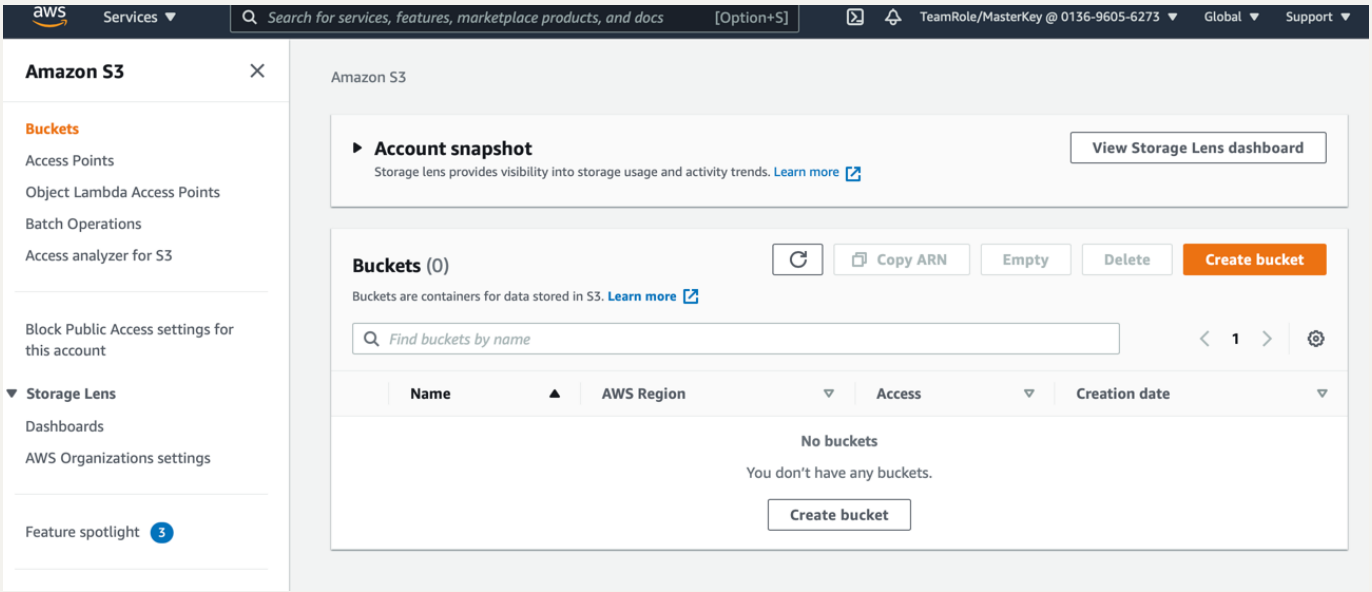
Automating pipelines with new data uploads

This sesssion shows use of AWS Lambda integrated with S3 to automate a pipeline when new training data is uploaded to the S3 bucket.

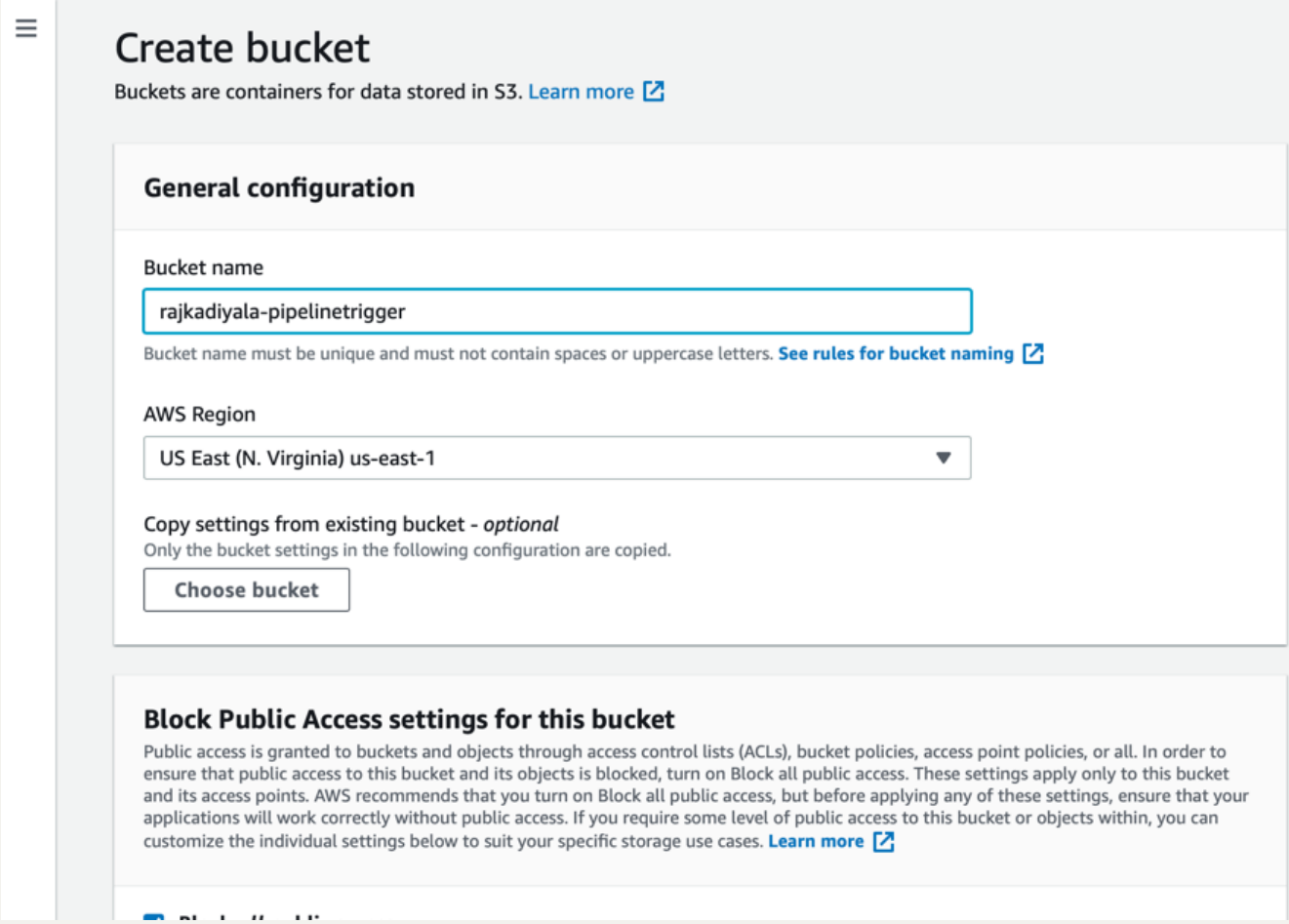
Create a S3 bucket

Type S3 in the search bar and click on Amazon S3.

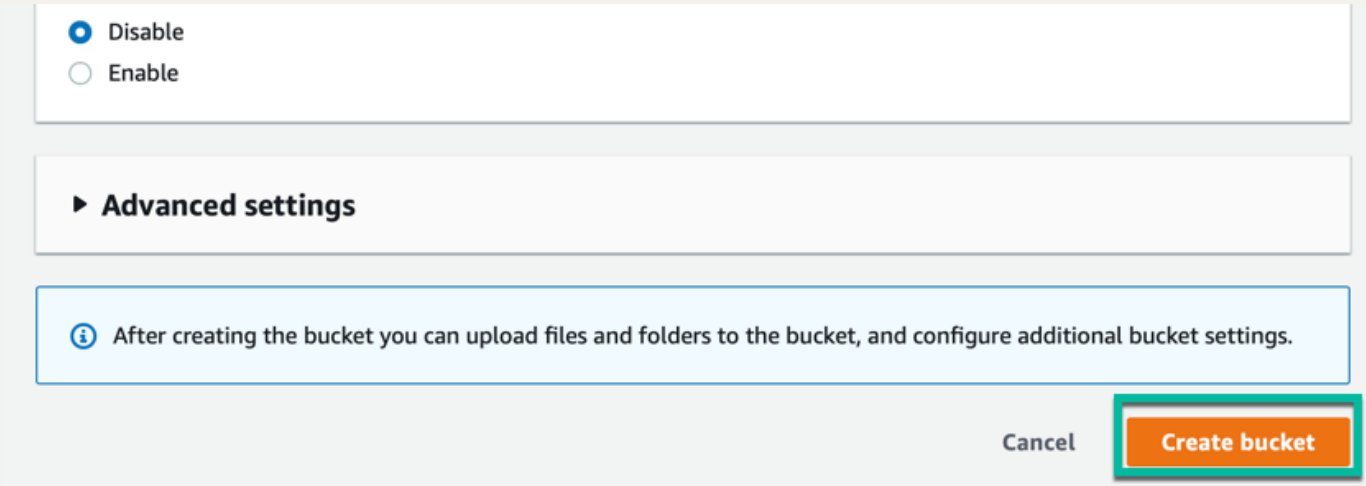


Click on Create Bucket and use the naming format your **firstlastname**-pipelinetrigger

Note: Bucket names have to be globally unique, so if you happen to have a conflict first and last names, use some other string of your choice.



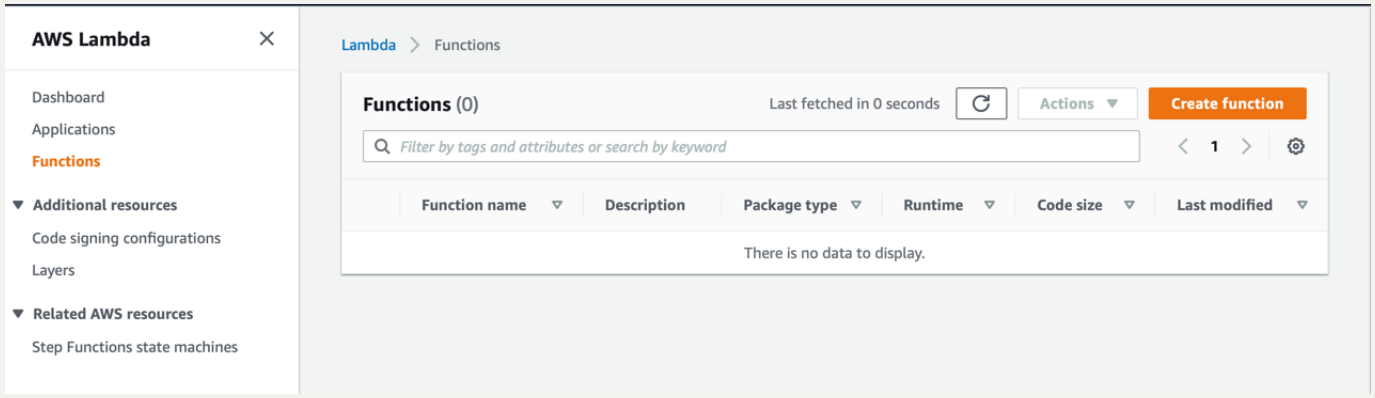
Leave everything else at default options and click create bucket.



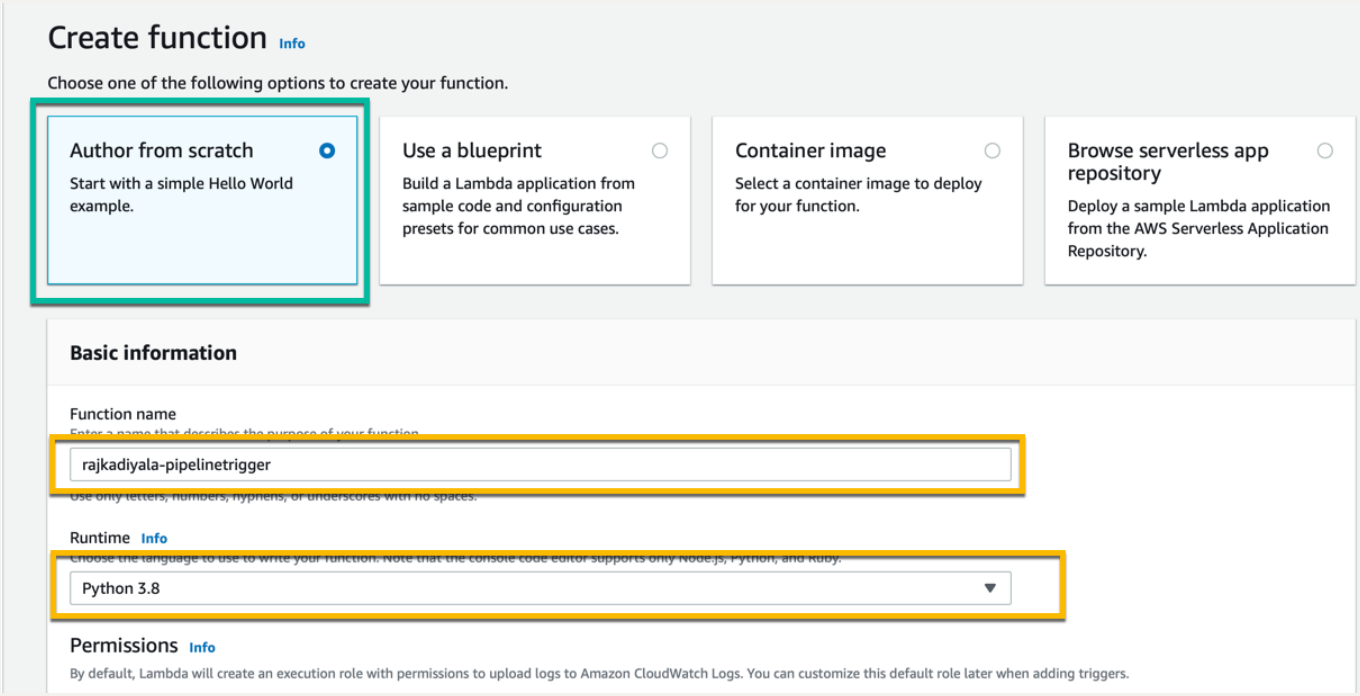
Click on your bucket and create two folders called *input* and *batch*.

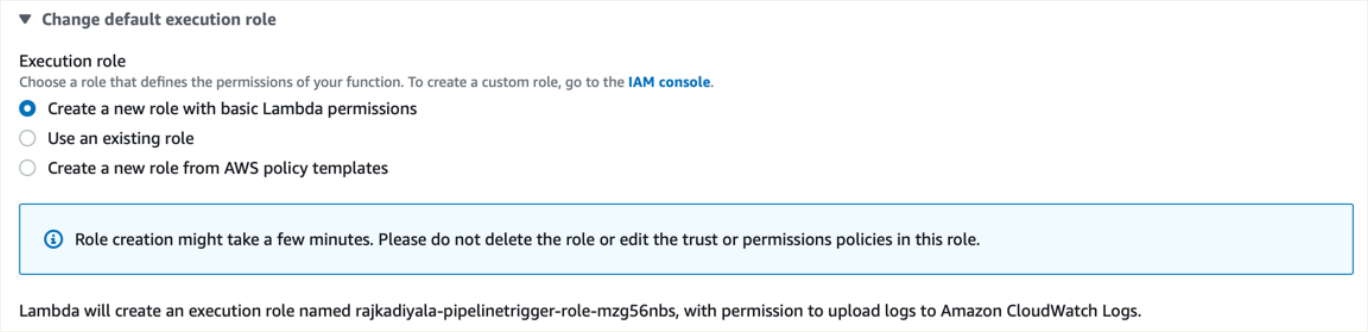
Create a lambda function

Type lambda in the search bar and click on AWS Lambda.



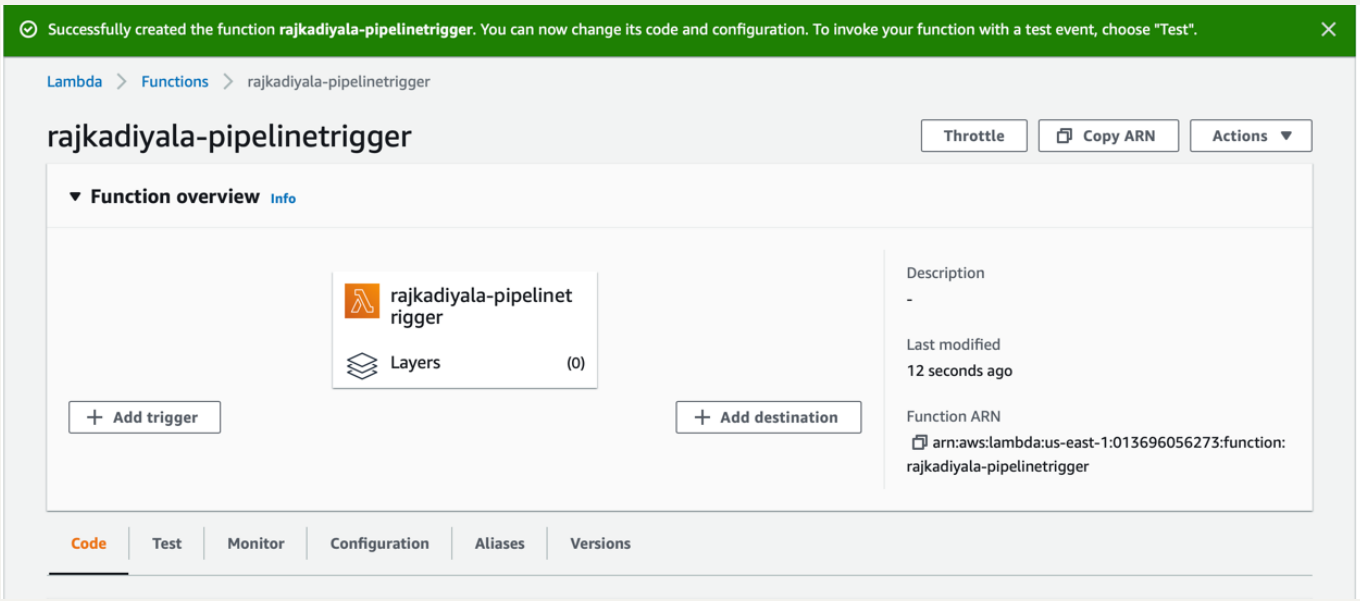
Click on create function and select the options below and provide a name for your function.

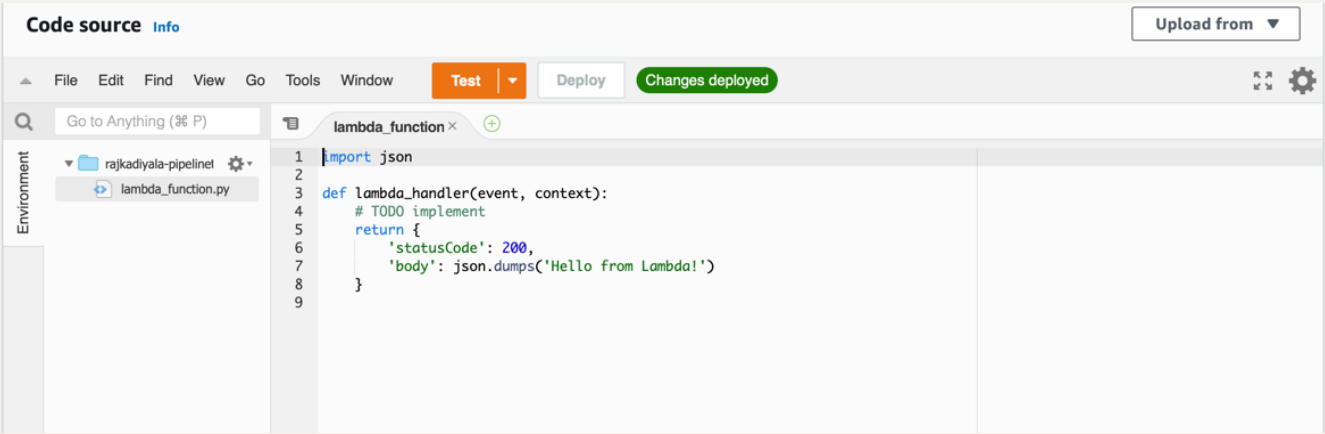




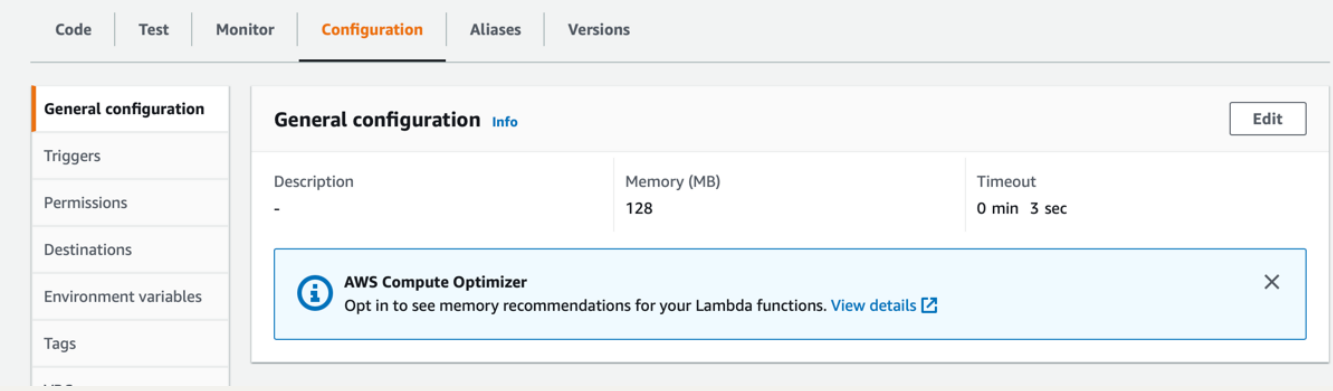
This will create a default role for your function to which we will add permissions to. Click on Create Function

The function screen will look like below

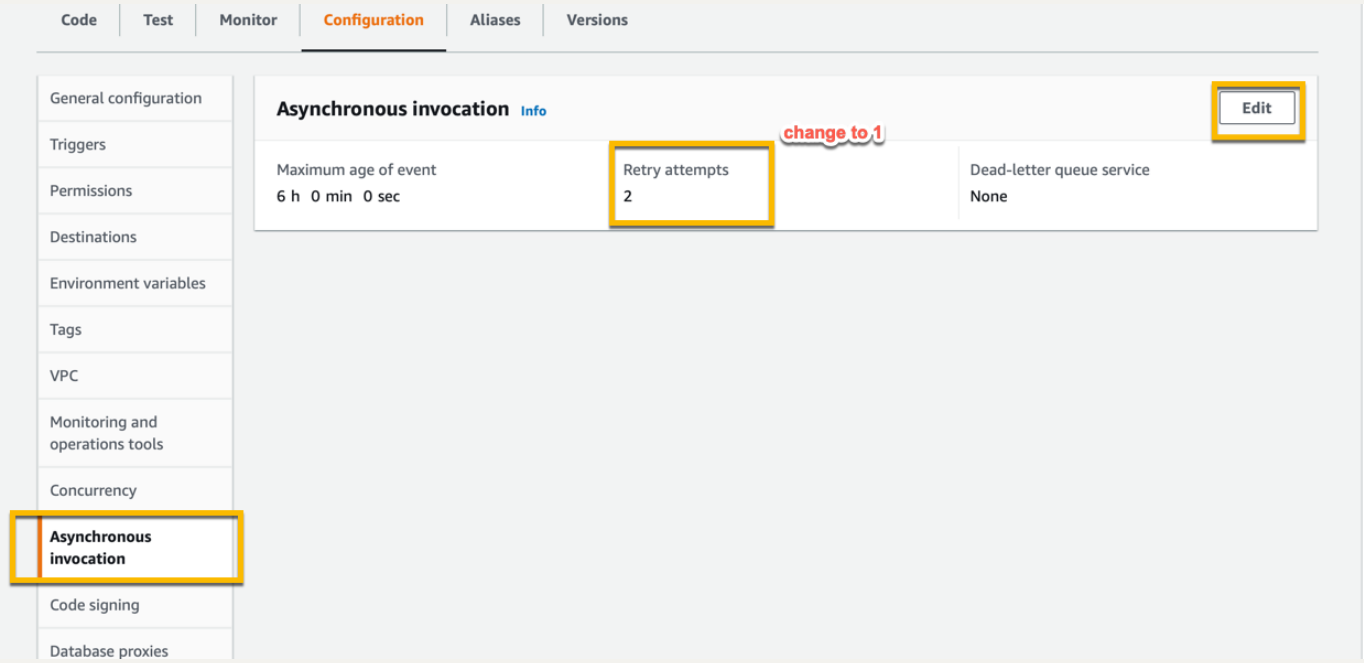




Edit the configuration and change the time out from 3 sec to 5min



Change the retry attempts to 1 for the sake of workshop and initial testing.



Update the code for the lambda function, replace xxxxxx with your account number shown at the top of your console screen. Also pay attention to pipeline name in SageMaker Studio and ensure the name matches.


```
import json
import boto3
import urllib.parse
import uuid

sm = boto3.client("sagemaker")

s3 = boto3.resource('s3')


def lambda_handler(event, context):
    # using s3 object that triggered the pipeline
```

```
    # read the object, copy the file to default bucket
    # and update the inputData param

    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']
['object']['key'], encoding='utf-8')

    print("bucket:" + str(bucket))
    print("key:" + str(key))

    print("copying input")
    copy_source = {
        'Bucket': bucket,
        'Key': key
    }
    s3.meta.client.copy(copy_source, 'sagemaker-us-east-1-
xxxxxxxxxx', 'input/abalone-dataset.csv')

    print("copying batch")
    copy_source = {
        'Bucket': bucket,
        'Key': 'batch/abalone-dataset-batch'
    }
    s3.meta.client.copy(copy_source, 'sagemaker-us-east-1-
xxxxxxxxxxx', 'batch/abalone-dataset-batch')

    response = sm.start_pipeline_execution(
        PipelineName='AbalonePipeline',
        PipelineExecutionDisplayName='AbalonePipelineTriggered',
        PipelineParameters=[
            {
                'Name': 'InputData',
                'Value': 's3://sagemaker-us-east-1-
xxxxxxxxxx/input/abalone-dataset.csv'
            },
```

```
            {'Name': 'BatchData',
             'Value': 's3://sagemaker-us-east-1-
xxxxxxxx/batch/abalone-dataset-batch'

             },
        ],
        PipelineExecutionDescription='triggeredbys3',
        ClientRequestToken=str(uuid.uuid4())
    )

    print(response)

    return {
        'statusCode': 200,
        'body': json.dumps('Completed Pipeline Trigger')
    }
```

Copy the code to code section in AWS Lambda console

File Edit Find View Go Tools Window Test Deploy Changes not deployed

Go to Anything (⌘ P)

Environment

rajkadiyala-pipeline1

lambda_function.py

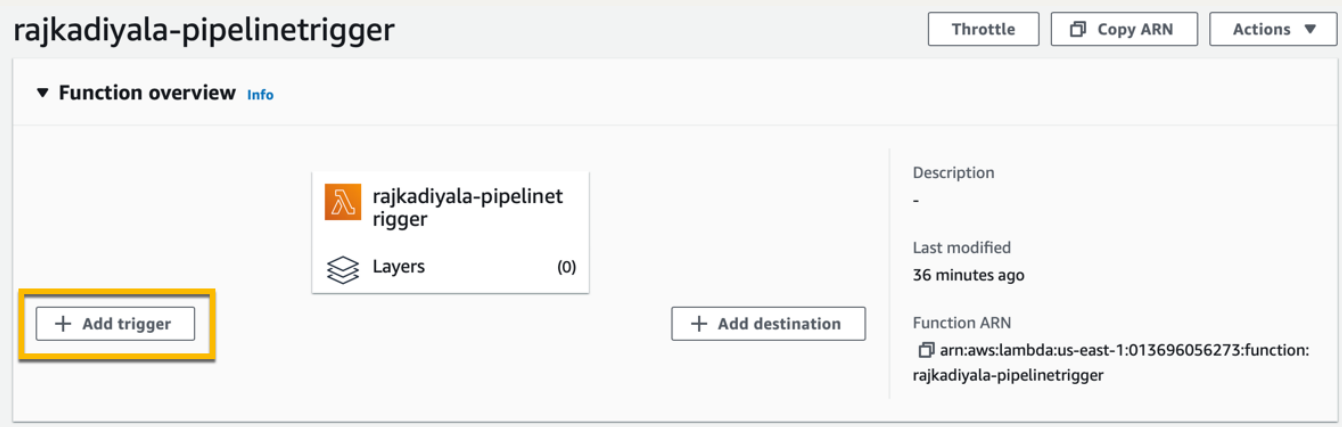
lambda_function

```python
import json
import boto3
import urllib.parse
import uuid

sm = boto3.client("sagemaker")

s3 = boto3.resource('s3')


def lambda_handler(event, context):
    # using s3 object that triggered the pipeline
    # read the object, copy the file to default bucket
    # and update the inputData param

    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')

    print("bucket:" + str(bucket))
    print("key:" + str(key))

    print("copying input")
    copy_source = {
        'Bucket': bucket,
        'Key': key
    }
    s3.meta.client.copy(copy_source, 'sagemaker-us-east-1-013696056273', 'input/abalone-dataset.csv')

    print("copying batch")
    copy_source = {
        'Bucket': bucket,
        'Key': 'batch/abalone-dataset-batch'
    }
    s3.meta.client.copy(copy_source, 'sagemaker-us-east-1-013696056273', 'batch/abalone-dataset-batch')

    response = sm.start_pipeline_execution(
        PipelineName='AbalonePipeline',
        PipelineExecutionDisplayName='AbalonePipelineTriggered',
        PipelineParameters=[
            {
                'Name': 'InputData',
                'Value': 's3://sagemaker-us-east-1-013696056273/input/abalone-dataset.csv'
            },

            {'Name': 'BatchData',
             'Value': 's3://sagemaker-us-east-1-013696056273/batch/abalone-dataset-batch'

            },
        ],
        PipelineExecutionDescription='triggeredbys3',
        ClientRequestToken=str(uuid.uuid4())
    )

    print(response)

    return {
        'statusCode': 200,
        'body': json.dumps('Completed the Trigger!')
    }
```

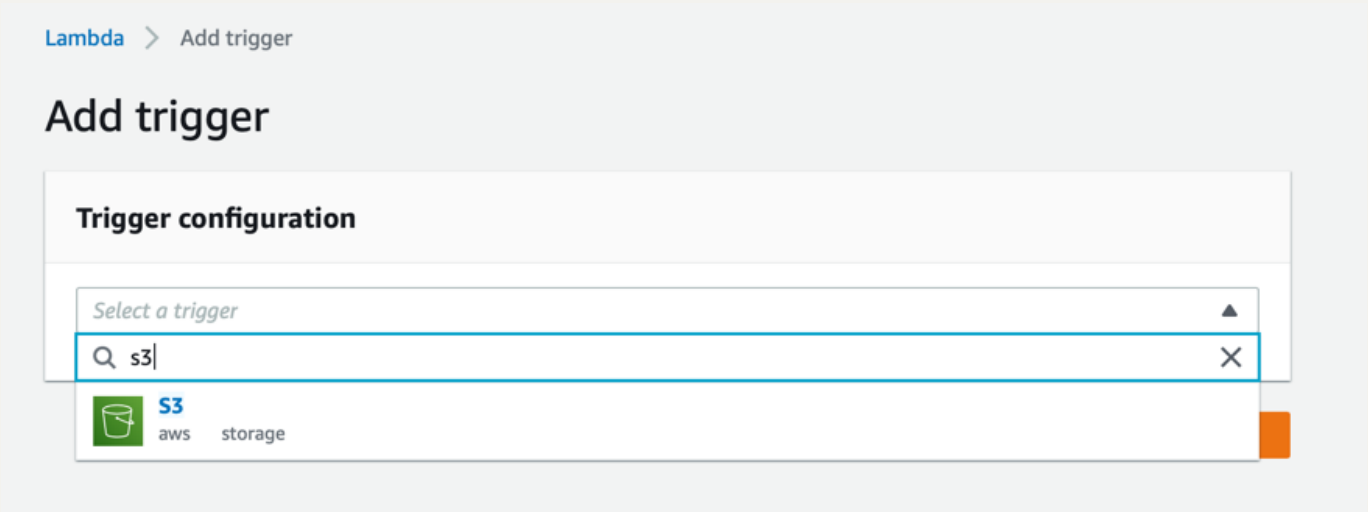
click file and save.

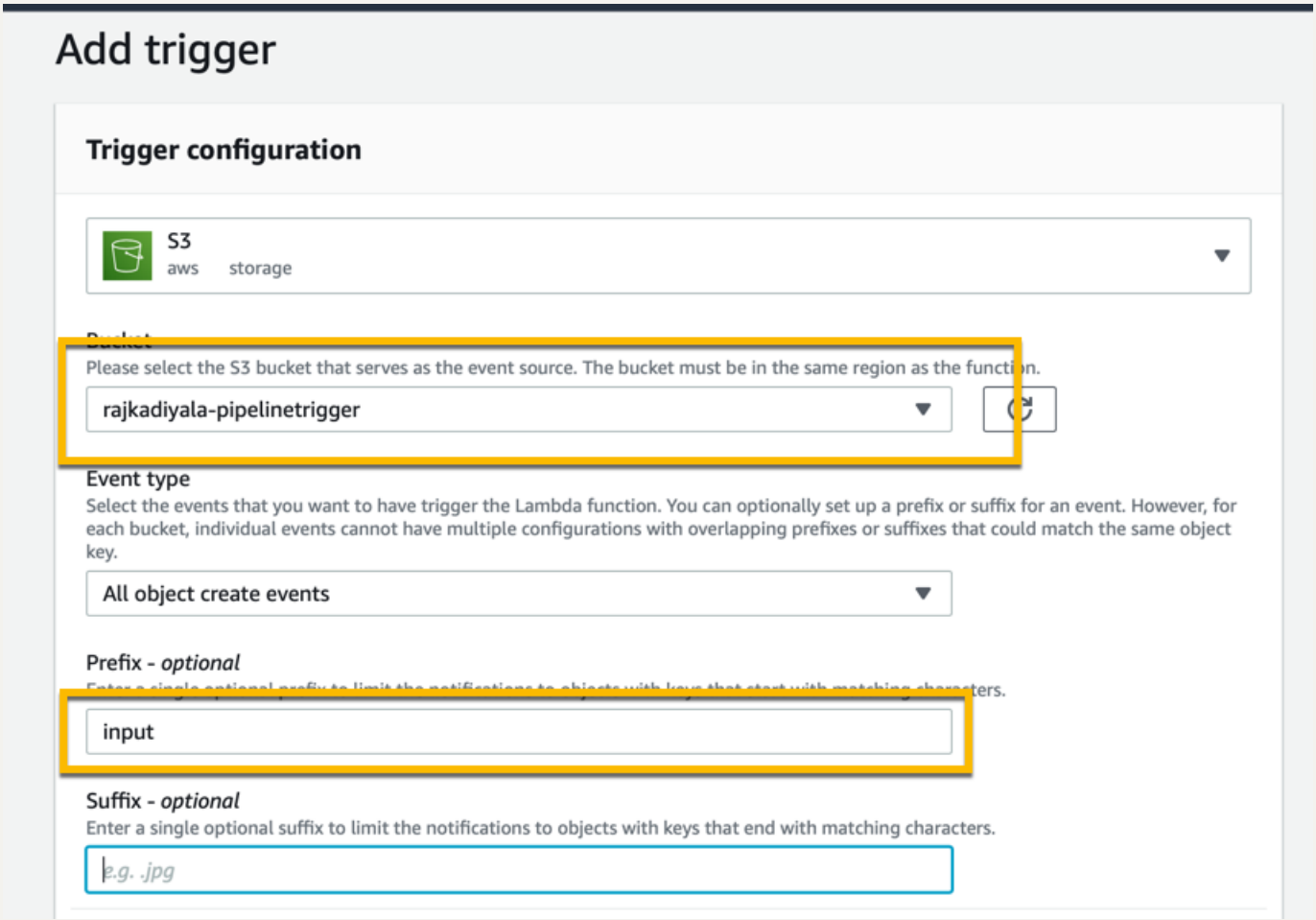
S3 Trigger for Lambda

Scroll to the top of lambda console and click on Add trigger.

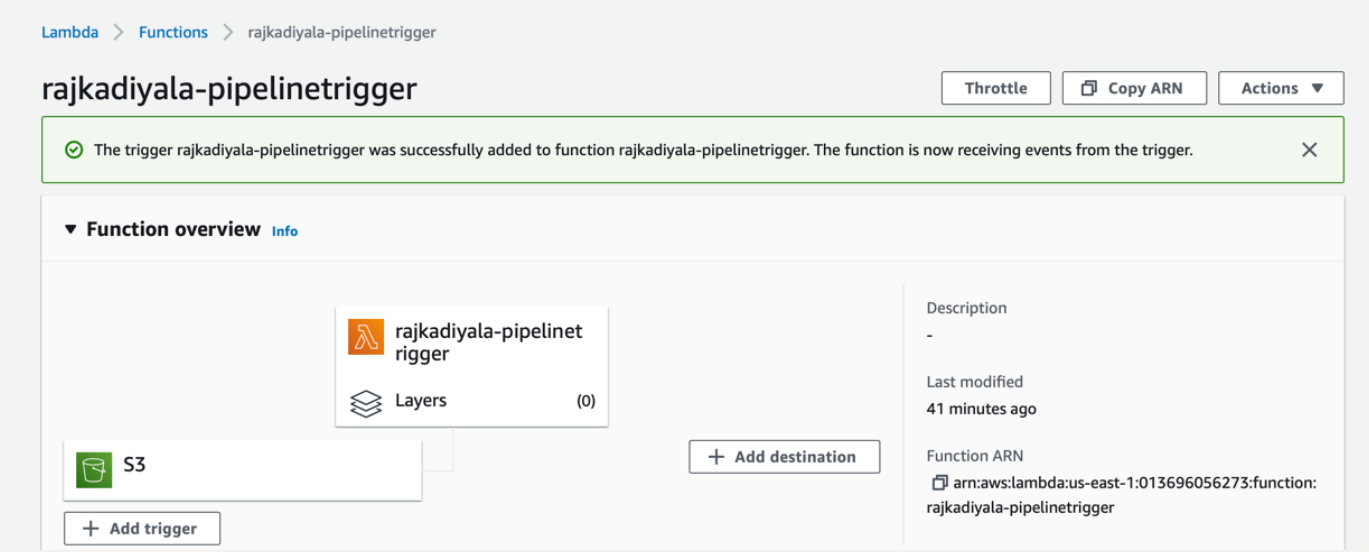


Type in S3 and click on S3 in drop down

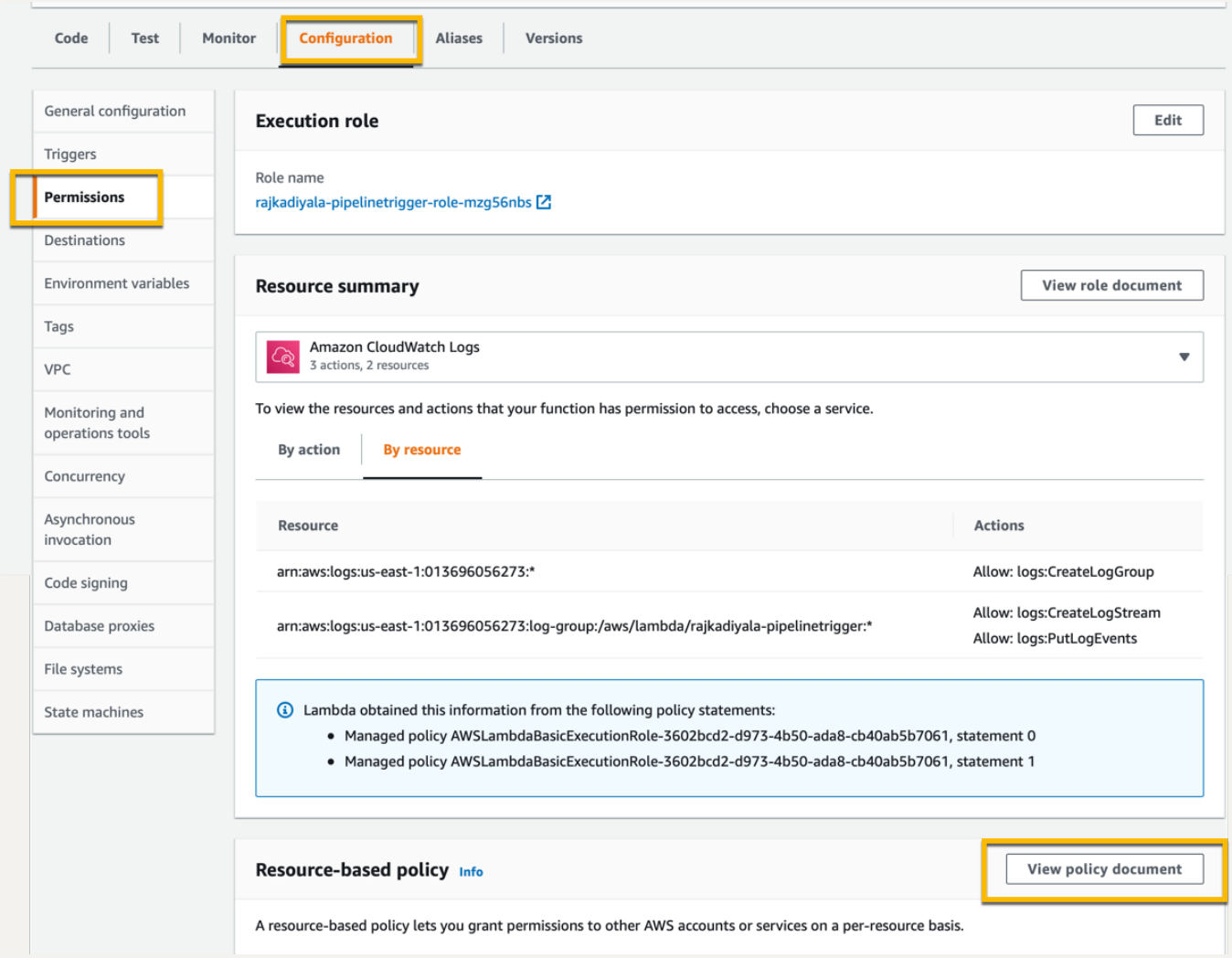




Acknowledge that you are not going to use a single bucket for read and write operations to avoid recursive calls and click add. you will be returned to your lambda console and now can see the S3 set up as a trigger.



Take a lake at what adding the trigger has done, First Lambda has added a resource based policy to allow the Lambda function to be triggered by S3, You can find it on the console by navigating per screenshot below and clicking on view policy document.

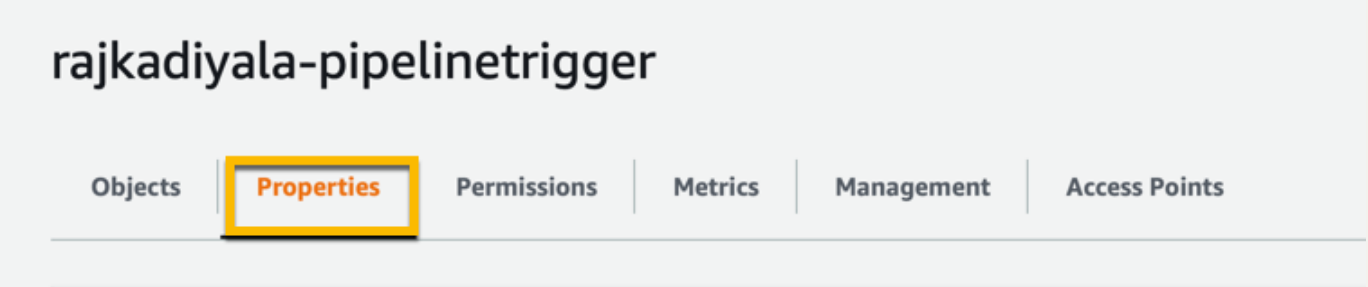


Resource-based policy document

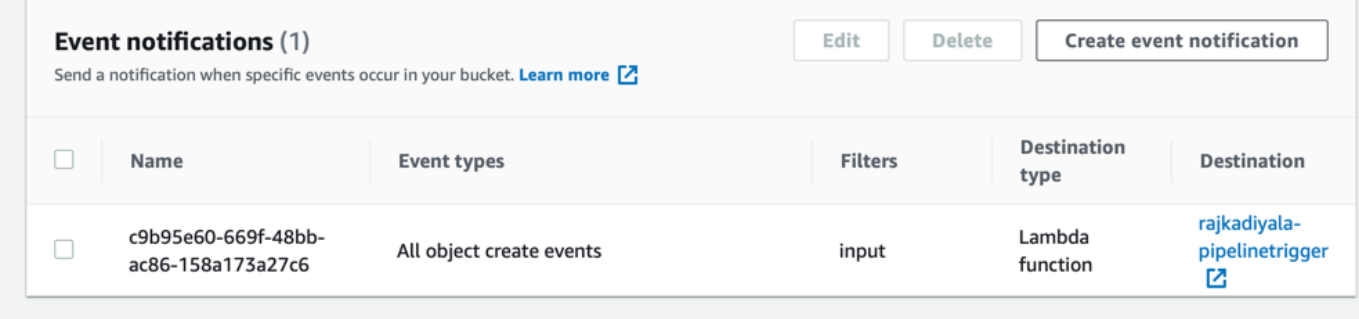
```
1  {
2    "Version": "2012-10-17",
3    "Id": "default",
4    "Statement": [
5      {
6        "Sid": "lambda-68c17345-ca91-42d9-84b5-a815835ae2cc",
7        "Effect": "Allow",
8        "Principal": {
9          "Service": "s3.amazonaws.com"
10       },
11       "Action": "lambda:InvokeFunction",
12       "Resource": "arn:aws:lambda:us-east-1:013696056273:function:rajkadiyala-pipelinet
13       "Condition": {
14         "StringEquals": {
15           "AWS:SourceAccount": "013696056273"
16         },
17         "ArnLike": {
18           "AWS:SourceArn": "arn:aws:s3:::rajkadiyala-pipelinetrigger"
19         }
20       }
21     }
22   ]
23 }
```

as you can see that an invoke function resource policy is added with our chosen s3 buckt as the source.

Now what has this done to the s3 bucket? Take a look at S3 console.



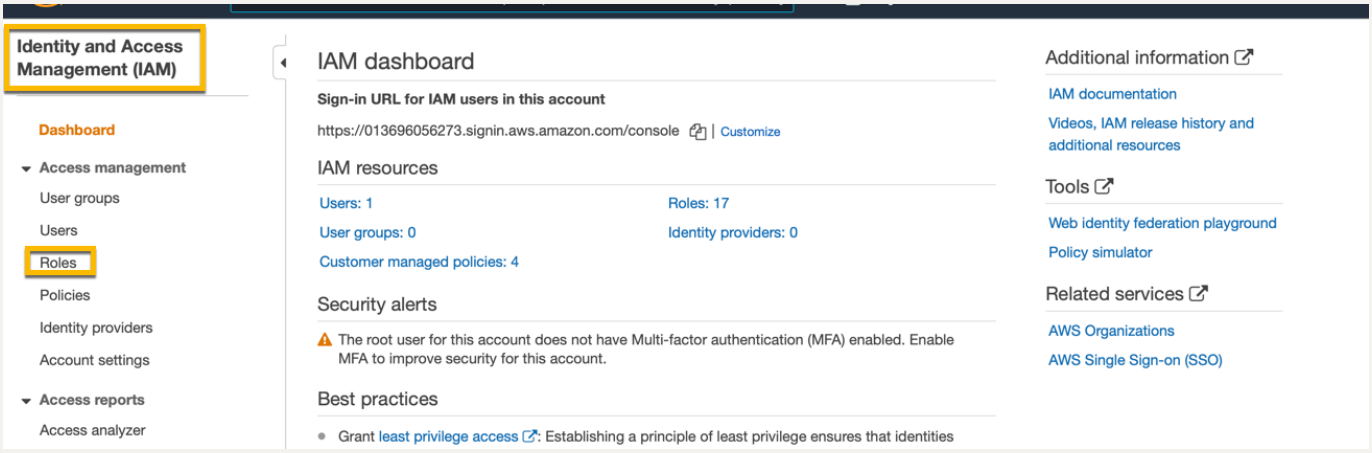
scroll down to event notifications

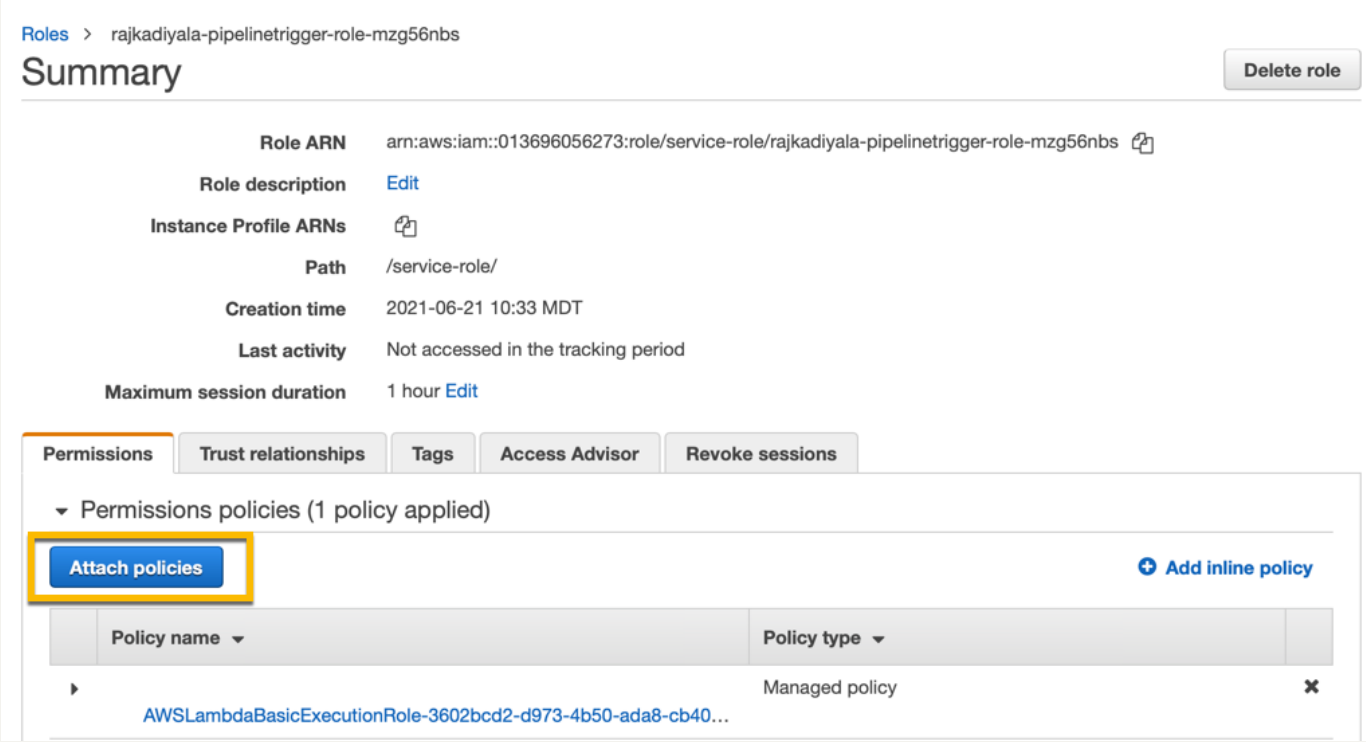


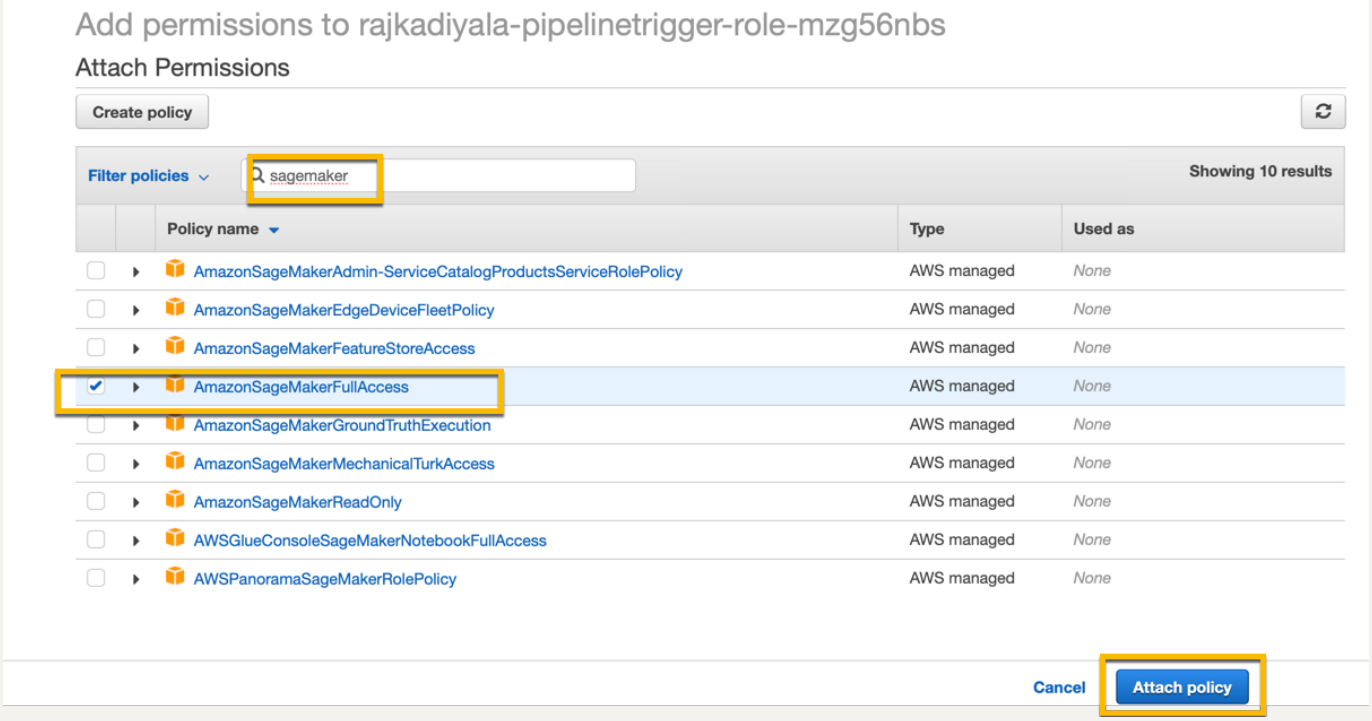
Lambda is added to event notification for that bucket.

IAM Roles

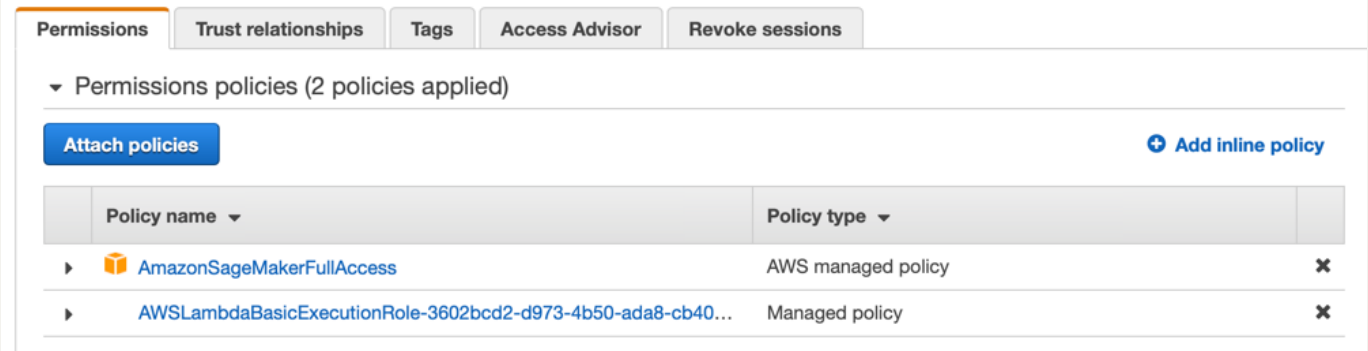
Provision Lambda permissions to trigger Sagemaker pipeline, For the sake of simplicity, we will provision lambda too open of permissions by add Sagemaker Full Access. In Practice, you want to narrow these permissions down to minimum needed.



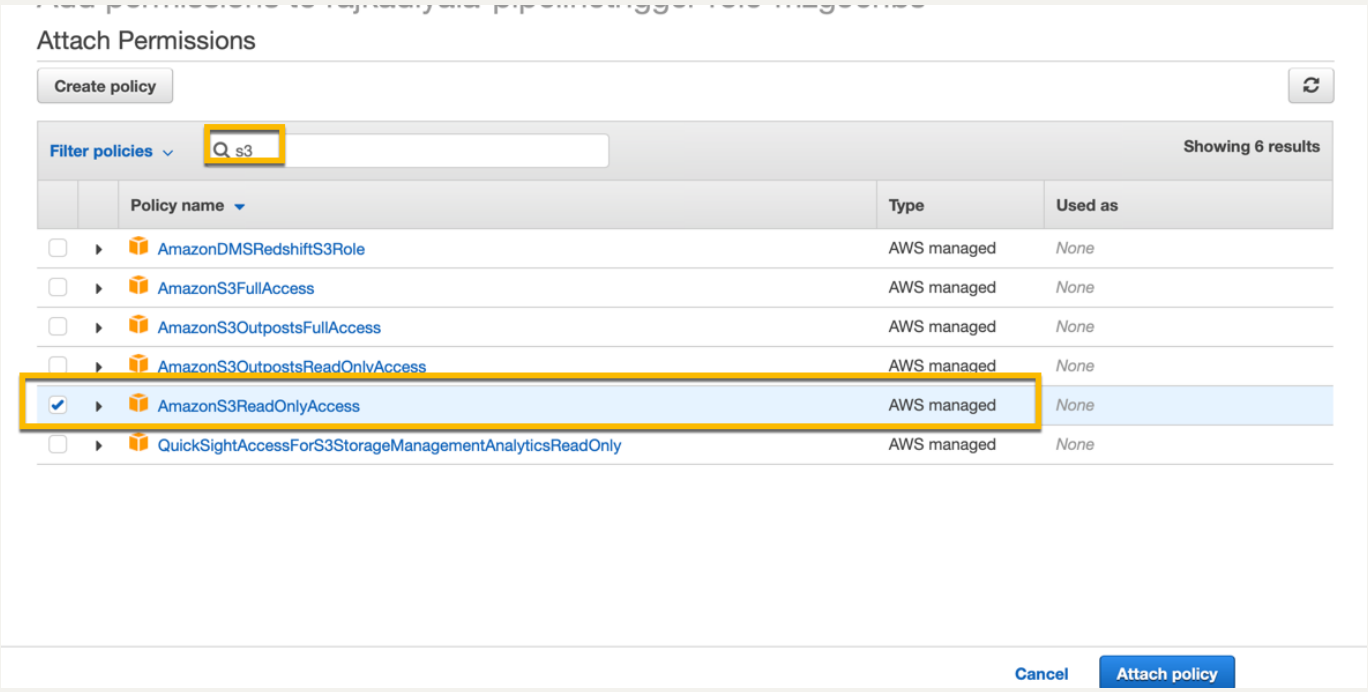




once added you can see that role now has the sagemaker access



Do the same giving Lambda S3 read access.



This is needed to get information about the file when Lambda is triggered by an upload.

Execution and Testing (Follow Along)

Our pipeline is up and available. Lambda is set up to trigger the pipeline and testing is as simple as uploading a file to S3 and seeing the pipeline trigger.

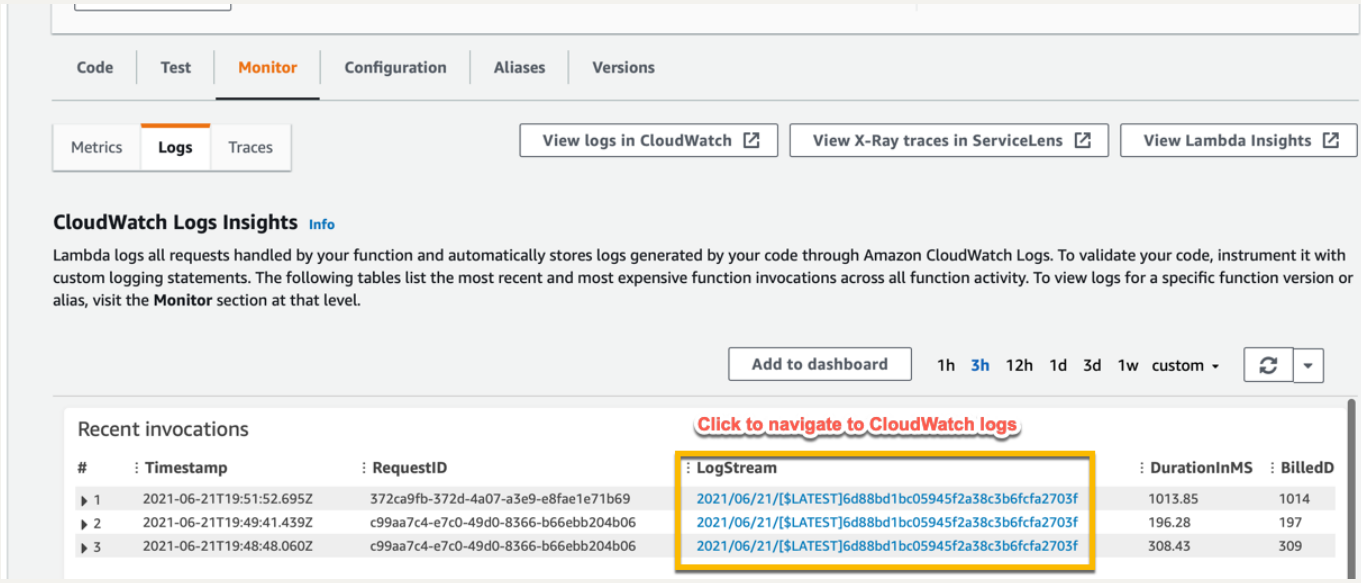
Download the abalone-dataset.csv and abalone-batch-dataset to your local from Sagemaker Studio.

Navigate to s3 bucket and

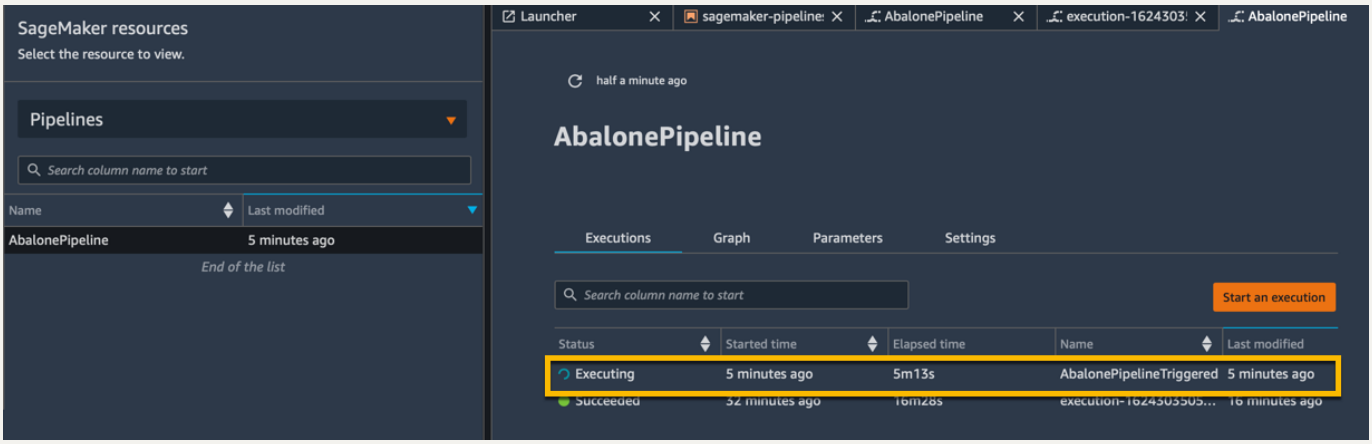
- to the batch folder, upload the batch dataset
- next to the input folder and click on upload. Select the abalone-dataset.csv in your local folder to input folder in s3 bucket.

Follow along on how to check for errors using cloud watch and how to validate that your set has worked.

You can check for the logs on lambda console



You can see if the pipeline is triggered in Sagemaker Studio Pipelines section.



Note: ***This set up triggers the pipeline on data uploads, it doesn't ensure that data is in right format or that pipeline runs successfully. That is on user put in conditions or logic to ensure you have successful runs.***