Case Study

Serverless Image Processing Workflow

Name: Yash Santosh Rahate
Class: D15B
Roll no.: 48

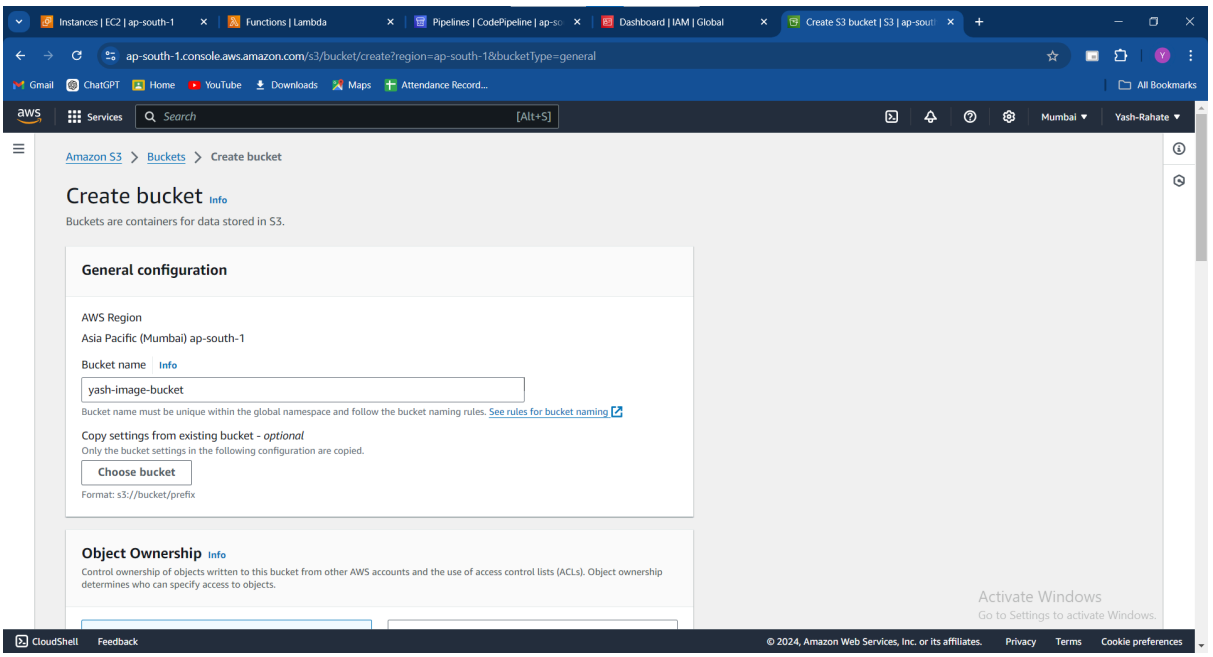
Problem Statement:

- **Concepts Used**: AWS Lambda, S3, and CodePipeline.
- **Problem Statement**: "Create a serverless workflow that triggers an AWS Lambda function when a new image is uploaded to an S3 bucket. Use CodePipeline to automate the deployment of the Lambda function."
- **Tasks**:
    - Create a Lambda function in Python that logs and processes an image when uploaded to a specific S3 bucket.
    - Set up AWS CodePipeline to automatically deploy updates to the Lambda function.
    - Upload a sample image to S3 and verify that the Lambda function is triggered and logs the event.

SOLUTION

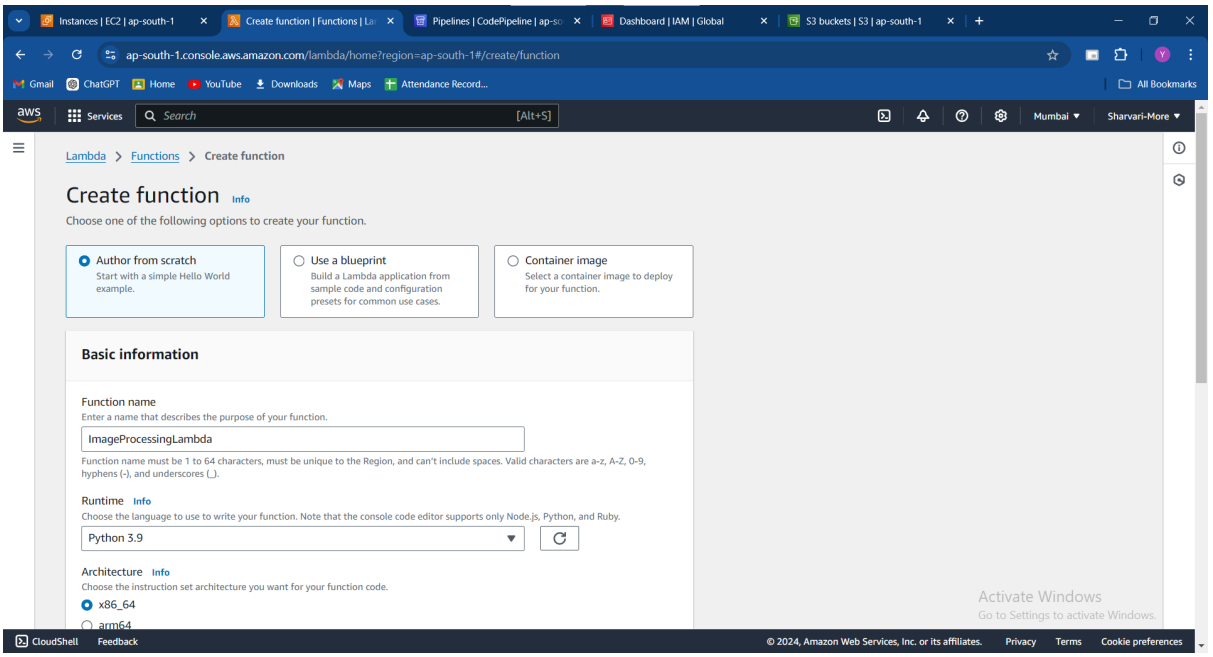
Step 1: Set Up an S3 Bucket

1. **Log in to AWS Console** and go to the **S3** service.
2. Click **Create Bucket**, give it a unique name (e.g., image-processing-bucket), and choose a region.
3. Enable **versioning** if needed and leave other options as default. Click **Create Bucket**.



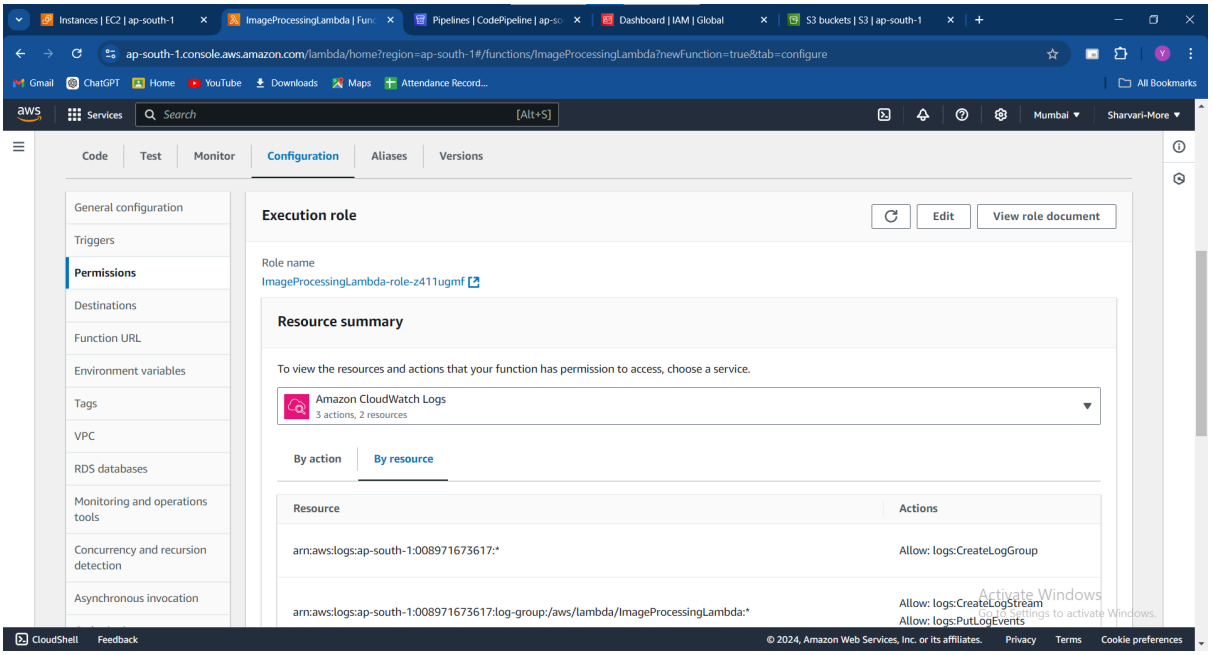
Step 2: Create a Lambda Function to Process Images

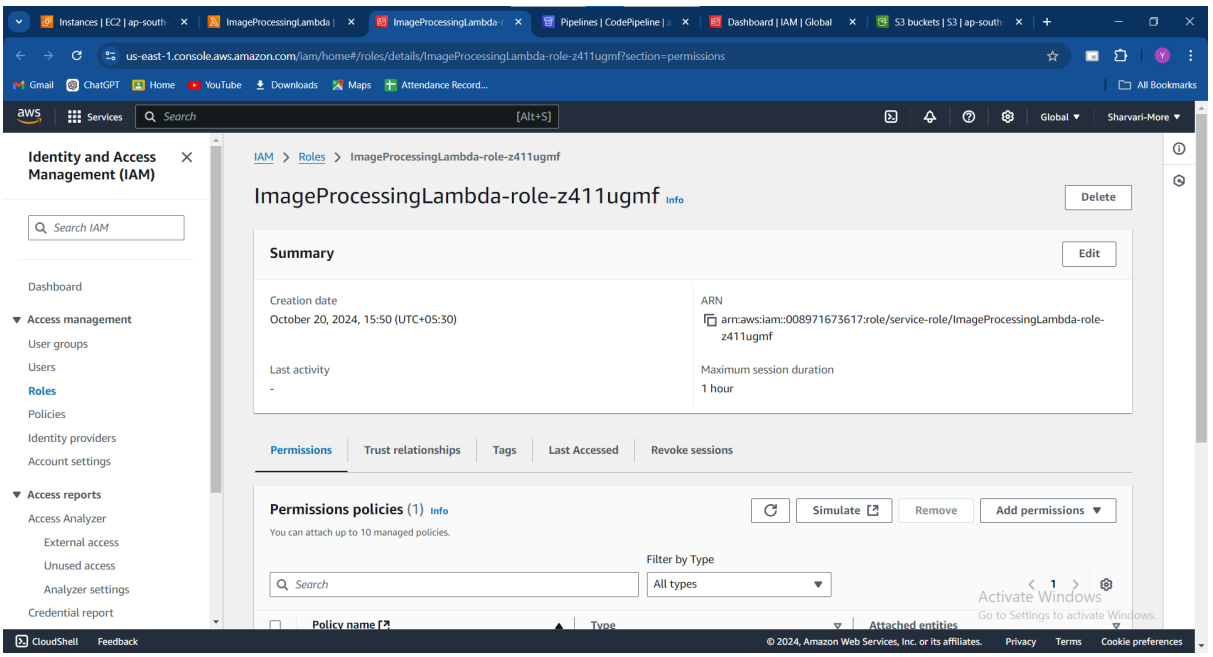
1. Go to the **Lambda** service in AWS.
2. Click **Create Function** and choose **Author from Scratch**.
    - **Name**: ImageProcessingLambda
    - **Runtime**: Python 3.x (e.g., Python 3.9)

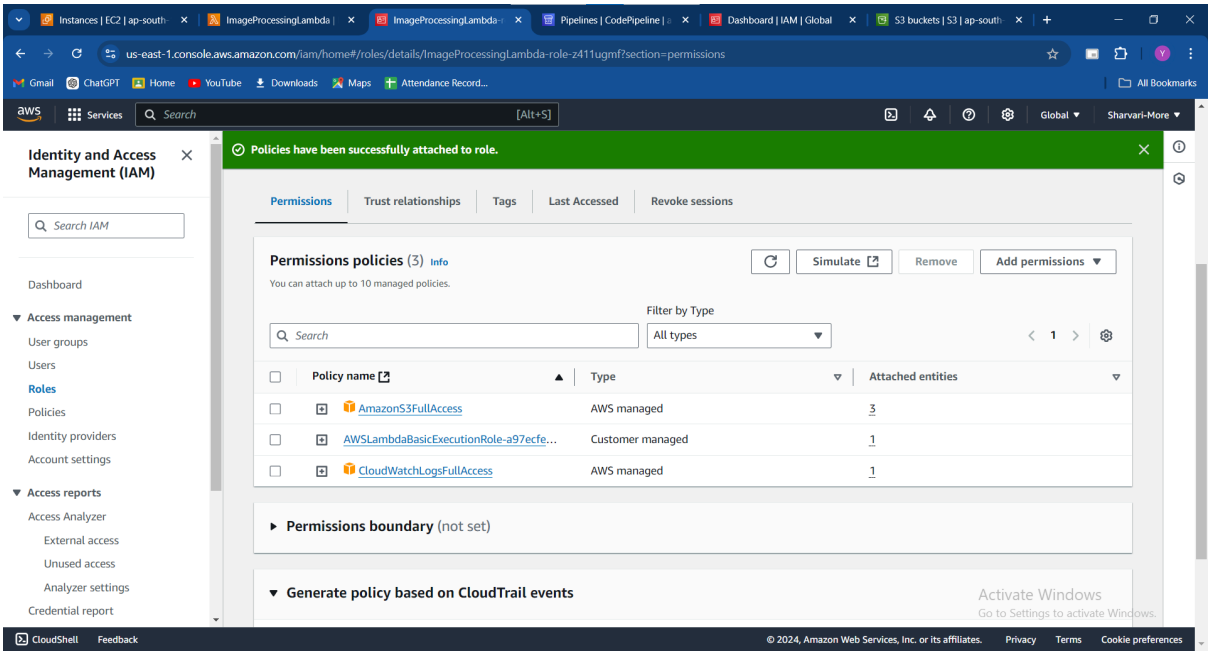


3. **IAM Role for Lambda**:
    - Create a new role with basic Lambda permissions:
        - Choose **Create a new role with basic Lambda permissions**.
        - It automatically assigns the policy **AWSLambdaBasicExecutionRole** to the role, which allows the function to write logs to **CloudWatch**.

4. After the function is created, add the following permissions to access the S3 bucket:
    - Click on **Configuration** > **Permissions** > **Execution Role**.
    - Click on the role and attach the following permissions:
        - **AmazonS3FullAccess**
        - **CloudWatchLogsFullAccess**





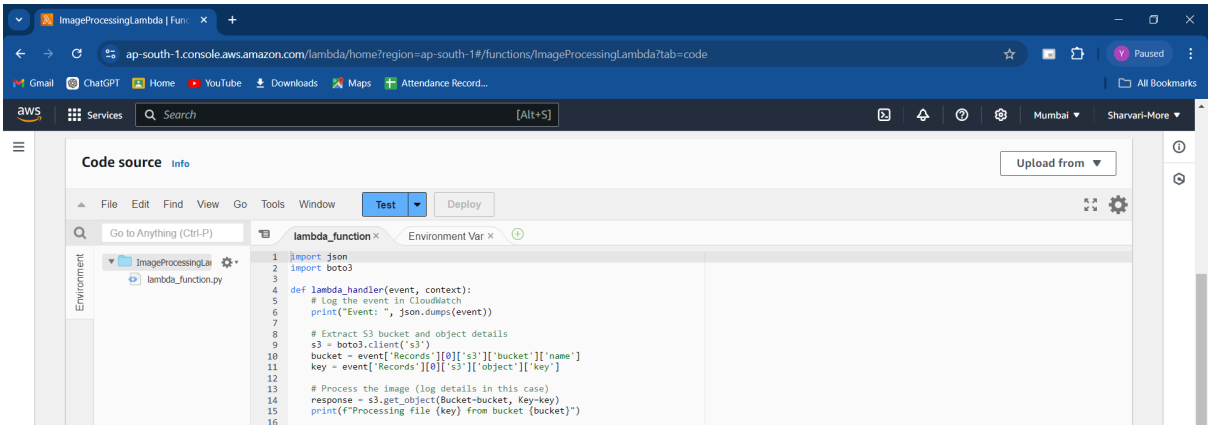


5. **Add Python Code to Process Images**:
   - Go back to **Code** section and replace the sample code with:

CODE:

```
import json
import boto3
def lambda_handler(event, context):
    # Log the event in CloudWatch
    print("Event: ", json.dumps(event))

    # Extract S3 bucket and object details
```

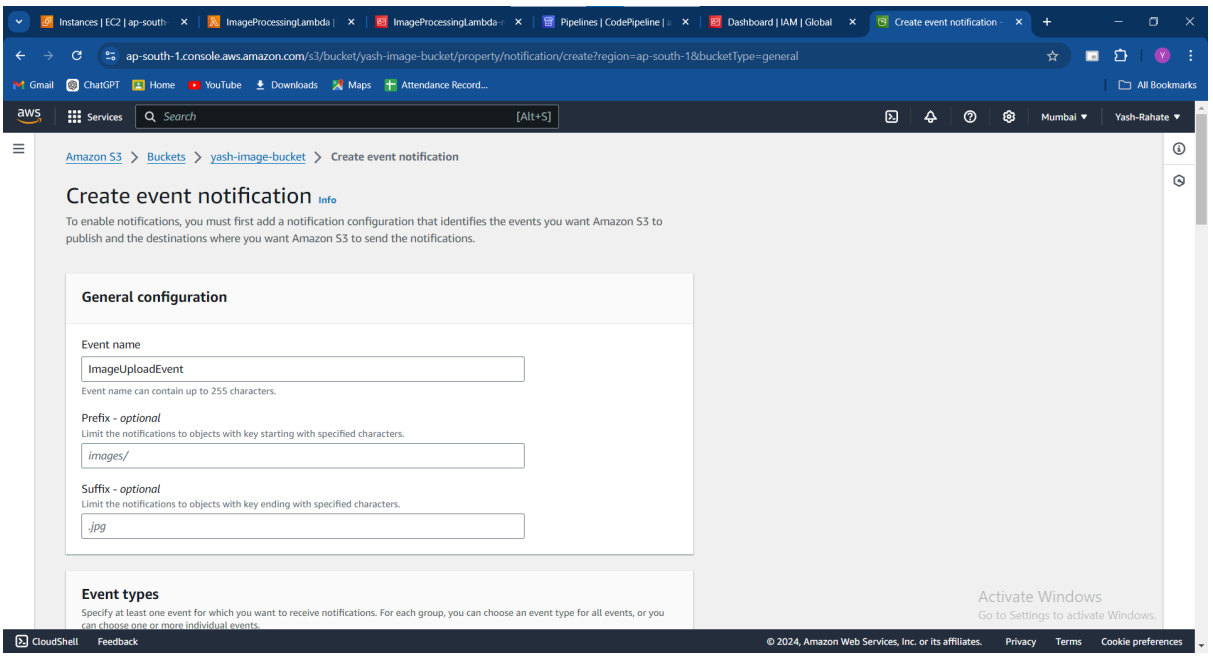
```
s3 = boto3.client('s3')
bucket = event['Records'][0]['s3']['bucket']['name']
key = event['Records'][0]['s3']['object']['key']

# Process the image (log details in this case)
response = s3.get_object(Bucket=bucket, Key=key)
print(f"Processing file {key} from bucket {bucket}")

return {
    'statusCode': 200,
    'body': json.dumps('Image processed successfully!')
}
```

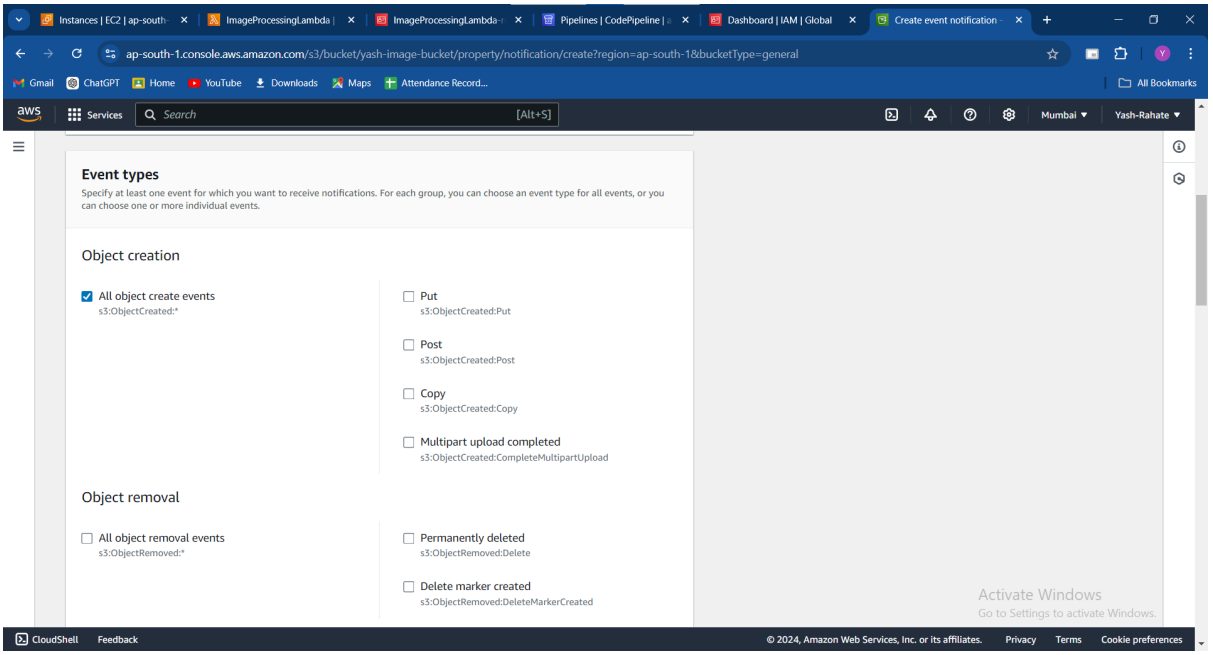


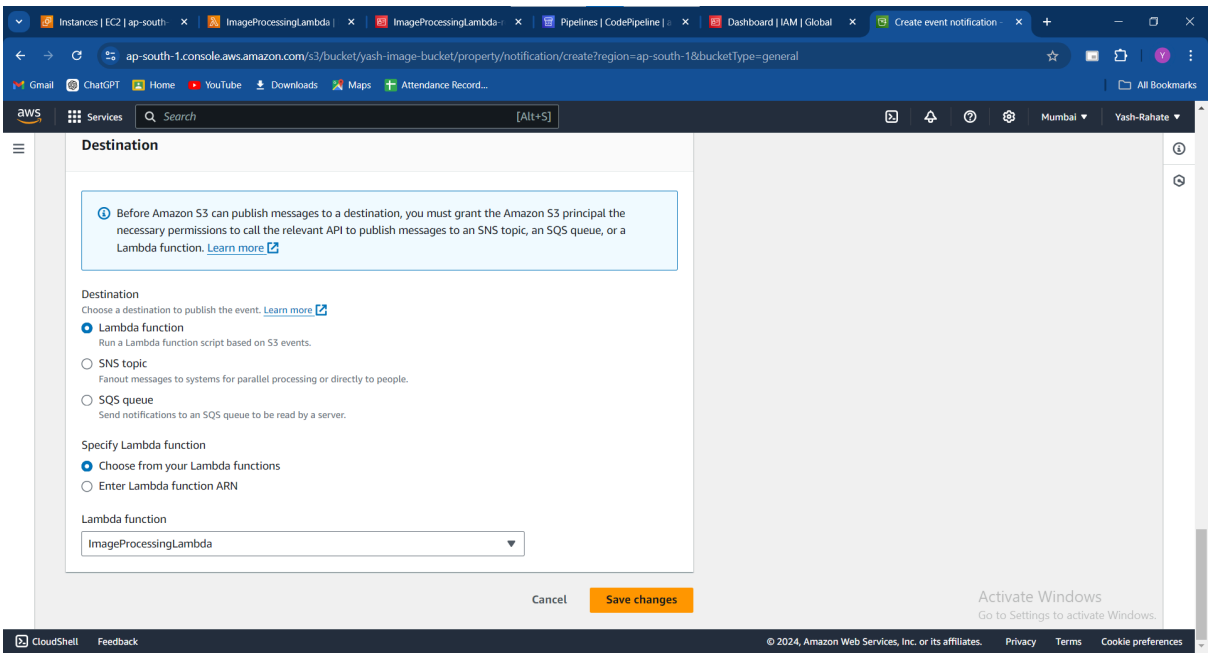
- This code logs the S3 event and retrieves basic information about the uploaded image.

Step 3: Set Up S3 Event Notification to Trigger Lambda

1. Go back to the **S3** service and select your bucket (image-processing-bucket).
2. In the **Properties** tab, scroll to the **Event Notifications** section and click **Create Event Notification**.
   - **Event Name**: ImageUploadEvent
   - **Event Type**: Select **All object create events** (i.e., triggers when any file is uploaded).
   - **Destination**: Choose **Lambda function** and select ImageProcessingLambda.


Instances | EC2 | ap-south-
ImageProcessingLambda |
ImageProcessingLambda-r
Pipelines | CodePipeline | a
Dashboard | IAM | Global
Create event notification -
ap-south-1.console.aws.amazon.com/s3/bucket/yash-image-bucket/property/notification/create?region=ap-south-1&bucketType=general
Gmail
ChatGPT
Home
YouTube
Downloads
Maps
Attendance Record...
All Bookmarks
aws
Services
Search
[Alt+S]
Mumbai
Yash-Rahate
Amazon S3 > Buckets > yash-image-bucket > Create event notification
Create event notification Info
To enable notifications, you must first add a notification configuration that identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications.
General configuration
Event name
ImageUploadEvent
Event name can contain up to 255 characters.
Prefix - optional
Limit the notifications to objects with key starting with specified characters.
images/
Suffix - optional
Limit the notifications to objects with key ending with specified characters.
.jpg
Event types
Specify at least one event for which you want to receive notifications. For each group, you can choose an event type for all events, or you can choose one or more individual events.
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Instances | EC2 | ap-south-
ImageProcessingLambda |
ImageProcessingLambda-r
Pipelines | CodePipeline | a
Dashboard | IAM | Global
Create event notification -
ap-south-1.console.aws.amazon.com/s3/bucket/yash-image-bucket/property/notification/create?region=ap-south-1&bucketType=general
Gmail
ChatGPT
Home
YouTube
Downloads
Maps
Attendance Record...
All Bookmarks
aws
Services
Search
[Alt+S]
Mumbai
Yash-Rahate
Event types
Specify at least one event for which you want to receive notifications. For each group, you can choose an event type for all events, or you can choose one or more individual events.
Object creation
All object create events
s3:ObjectCreated:*
Put
s3:ObjectCreated:Put
Post
s3:ObjectCreated:Post
Copy
s3:ObjectCreated:Copy
Multipart upload completed
s3:ObjectCreated:CompleteMultipartUpload
Object removal
All object removal events
s3:ObjectRemoved:*
Permanently deleted
s3:ObjectRemoved:Delete
Delete marker created
s3:ObjectRemoved:DeleteMarkerCreated
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



3. Click **Save Changes**.

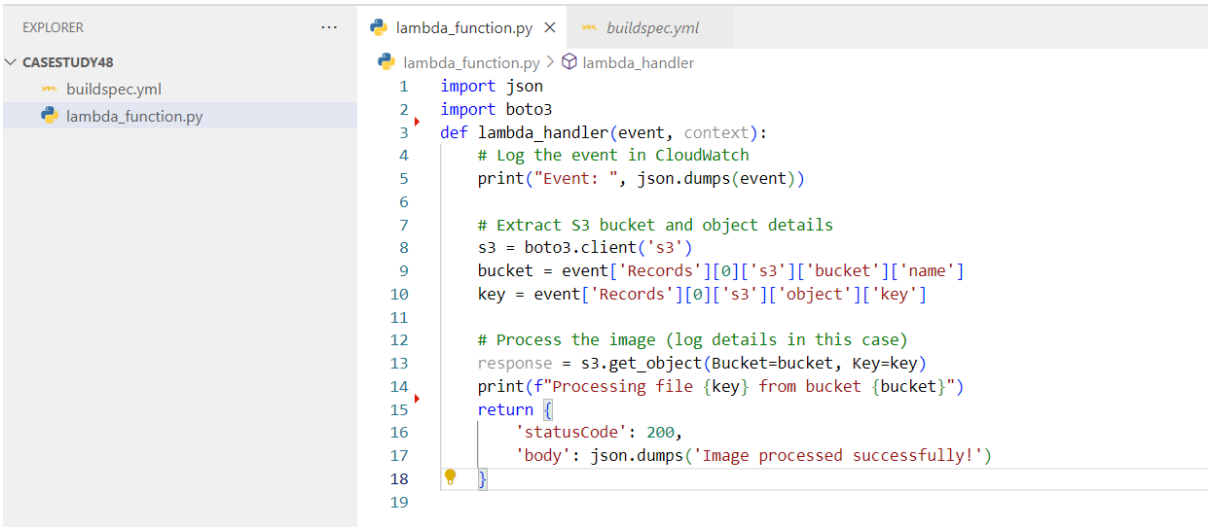
Step 4 :Step-by-Step Guide Using CodeBuild:

1. **Create a Buildspec File**: In your GitHub repo (where your lambda_function.py is), add a buildspec.yml file. This file will tell CodeBuild how to package and deploy your Lambda function.
   Example buildspec.yml:

   CODE:

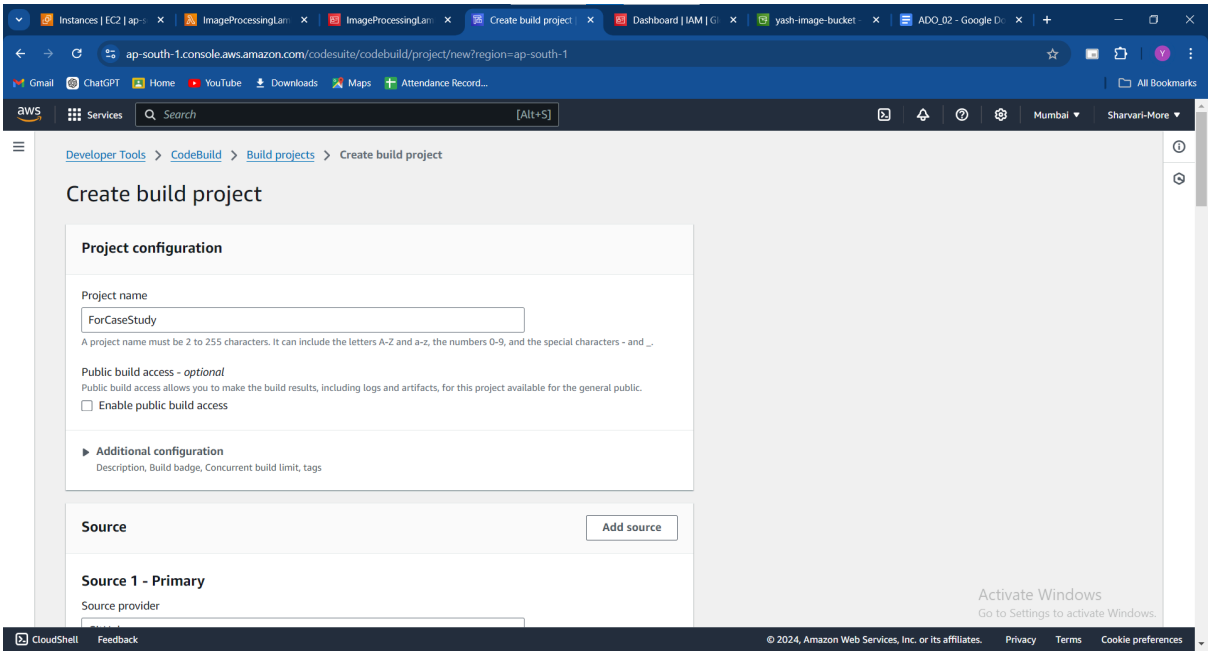
```
version: 0.2

phases:
  install:
    commands:
      - pip install --upgrade awscli
  build:
    commands:
      - zip function.zip lambda_function.py
```

- aws lambda update-function-code --function-name ImageProcessingLambda --zip-file fileb://function.zip

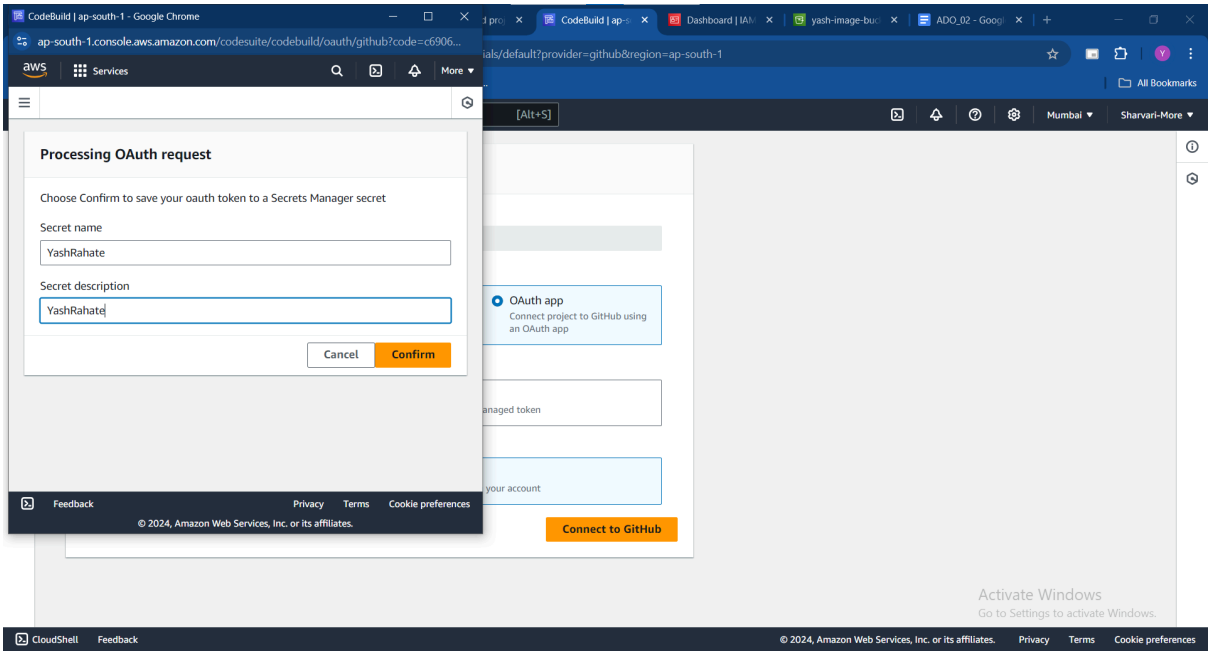


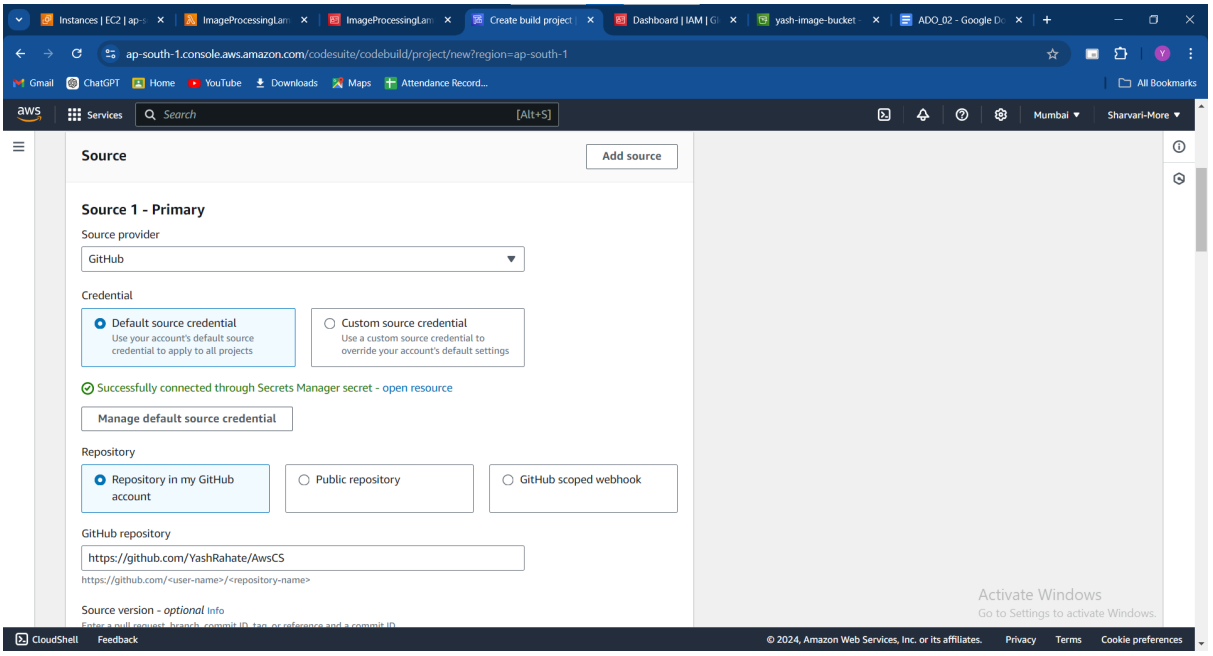
2. **Create a CodeBuild Project**:

- Go to **AWS CodeBuild** and create a new build project.

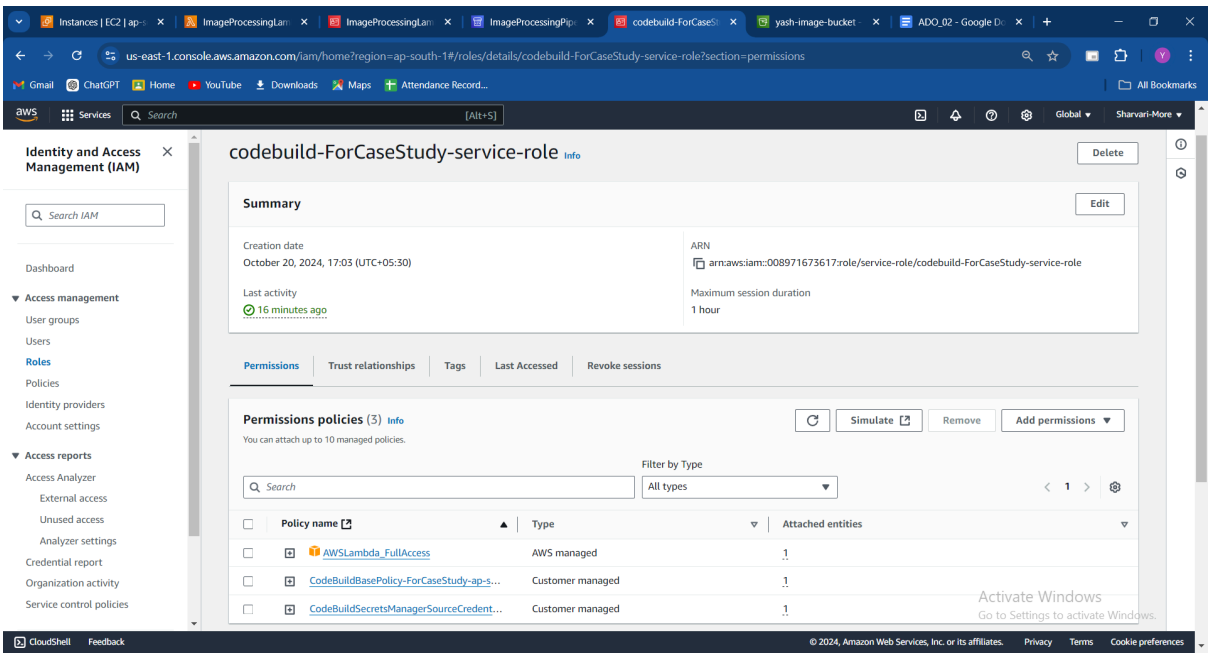


- For the **Source**, select the same GitHub repo you are using.

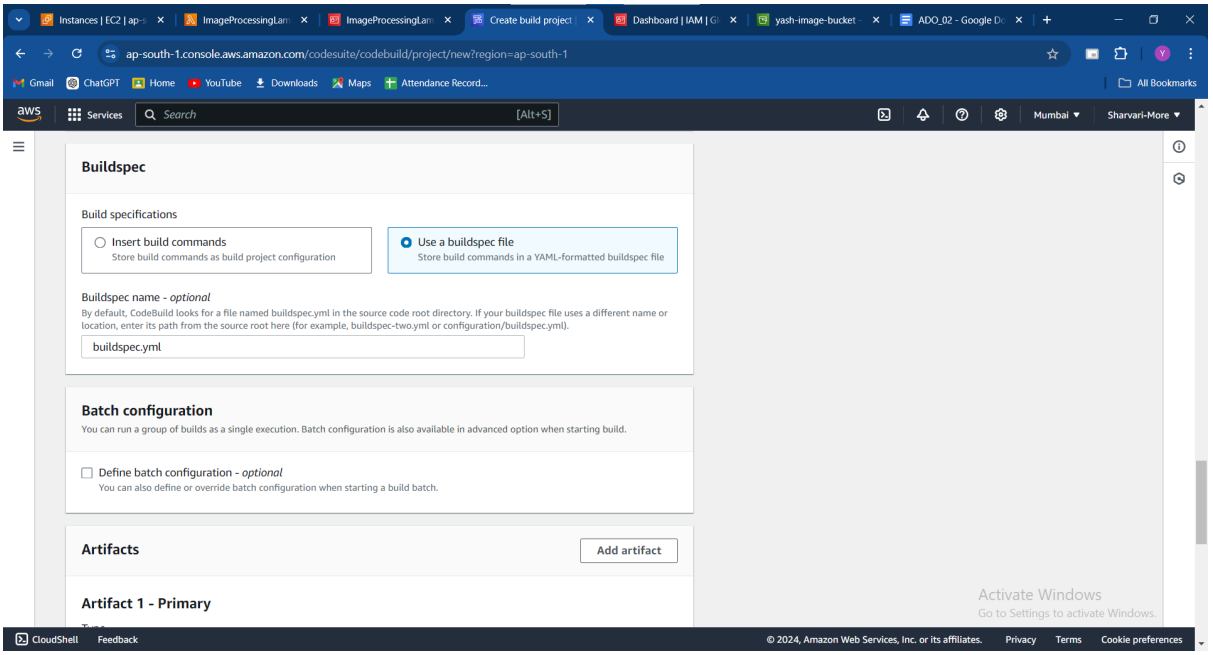




- For the **Environment**, select a managed image (e.g., Ubuntu with standard runtimes).
- Ensure that the environment has the correct permissions to update the Lambda function (using a role with AWSLambdaFullAccess or similar).

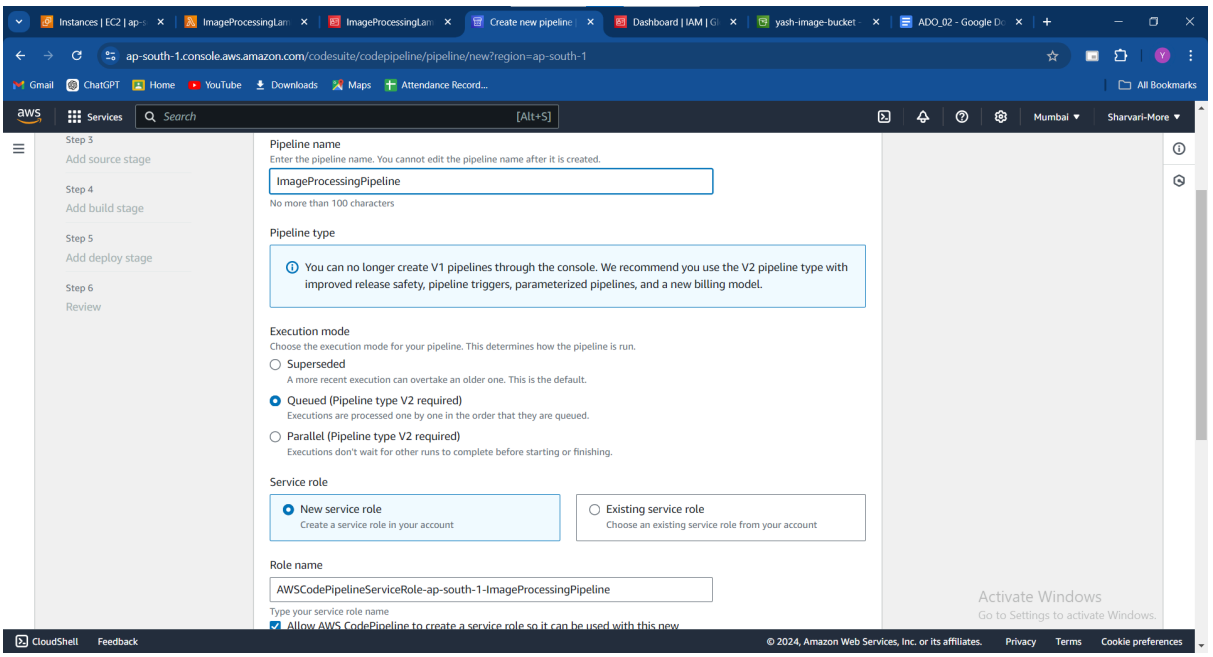


- Specify the **buildspec.yml** file from your GitHub repository.

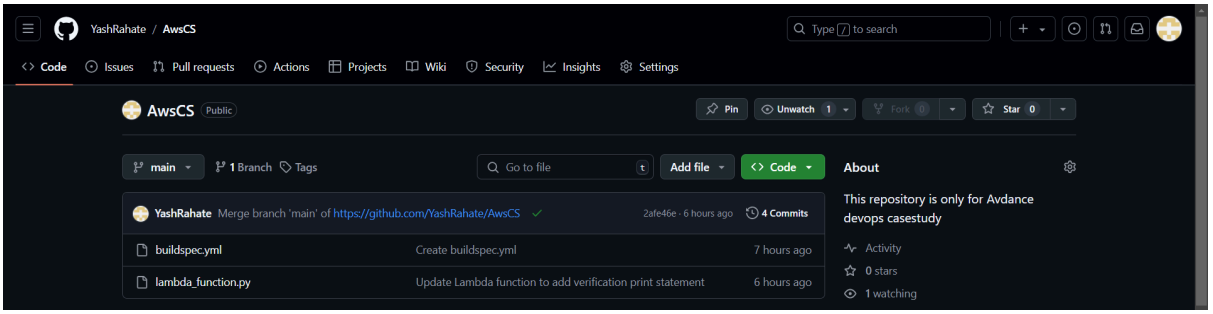


Step 5: Set Up AWS CodePipeline to Automate Lambda Deployment

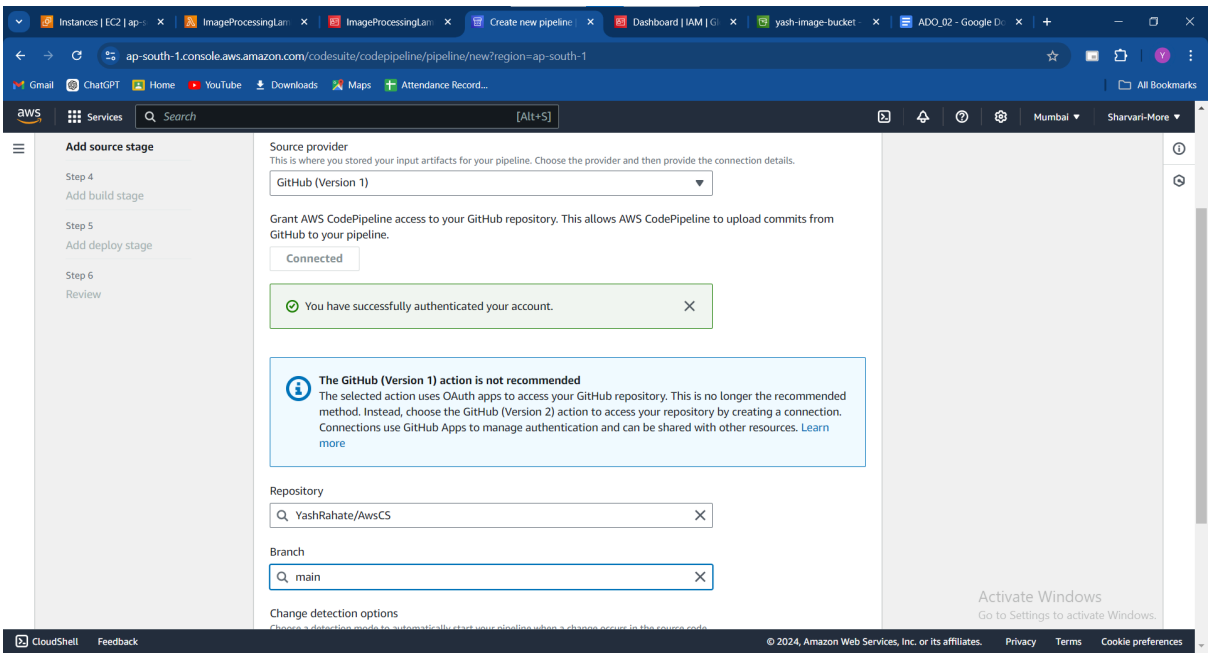
1. Go to the **CodePipeline** service and click **Create Pipeline**.
2. **Pipeline Settings**:
    - **Pipeline Name**: ImageProcessingPipeline
    - **Service Role**: Allow CodePipeline to create a new role.



3. **Source Stage** (Code Repository):
   - For **Source Provider**, choose **GitHub** or **AWS CodeCommit** based on your code repository.

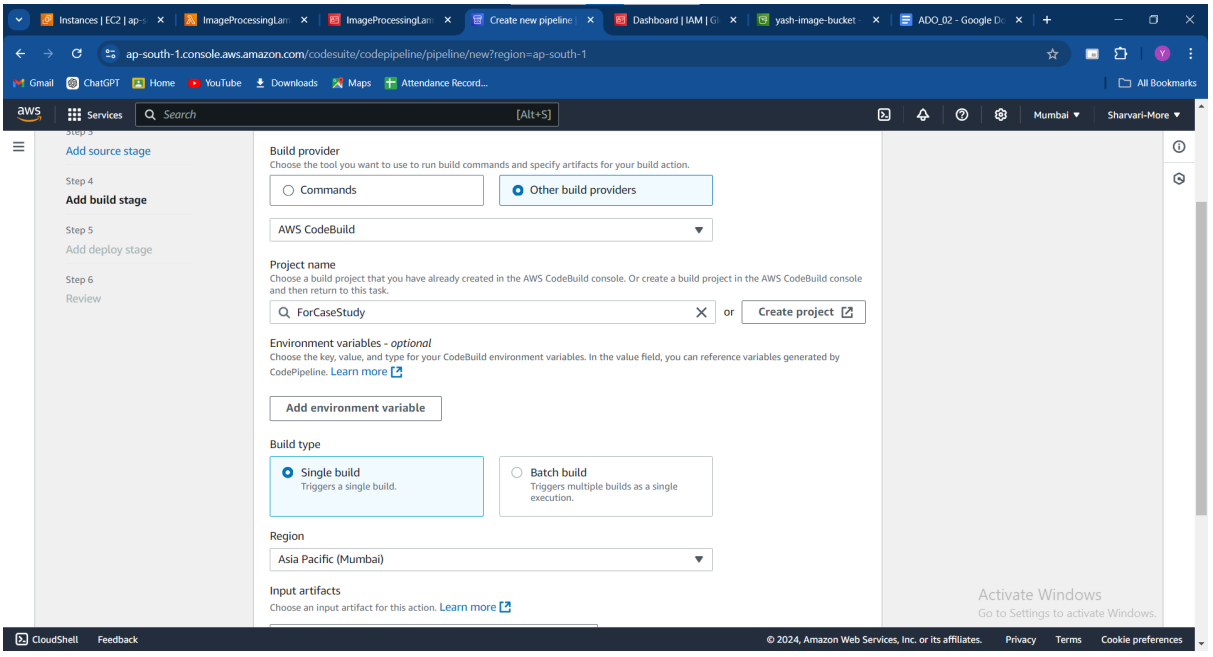


   - Connect your repository that contains the Lambda code (use the same code as in Step 2).

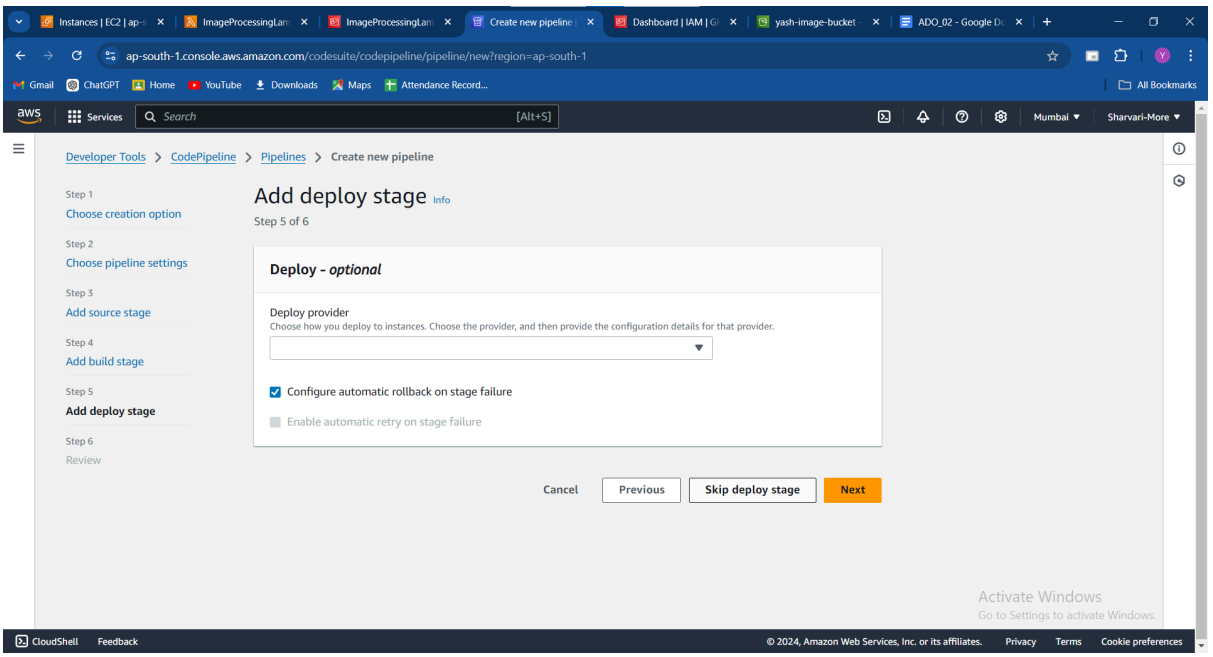


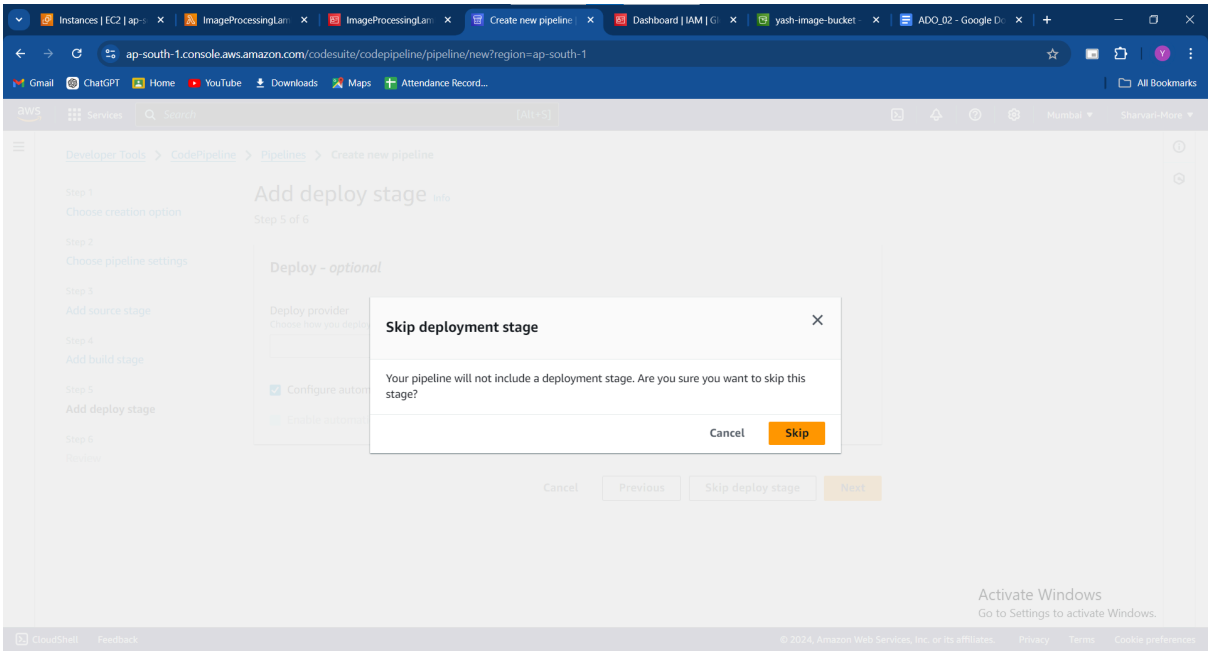
4. **Add CodeBuild to CodePipeline**:

- In your CodePipeline, add **CodeBuild** as the **Build Stage** (instead of a Deploy Stage).
- This will allow CodePipeline to trigger the CodeBuild project, which will run the buildspec.yml commands to package and deploy the Lambda function.

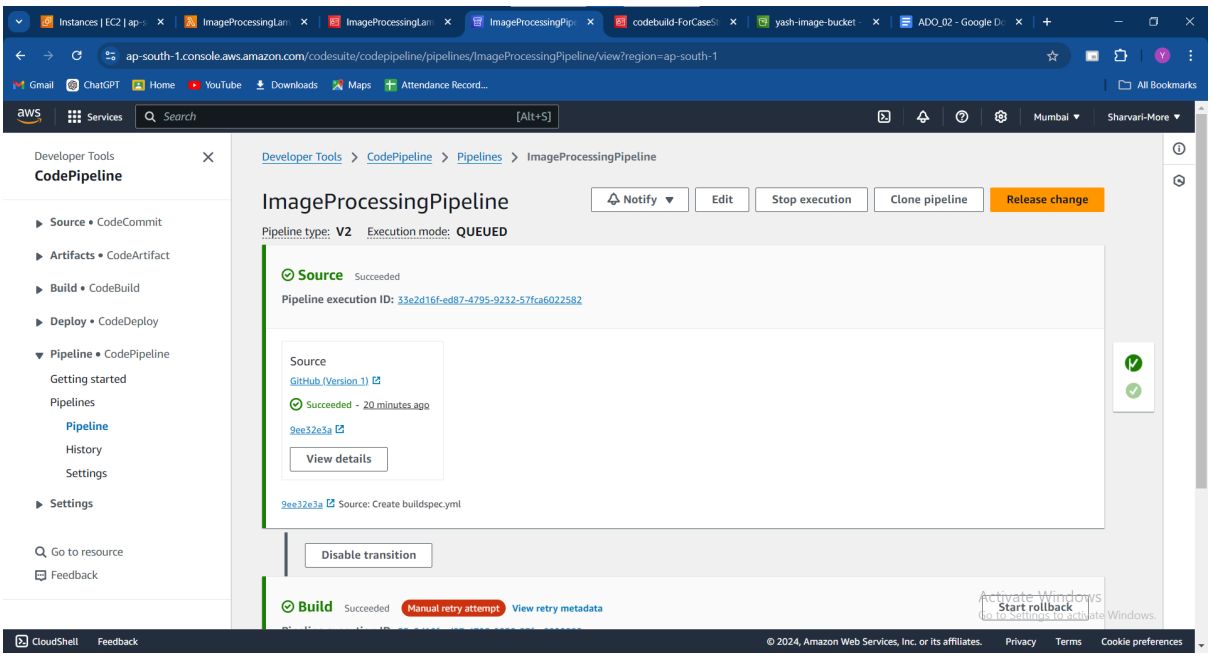


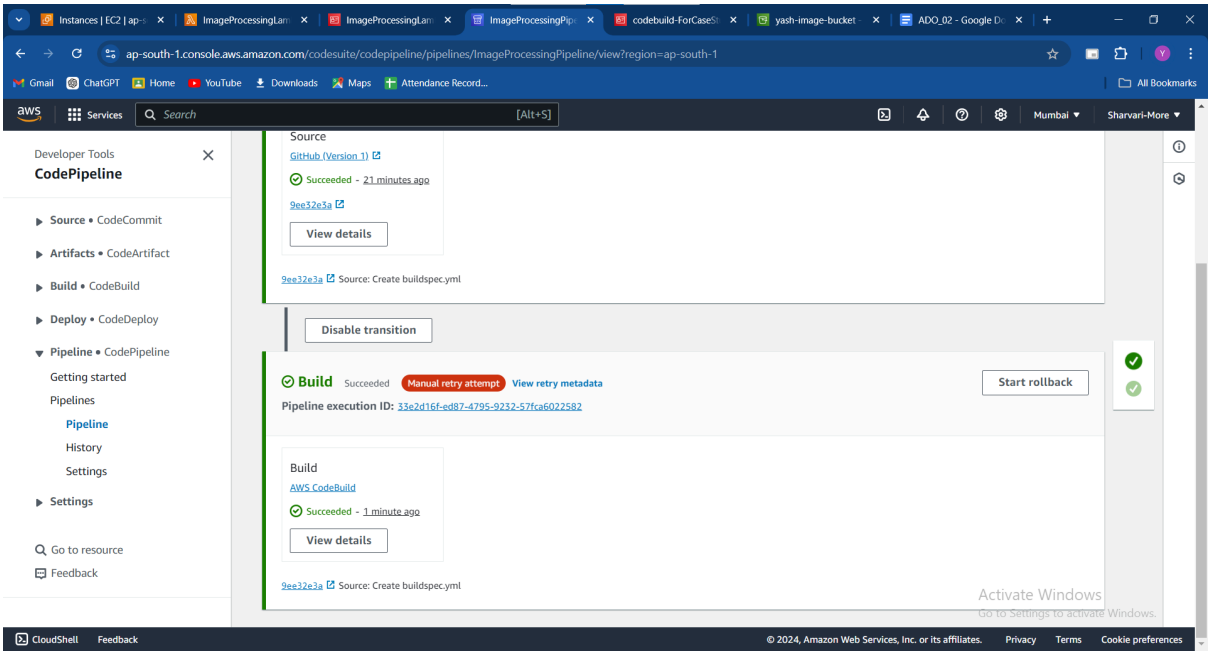
5. **Deploy Stage** (Deploy to Lambda):
    - SKIP THIS (as Choose **AWS Lambda** as the deploy provider Does not exist.)





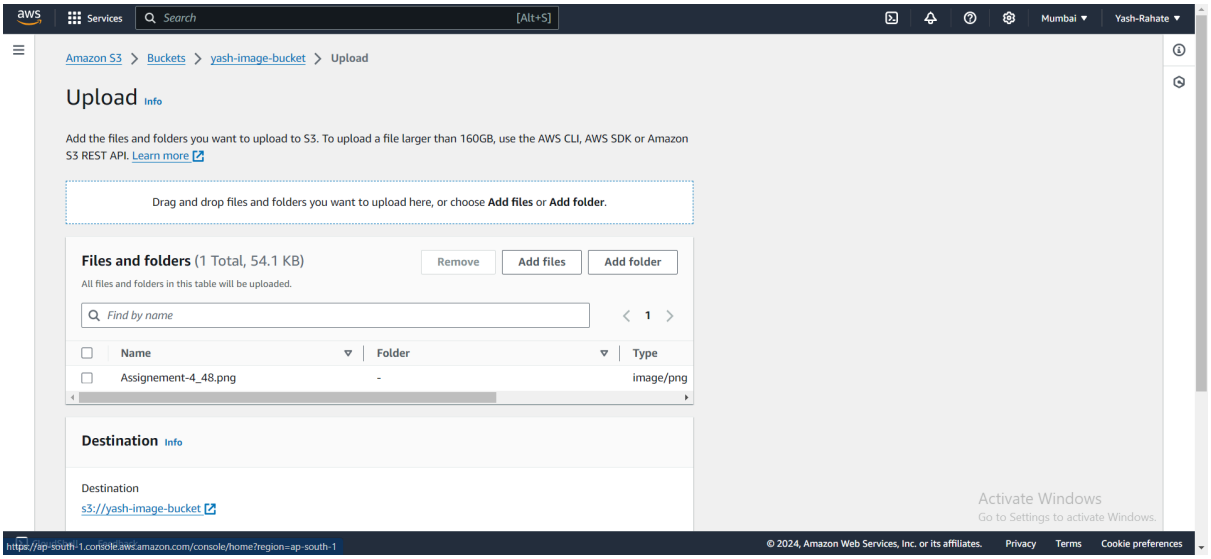
6. Click **Create Pipeline** to finish setting up.



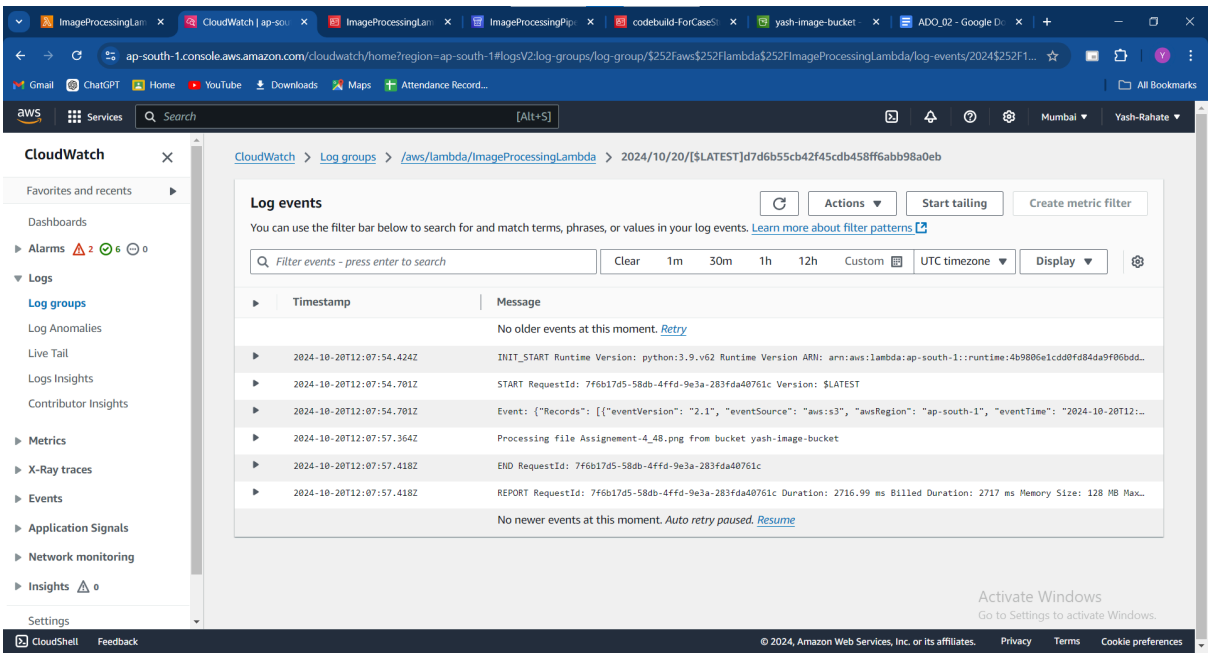


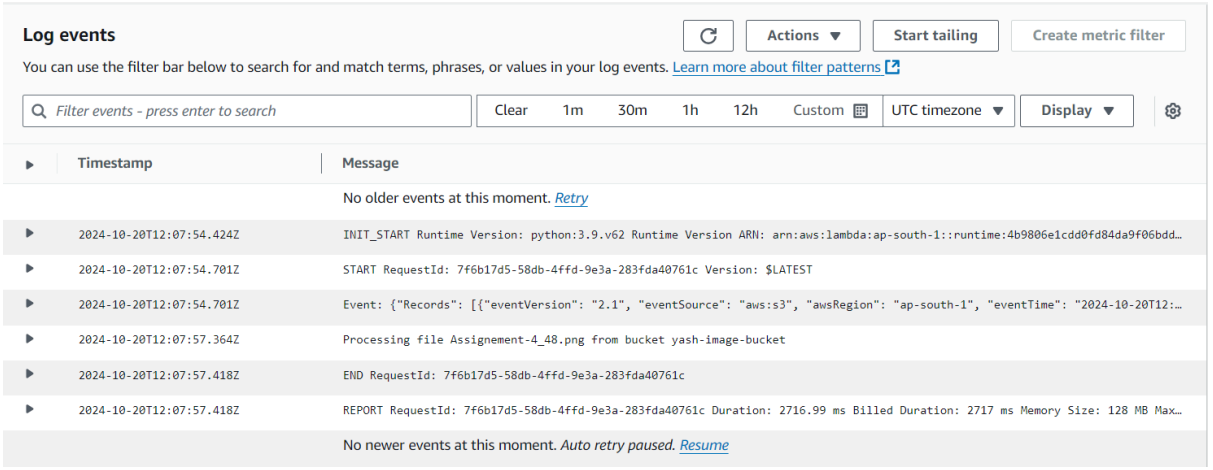
Step 6: Test the Serverless Workflow

1. **Upload a sample image** to your S3 bucket:
   - Go to **S3**, select the bucket image-processing-bucket, and click **Upload**.
   - Upload any image



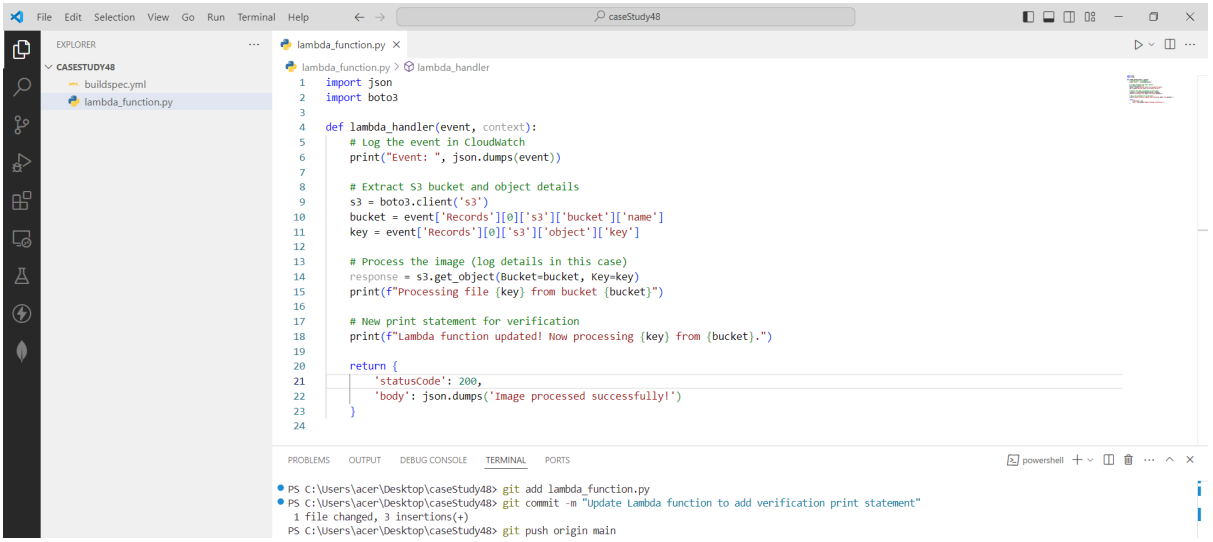
2. **Check CloudWatch Logs**:
   - Go to **CloudWatch** > **Logs** > **Log groups**.
   - You should see a new log group for ImageProcessingLambda.
   - In the logs, you'll see details about the S3 event, including the bucket name and the key (filename).





Step 7: Verify CodePipeline Automation

1. **Make a change** to the Lambda function code (e.g., update the print statement).

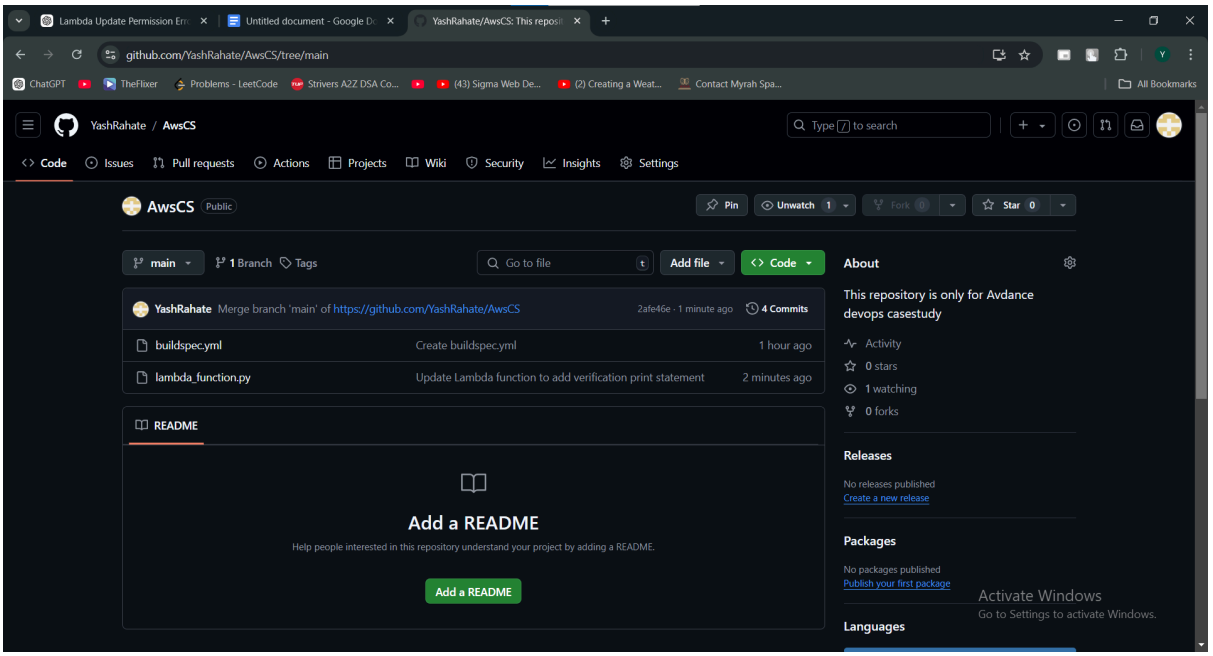


New Code:

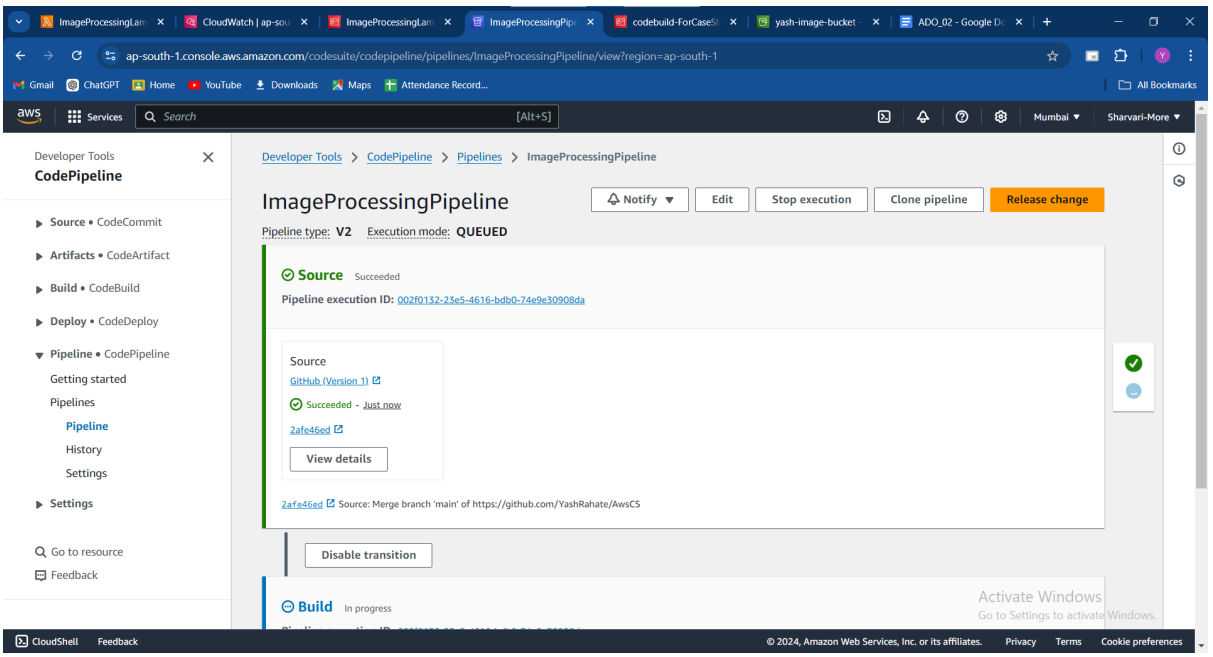
```
import json
import boto3

def lambda_handler(event, context):
    # Log the event in CloudWatch
    print("Event: ", json.dumps(event))

    # Extract S3 bucket and object details
    s3 = boto3.client('s3')
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']

    # Process the image (log details in this case)
    response = s3.get_object(Bucket=bucket, Key=key)
    print(f"Processing file {key} from bucket {bucket}")

    # New print statement for verification
    print(f"Lambda function updated! Now processing {key} from {bucket}.")

    return {
        'statusCode': 200,
        'body': json.dumps('Image processed successfully!')
    }
```

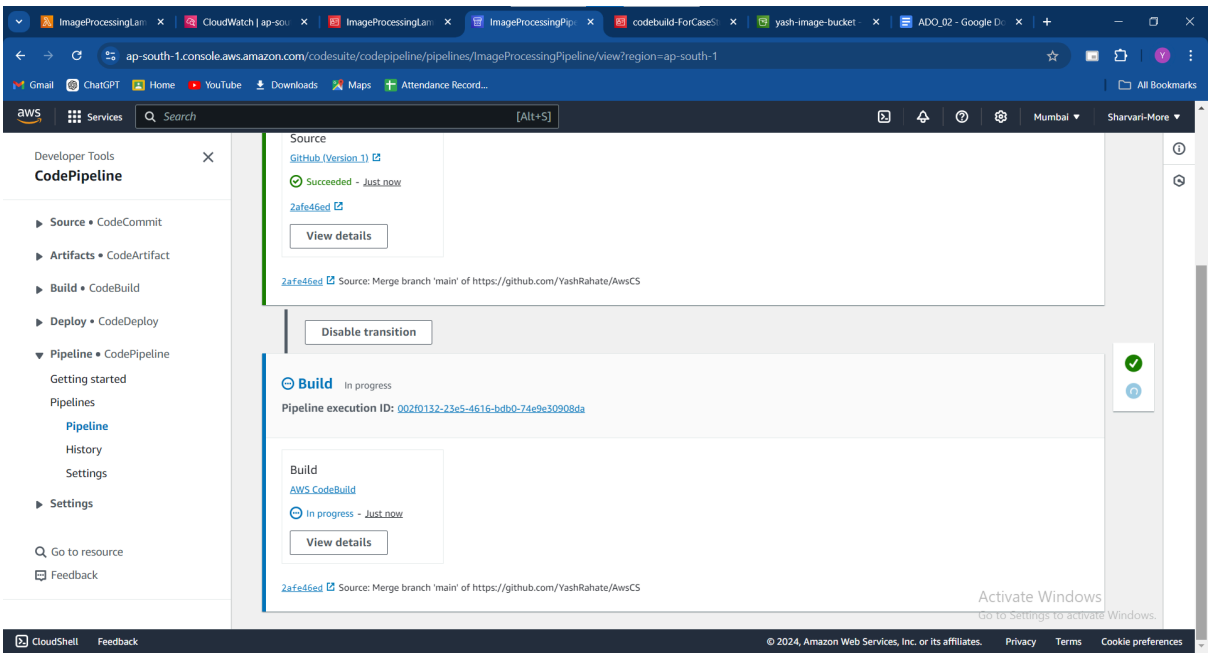
2. **Push the changes** to the GitHub or CodeCommit repository.
   git add lambda_function.py
   git commit -m "Update Lambda function to add verification print statement"
   git push origin main

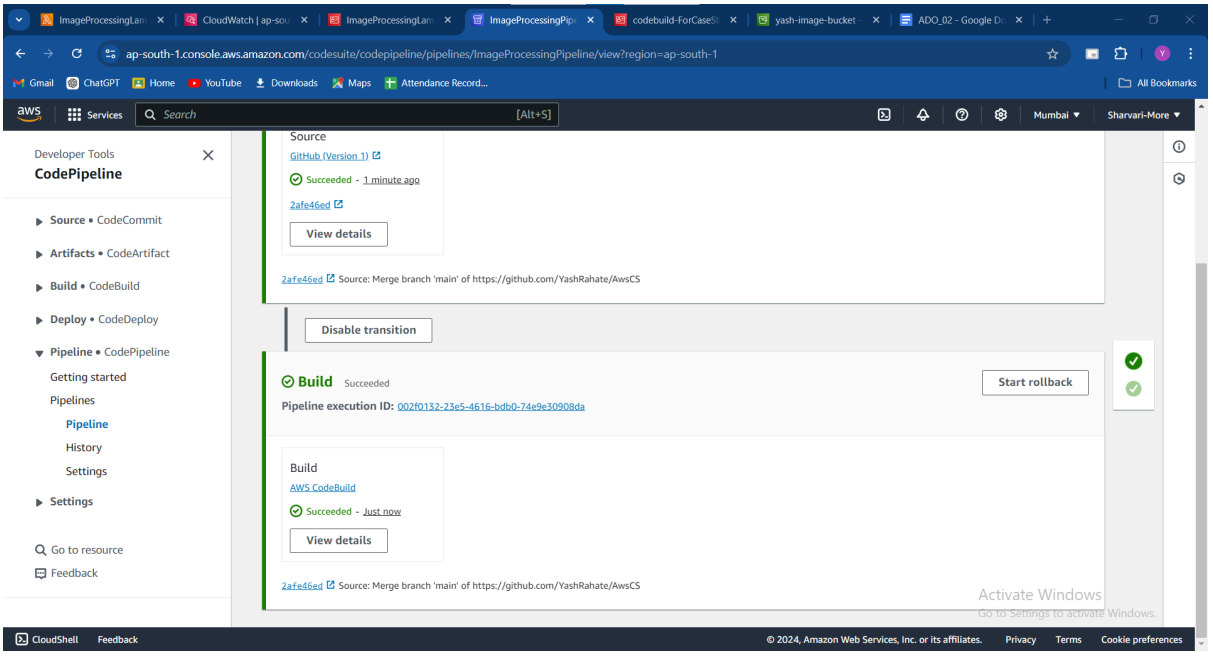
```
PS C:\Users\acer\Desktop\caseStudy48> git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 742 bytes | 371.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/YashRahate/AwsCS.git
```



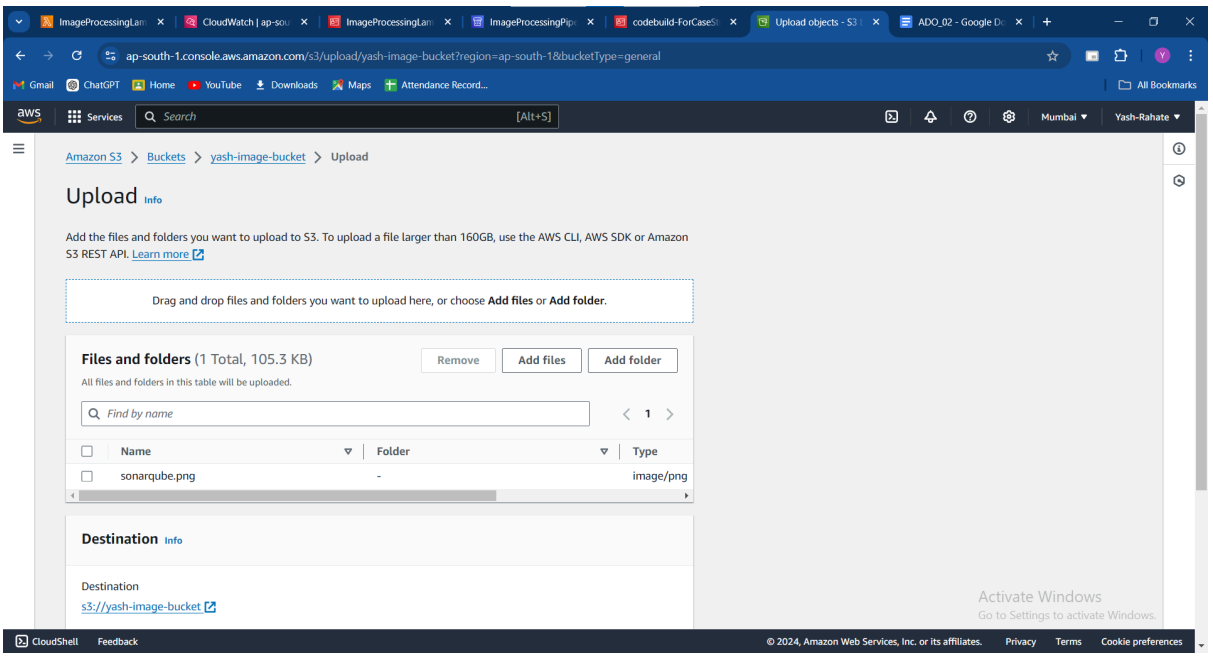
3. CodePipeline will automatically detect the changes and redeploy the updated Lambda function.

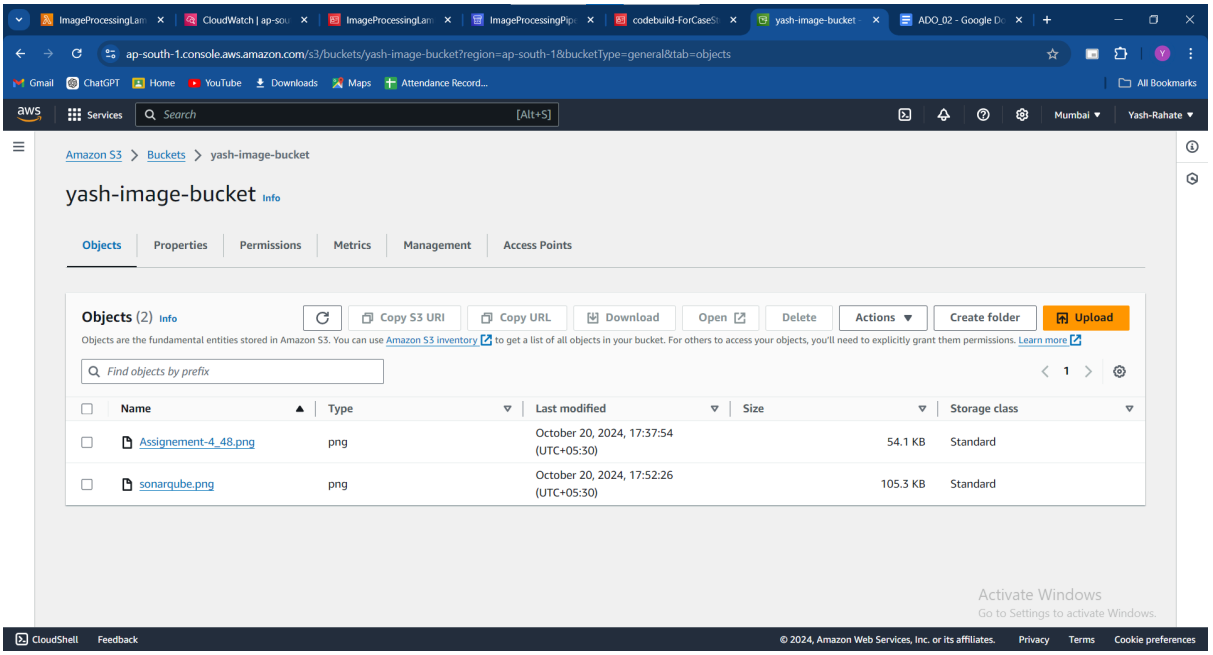

ap-south-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/ImageProcessingPipeline/view?region=ap-south-1
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Developer Tools > CodePipeline > Pipelines > ImageProcessingPipeline
ImageProcessingPipeline
Notify
Edit
Stop execution
Clone pipeline
Release change
Pipeline type: V2
Execution mode: QUEUED
Source Succeeded
Pipeline execution ID: 002f0132-23e5-4616-bdb0-74e9e30908da
Source
GitHub (Version 1)
Succeeded - Just now
2afe46ed
View details
2afe46ed Source: Merge branch 'main' of https://github.com/YashRahate/AwsCS
Disable transition
Build In progress
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

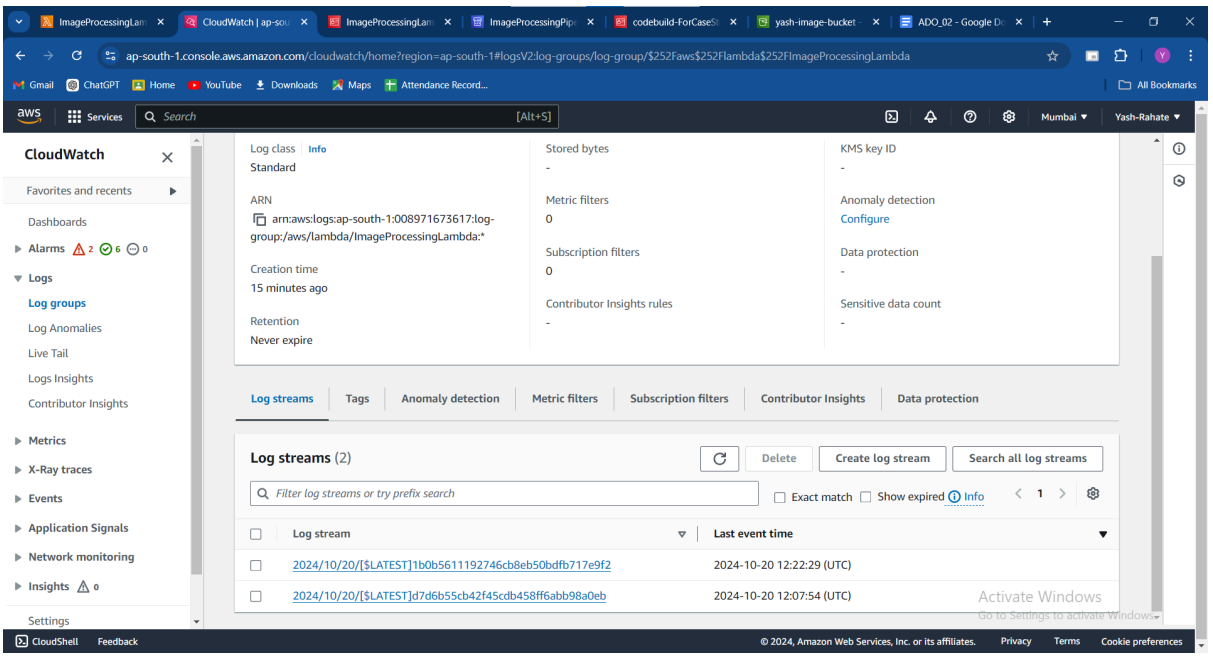


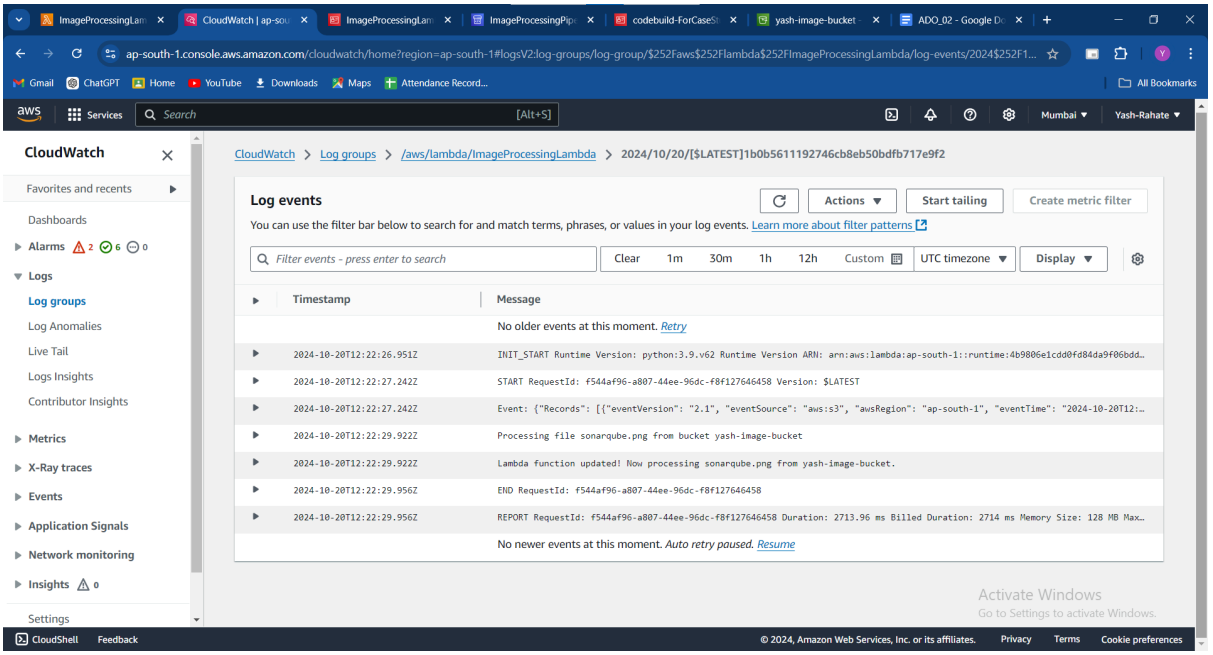


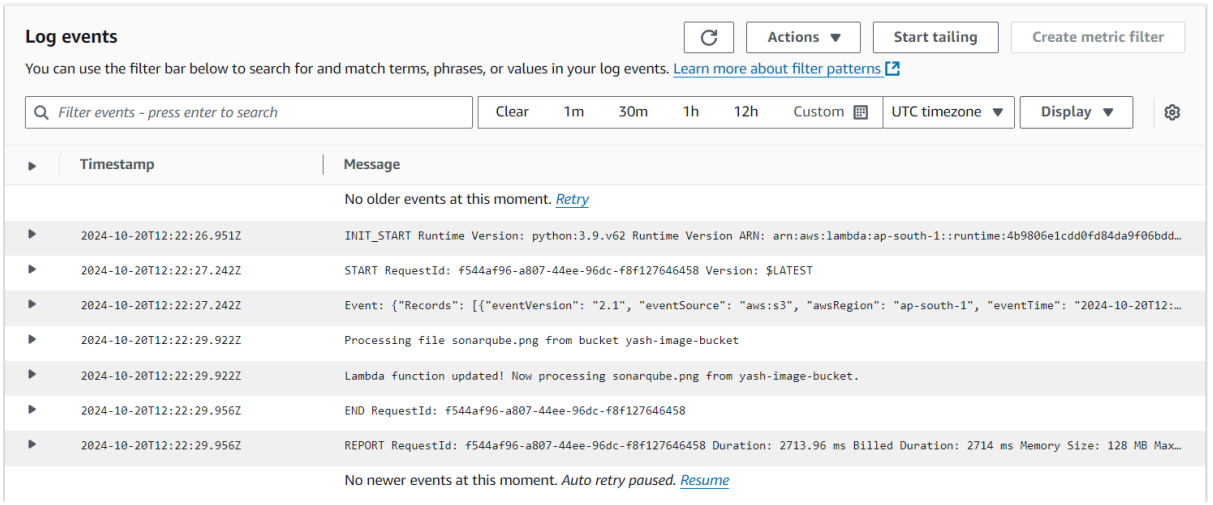
4. Verify that the updated function gets deployed by checking CloudWatch logs after uploading another image.


ap-south-1.console.aws.amazon.com/s3/upload/yash-image-bucket?region=ap-south-1&bucketType=general
Amazon S3 > Buckets > yash-image-bucket > 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 (1 Total, 105.3 KB)
Remove
Add files
Add folder
All files and folders in this table will be uploaded.
Find by name
Name
Folder
Type
sonarqube.png
-
image/png
Destination Info
Destination
s3://yash-image-bucket
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


ap-south-1.console.aws.amazon.com/s3/buckets/yash-image-bucket?region=ap-south-1&bucketType=general&tab=objects
Amazon S3 > Buckets > yash-image-bucket
yash-image-bucket Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (2) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name
Type
Last modified
Size
Storage class
Assignement-4_48.png
png
October 20, 2024, 17:37:54 (UTC+05:30)
54.1 KB
Standard
sonarqube.png
png
October 20, 2024, 17:52:26 (UTC+05:30)
105.3 KB
Standard
Activate Windows
Go to Settings to activate Windows.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


CloudWatch
Log class Info
Standard
ARN
arn:aws:logs:ap-south-1:008971673617:log-group:/aws/lambda/ImageProcessingLambda:*
Creation time
15 minutes ago
Retention
Never expire
Stored bytes
-
Metric filters
0
Subscription filters
0
Contributor Insights rules
-
KMS key ID
-
Anomaly detection
Configure
Data protection
-
Sensitive data count
-
Log streams
Tags
Anomaly detection
Metric filters
Subscription filters
Contributor Insights
Data protection
Log streams (2)
Delete
Create log stream
Search all log streams
Filter log streams or try prefix search
Exact match
Show expired Info
Log stream
Last event time
2024/10/20/[$LATEST]1b0b5611192746cb8eb50bdfb717e9f2
2024-10-20 12:22:29 (UTC)
2024/10/20/[$LATEST]d7d6b55cb42f45cdb458ff6abb98a0eb
2024-10-20 12:07:54 (UTC)
Activate Windows
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


CloudWatch
CloudWatch > Log groups > /aws/lambda/ImageProcessingLambda > 2024/10/20/[$LATEST]1b0b5611192746cb8eb50bdfb717e9f2
Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display
Timestamp
Message
No older events at this moment. Retry
2024-10-20T12:22:26.951Z INIT_START Runtime Version: python:3.9.v62 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:4b9806e1cdd0fd84da9f06bdd…
2024-10-20T12:22:27.242Z START RequestId: f544af96-a807-44ee-96dc-f8f127646458 Version: $LATEST
2024-10-20T12:22:27.242Z Event: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "ap-south-1", "eventTime": "2024-10-20T12:…
2024-10-20T12:22:29.922Z Processing file sonarqube.png from bucket yash-image-bucket
2024-10-20T12:22:29.922Z Lambda function updated! Now processing sonarqube.png from yash-image-bucket.
2024-10-20T12:22:29.956Z END RequestId: f544af96-a807-44ee-96dc-f8f127646458
2024-10-20T12:22:29.956Z REPORT RequestId: f544af96-a807-44ee-96dc-f8f127646458 Duration: 2713.96 ms Billed Duration: 2714 ms Memory Size: 128 MB Max…
No newer events at this moment. Auto retry paused. Resume
Activate Windows
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display
Timestamp
Message
No older events at this moment. Retry
2024-10-20T12:22:26.951Z INIT_START Runtime Version: python:3.9.v62 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:4b9806e1cdd0fd84da9f06bdd…
2024-10-20T12:22:27.242Z START RequestId: f544af96-a807-44ee-96dc-f8f127646458 Version: $LATEST
2024-10-20T12:22:27.242Z Event: {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "ap-south-1", "eventTime": "2024-10-20T12:…
2024-10-20T12:22:29.922Z Processing file sonarqube.png from bucket yash-image-bucket
2024-10-20T12:22:29.922Z Lambda function updated! Now processing sonarqube.png from yash-image-bucket.
2024-10-20T12:22:29.956Z END RequestId: f544af96-a807-44ee-96dc-f8f127646458
2024-10-20T12:22:29.956Z REPORT RequestId: f544af96-a807-44ee-96dc-f8f127646458 Duration: 2713.96 ms Billed Duration: 2714 ms Memory Size: 128 MB Max…
No newer events at this moment. Auto retry paused. Resume

Conclusion

This workflow will set up a fully serverless image processing system that triggers an AWS Lambda function whenever a new image is uploaded to S3, and it will automate the deployment using AWS CodePipeline.