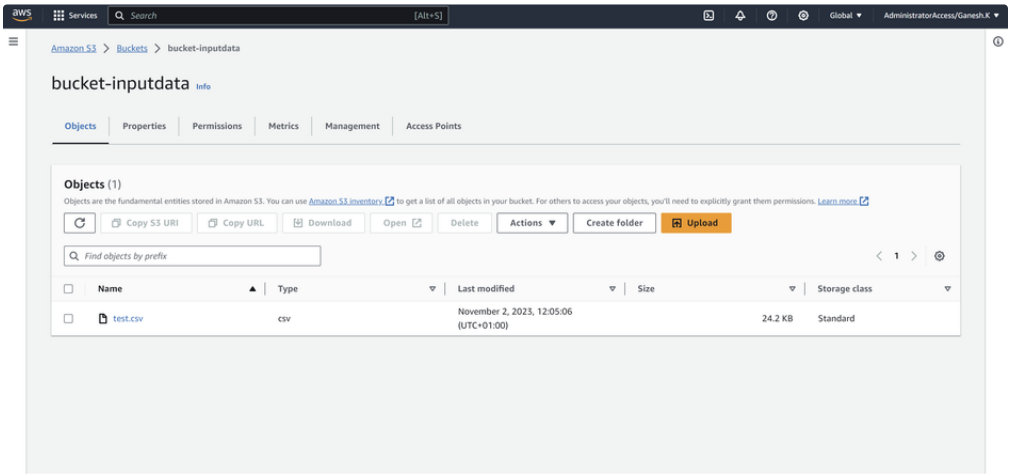


12. After changing the **time click save**

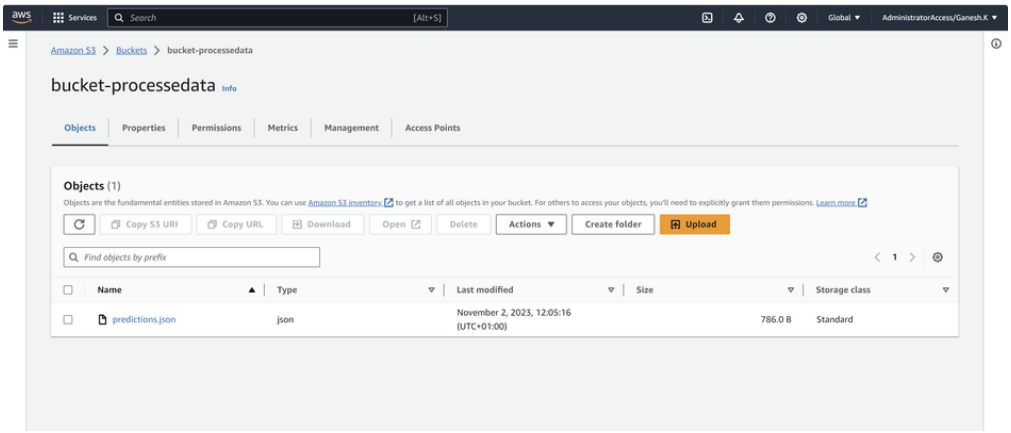
    If we don't change in case the execution will take more time than specified the function will stop due to timeout

Checking

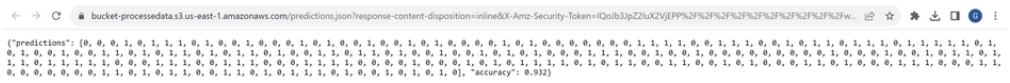
1. Now upload a **test.csv** file in the bucket-inputdata



2. You should be able to see the **predictions.json** that stores the predictions and accuracy that have been processed using the sagemaker endpoint.



3. When you open the file, you could able to see the output predictions and accuracy



4. In case it didn't give the required output see the error in Cloudwatch (the code given in the document works fine)