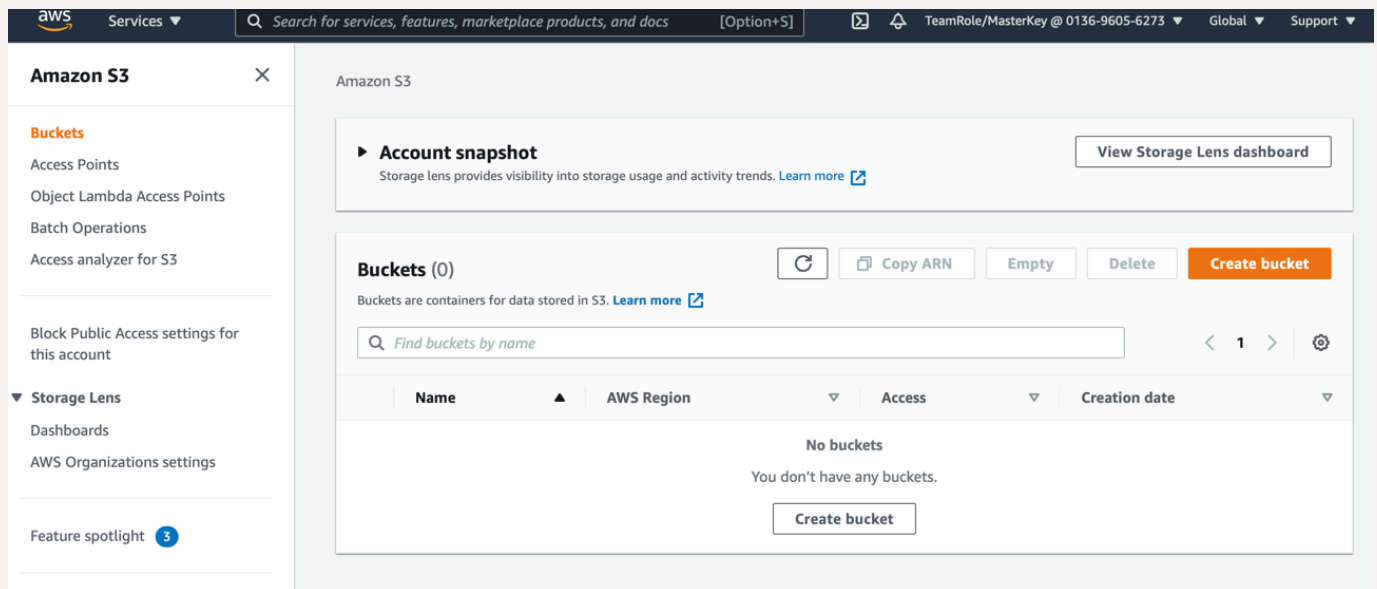


Automating pipelines with new data uploads

This session 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 **firstname-lastname**-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.



Create bucket

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

rajkadiyala-pipelinetrigger

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

US East (N. Virginia) us-east-1

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

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](#)

☒ Disable

☐ Enable

► Advanced settings

i After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel

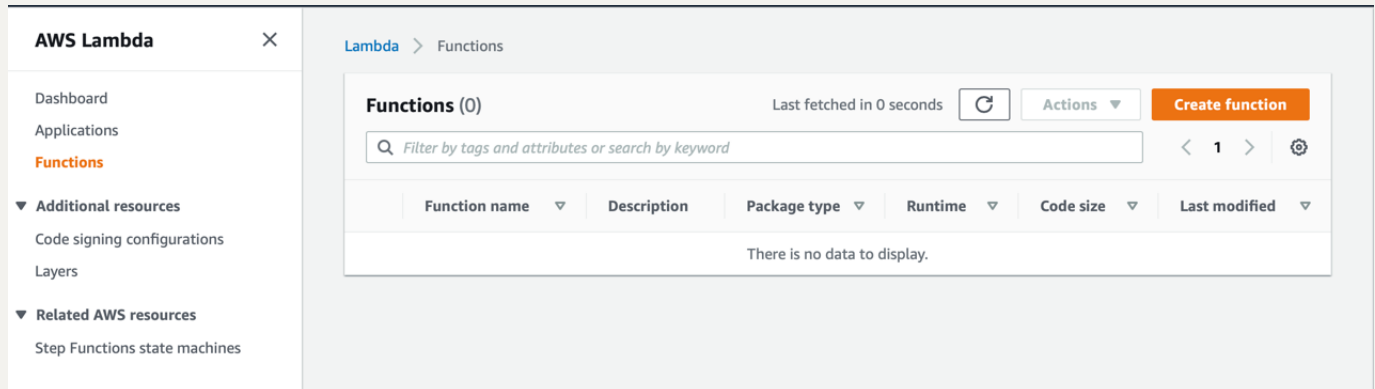
Create bucket

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.

Create function [Info](#)

Choose one of the following options to create your function.

Author from scratch ☒
Start with a simple Hello World example.

Use a blueprint ☐
Build a Lambda application from sample code and configuration presets for common use cases.

Container image ☐
Select a container image to deploy for your function.

Browse serverless app repository ☐
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☒ Create a new role with basic Lambda permissions

☐ Use an existing role

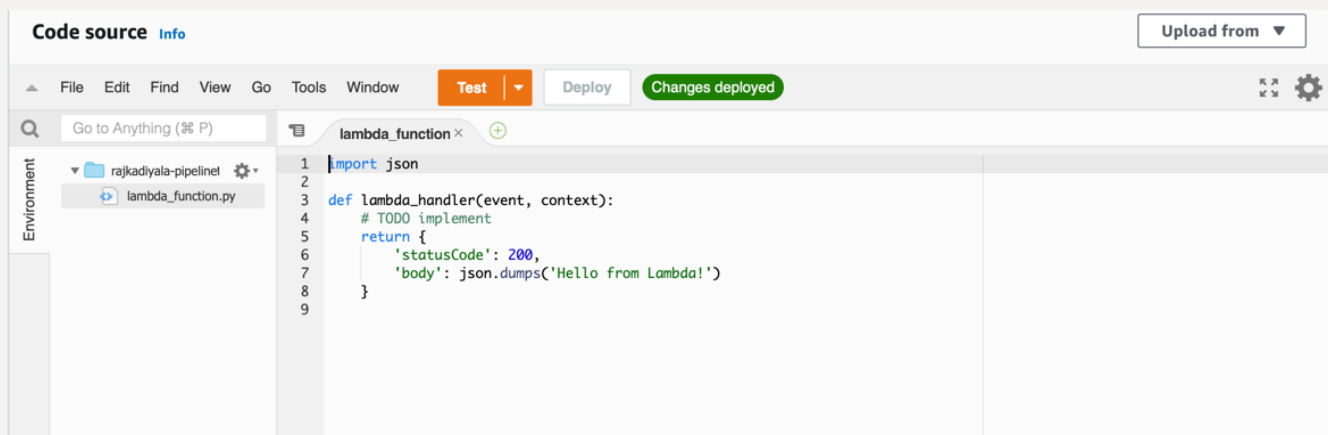
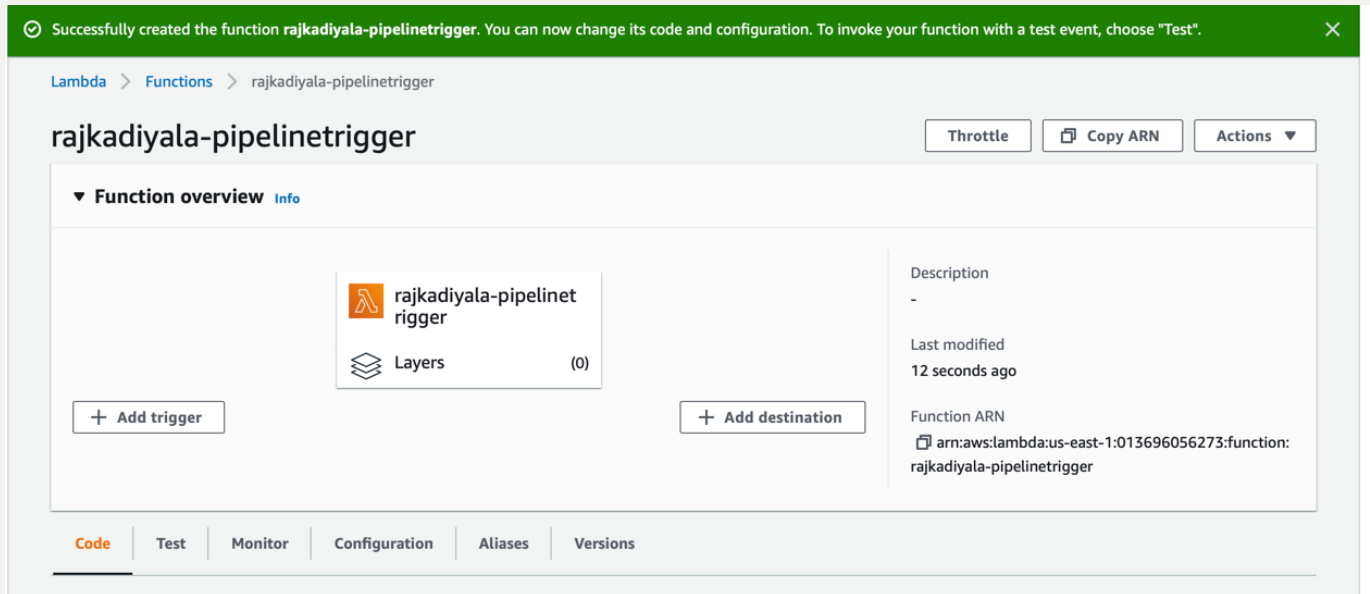
☐ Create a new role from AWS policy templates

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

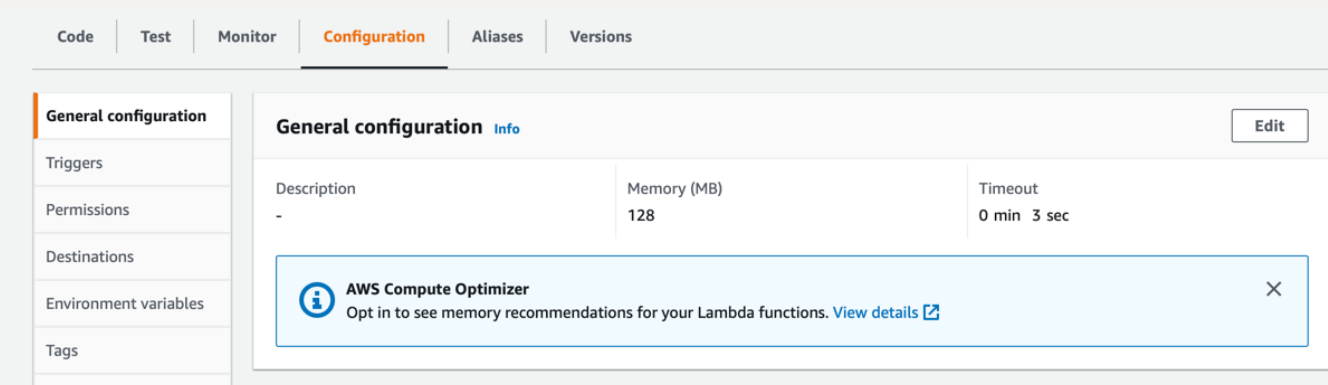
Lambda will create an execution role named `rajkadiyala-pipelinetrigger-role-mzg56nbs`, with permission to upload logs to Amazon CloudWatch Logs.

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



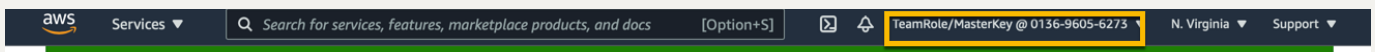
Edit the configuration and change the time out from 3 sec to 5min



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

The screenshot shows the AWS SageMaker Studio interface. At the top, there are tabs: Code, Test, Monitor, Configuration (selected), Aliases, and Versions. On the left, a sidebar lists various configuration categories: General configuration, Triggers, Permissions, Destinations, Environment variables, Tags, VPC, Monitoring and operations tools, Concurrency, Asynchronous invocation (highlighted with a yellow box), Code signing, and Database proxies. The main area displays the 'Asynchronous invocation' configuration. It includes a table with the following settings: 'Maximum age of event' set to '6 h 0 min 0 sec', 'Retry attempts' set to '2' (highlighted with a yellow box and a red callout bubble saying 'change to 1'), and 'Dead-letter queue service' set to 'None'. An 'Edit' button is located in the top right corner of the configuration table.

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-
xxxxxxxxxx', '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

```
1 import json
2 import boto3
3 import urllib.parse
4 import uuid
5
6 sm = boto3.client("sagemaker")
7
8 s3 = boto3.resource('s3')
9
10
11 def lambda_handler(event, context):
12     # using s3 object that triggered the pipeline
13     # read the object, copy the file to default bucket
14     # and update the inputData param
15
16     bucket = event['Records'][0]['s3']['bucket']['name']
17     key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')
18
19     print("bucket:" + str(bucket))
20     print("key:" + str(key))
21
22     print("copying input")
23     copy_source = {
24         'Bucket': bucket,
25         'Key': key
26     }
27     s3.meta.client.copy(copy_source, 'sagemaker-us-east-1-013696056273', 'input/abalone-dataset.csv')
28
29     print("copying batch")
30     copy_source = {
31         'Bucket': bucket,
32         'Key': 'batch/abalone-dataset-batch'
33     }
34     s3.meta.client.copy(copy_source, 'sagemaker-us-east-1-013696056273', 'batch/abalone-dataset-batch')
35
36     response = sm.start_pipeline_execution(
37         PipelineName='AbalonePipeline',
38         PipelineExecutionDisplayName='AbalonePipelineTriggered',
39         PipelineParameters=[
40             {
41                 'Name': 'InputData',
42                 'Value': 's3://sagemaker-us-east-1-013696056273/input/abalone-dataset.csv'
43             },
44             {
45                 'Name': 'BatchData',
46                 'Value': 's3://sagemaker-us-east-1-013696056273/batch/abalone-dataset-batch'
47             }
48         ],
49         PipelineExecutionDescription='triggered by s3',
50         ClientRequestToken=str(uuid.uuid4())
51     )
52
53     print(response)
54
55     return {
56         'statusCode': 200,
57         'body': json.dumps('Completed the Trigger!')
58     }
```

click file and save.

S3 Trigger for Lambda

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

rajkadiyala-pipelinetrigger

Throttle

Copy ARN

Actions ▼

▼ Function overview Info



rajkadiyala-pipelinet
rigger



Layers (0)

+ Add trigger

+ Add destination

Description

-

Last modified

36 minutes ago

Function ARN

arn:aws:lambda:us-east-1:013696056273:function:
rajkadiyala-pipelinetrigger

Type in S3 and click on S3 in drop down

Lambda > Add trigger

Add trigger

Trigger configuration

Select a trigger


Q s3|



S3
aws storage

Add trigger

Trigger configuration

 **S3**
aws storage

Bucket

Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

rajkadiyala-pipelinetrigger

Event type

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.

All object create events

Prefix - optional

Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

input

Suffix - optional

Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

e.g. .jpg

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.

Lambda > Functions > rajkadiyala-pipelinetrigger

rajkadiyala-pipelinetrigger


Throttle


Copy ARN

Actions

✓ The trigger rajkadiyala-pipelinetrigger was successfully added to function rajkadiyala-pipelinetrigger. The function is now receiving events from the trigger.

Function overview

 **rajkadiyala-pipelinetrigger**

 Layers (0)

 S3

+ Add trigger

+ Add destination

Description

-

Last modified

41 minutes ago

Function ARN

arn:aws:lambda:us-east-1:013696056273:function:rajkadiyala-pipelinetrigger

Take a look 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.

Code

Test

Monitor

Configuration

Aliases

Versions

General configuration

Triggers

Permissions

Destinations

Environment variables

Tags

VPC

Monitoring and operations tools

Concurrency

Asynchronous invocation

Code signing

Database proxies

File systems

State machines

Execution role

Edit

Role name

rajkadiyala-pipelinetrigger-role-mzg56nbs

Resource summary

View role document

Amazon CloudWatch Logs

3 actions, 2 resources

To view the resources and actions that your function has permission to access, choose a service.

By action

By resource

Resource	Actions
arn:aws:logs:us-east-1:013696056273:*	Allow: logs:CreateLogGroup
arn:aws:logs:us-east-1:013696056273:log-group:/aws/lambda/rajkadiyala-pipelinetrigger:*	Allow: logs:CreateLogStream Allow: logs:PutLogEvents

Lambda obtained this information from the following policy statements:

- Managed policy AWSLambdaBasicExecutionRole-3602bcd2-d973-4b50-ada8-cb40ab5b7061, statement 0
- Managed policy AWSLambdaBasicExecutionRole-3602bcd2-d973-4b50-ada8-cb40ab5b7061, statement 1

Resource-based policy

Info

View policy document

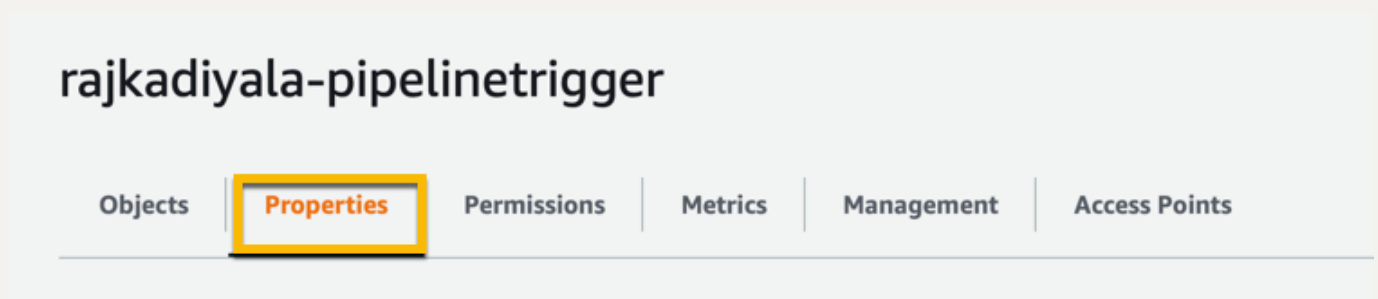
A resource-based policy lets you grant permissions to other AWS accounts or services on a per-resource basis.

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-pipeline",
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 bucket as the source.

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



scroll down to event notifications

Event notifications (1)					
Send a notification when specific events occur in your bucket. Learn more					
<input type="checkbox"/>	Name	Event types	Filters	Destination type	Destination
<input type="checkbox"/>	c9b95e60-669f-48bb-ac86-158a173a27c6	All object create events	input	Lambda function	rajkadiyala-pipelinetrigger

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.

Identity and Access Management (IAM)

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Access analyzer

IAM dashboard

Sign-in URL for IAM users in this account

https://013696056273.signin.aws.amazon.com/console | [Customize](#)

IAM resources

Users: 1

Roles: 17

User groups: 0

Identity providers: 0

Customer managed policies: 4

Security alerts

⚠ The root user for this account does not have Multi-factor authentication (MFA) enabled. Enable MFA to improve security for this account.

Best practices

• Grant [least privilege access](#): Establishing a principle of least privilege ensures that identities

Additional information

[IAM documentation](#)

[Videos, IAM release history and additional resources](#)

Tools

[Web identity federation playground](#)

[Policy simulator](#)

Related services

[AWS Organizations](#)

[AWS Single Sign-on \(SSO\)](#)

Roles > rajkadiyala-pipelinetrigger-role-mzg56nbs

Summary

[Delete role](#)

Role ARN `arn:aws:iam::013696056273:role/service-role/rajkadiyala-pipelinetrigger-role-mzg56nbs` [Copy](#)

Role description [Edit](#)

Instance Profile ARNs [Copy](#)

Path `/service-role/`

Creation time 2021-06-21 10:33 MDT

Last activity Not accessed in the tracking period

Maximum session duration 1 hour [Edit](#)

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

▼ Permissions policies (1 policy applied)

[Attach policies](#) [Add inline policy](#)

Policy name ▼	Policy type ▼
▶ AWSLambdaBasicExecutionRole-3602bcd2-d973-4b50-ada8-cb40...	Managed policy X

Add permissions to rajkadiyala-pipelinetrigger-role-mzg56nbs

Attach Permissions

[Create policy](#) [Refresh](#)

Filter policies ▼ Showing 10 results

	Policy name ▼	Type	Used as
<input type="checkbox"/>	▶ AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy	AWS managed	None
<input type="checkbox"/>	▶ AmazonSageMakerEdgeDeviceFleetPolicy	AWS managed	None
<input type="checkbox"/>	▶ AmazonSageMakerFeatureStoreAccess	AWS managed	None
<input checked="" type="checkbox"/>	▶ AmazonSageMakerFullAccess	AWS managed	None
<input type="checkbox"/>	▶ AmazonSageMakerGroundTruthExecution	AWS managed	None
<input type="checkbox"/>	▶ AmazonSageMakerMechanicalTurkAccess	AWS managed	None
<input type="checkbox"/>	▶ AmazonSageMakerReadOnly	AWS managed	None
<input type="checkbox"/>	▶ AWSGlueConsoleSageMakerNotebookFullAccess	AWS managed	None
<input type="checkbox"/>	▶ AWSPanoramaSageMakerRolePolicy	AWS managed	None

[Cancel](#) [Attach policy](#)

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

Permissions Trust relationships Tags Access Advisor Revoke sessions

▼ Permissions policies (2 policies applied)

Attach policies Add inline policy

Policy name ▼	Policy type ▼	
▶ AmazonSageMakerFullAccess	AWS managed policy	✕
▶ AWSLambdaBasicExecutionRole-3602bcd2-d973-4b50-ada8-cb40...	Managed policy	✕

Do the same giving Lambda S3 read access.

Attach Permissions

Create policy

Filter policies ▼ Q s3 Showing 6 results

Policy name ▼	Type	Used as
▶ AmazonDMSRedshiftS3Role	AWS managed	None
▶ AmazonS3FullAccess	AWS managed	None
▶ AmazonS3OutpostsFullAccess	AWS managed	None
▶ AmazonS3OutpostsReadOnlyAccess	AWS managed	None
<input checked="" type="checkbox"/> ▶ AmazonS3ReadOnlyAccess	AWS managed	None
▶ QuickSightAccessForS3StorageManagementAnalyticsReadOnly	AWS managed	None

Cancel Attach policy

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

CloudWatch Logs Insights [Info](#)

Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables list the most recent and most expensive function invocations across all function activity. To view logs for a specific function version or alias, visit the **Monitor** section at that level.

[View logs in CloudWatch](#) [View X-Ray traces in ServiceLens](#) [View Lambda Insights](#)

[Add to dashboard](#) 1h **3h** 12h 1d 3d 1w custom [Refresh](#) [Filter](#)

Recent invocations

#	Timestamp	RequestID	LogStream	DurationInMS	BilledD
▶ 1	2021-06-21T19:51:52.695Z	372ca9fb-372d-4a07-a3e9-e8fae1e71b69	2021/06/21/[\$LATEST]6d88bd1bc05945f2a38c3b6fca2703f	1013.85	1014
▶ 2	2021-06-21T19:49:41.439Z	c99aa7c4-e7c0-49d0-8366-b66ebb204b06	2021/06/21/[\$LATEST]6d88bd1bc05945f2a38c3b6fca2703f	196.28	197
▶ 3	2021-06-21T19:48:48.060Z	c99aa7c4-e7c0-49d0-8366-b66ebb204b06	2021/06/21/[\$LATEST]6d88bd1bc05945f2a38c3b6fca2703f	308.43	309

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

SageMaker resources
Select the resource to view.

Pipelines

Search column name to start

Name	Last modified
AbalonePipeline	5 minutes ago

End of the list

AbalonePipeline

half a minute ago

Executions Graph Parameters Settings

Search column name to start [Start an execution](#)

Status	Started time	Elapsed time	Name	Last modified
Executing	5 minutes ago	5m13s	AbalonePipelineTriggered	5 minutes ago
Succeeded	32 minutes ago	16m12s	execution-1624305305...	16 minutes ago

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.