Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Advance Devops Practical Examination: AWS Case Study Assignment

Topic : Serverless Image Processing Workflow

1. Introduction

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.

Theory:

1. AWS Lambda

AWS Lambda is a **serverless compute service** that allows you to run code without provisioning or managing servers. It automatically scales your application by running code in response to triggers, such as changes in data or HTTP requests, without you needing to worry about infrastructure. Lambda supports various languages (e.g., Python, Node.js, Java, etc.).

- **Key Features**:
  - **Event-driven**: Triggered by events (e.g., S3 uploads, API Gateway requests).
  - **Auto-scaling**: Scales based on demand.
  - **Pay-as-you-go**: Charges based on the number of requests and the compute time used.

2. Amazon S3 (Simple Storage Service)

Amazon S3 is an **object storage service** that allows you to store and retrieve any amount of data from anywhere. It is widely used for storing large amounts of unstructured data like images, videos, backups, and logs.

- **Key Features**:
    - **Durability & Scalability**: Highly durable and scalable for storing large data sets.
    - **Access Control**: Offers fine-grained access controls using policies and permissions.
    - **Integration**: Integrates with other AWS services like Lambda, CloudFront, and Glacier for archiving.

3. AWS CodePipeline

AWS CodePipeline is a **continuous integration and continuous delivery (CI/CD) service** that automates the software release process. It defines a workflow (pipeline) for building, testing, and deploying your code automatically.

- **Key Features**:
    - **Automation**: Automates code deployment after every change.
    - **Multi-stage Pipeline**: Supports stages like source control, build, and deploy.
    - **Integration**: Integrates with tools like GitHub, Jenkins, and AWS services like CodeBuild and Lambda.

4. AWS IAM (Identity and Access Management)

AWS IAM is a service that helps you manage **access to AWS resources** securely. It allows you to create users, groups, and roles and assign permissions to control who can access what.

- **Key Features**:
    - **Granular Permissions**: Control access to AWS services and resources at a fine level.
    - **Multi-factor Authentication (MFA)**: Adds an extra layer of security for users.
    - **Roles**: Allows temporary access for services or users to perform specific tasks.

5.AWS CloudWatch:

AWS CloudWatch is a monitoring service that provides data and actionable insights to monitor applications, understand system-wide performance changes, and optimize resource usage. In this experiment, CloudWatch was used to capture logs from the Lambda function, allowing us to track S3 events and troubleshoot any issues that occurred during image uploads and processing.

- **Key Features**:
    - Logs: Collects and stores logs from AWS services like Lambda.
    - Metrics: Provides metrics based on the logs and AWS services.
    - Alerts: Configures alarms based on metrics to monitor resources.

6. AWS CodeBuild:

AWS CodeBuild is a fully managed build service in AWS that compiles source code, runs tests, and produces software packages. In this experiment, CodeBuild was used as the build stage of the pipeline to package the Lambda function code and deploy it automatically using AWS CLI.

- **Key Features**:
    - Continuous Integration: Automates the build and test phases.
    - Scalable: Scales automatically to handle multiple builds.
    - Integration: Easily integrates with other AWS services like CodePipeline and Lambda.

7. S3 Event Notifications:

S3 Event Notifications enable you to receive notifications when specific events occur in an S3 bucket. In this case study, S3 Event Notifications were used to trigger the Lambda function when an image was uploaded, ensuring the function is only invoked when necessary.

- **Key Features**:
    - Event Triggers: Supports events like object creation, deletion, or replication.
    - Destination: Can trigger Lambda functions, SNS, or SQS notifications.
    - Granular Control: Can configure events to be triggered for specific objects or prefixes in the bucket.

Case Study Overview:
The case study involves building a **serverless image processing workflow** using AWS services such as Lambda, S3, and CodePipeline. The goal is to create a serverless workflow that automatically triggers a Lambda function when a new image is uploaded to an S3 bucket. Additionally, the project sets up CodePipeline to automate the deployment of Lambda function updates.

Key Feature and Application:
The case study demonstrates how AWS Lambda can process events triggered by S3 uploads, facilitating **real-time image processing** without manual intervention. The project also highlights how CodePipeline automates Lambda function deployment, ensuring **continuous integration and continuous delivery (CI/CD)**.

Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

2. Step-by-Step Explanation

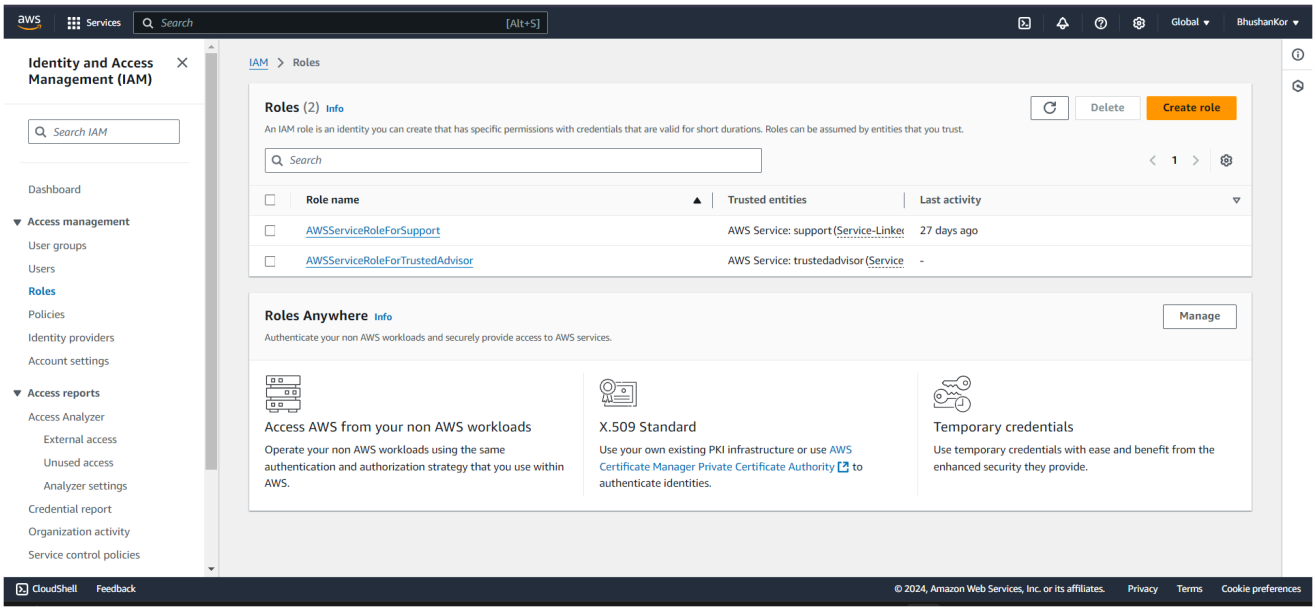
1. Create an IAM Role :

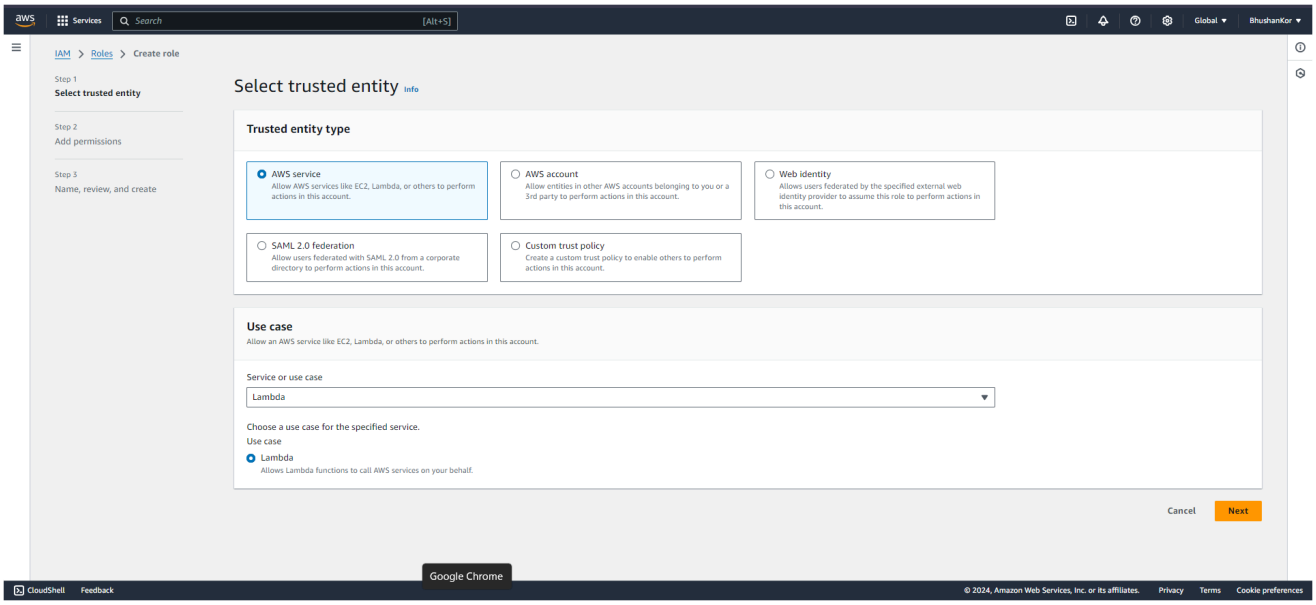
- Log in to the **AWS Management Console**.
- Go to the **IAM Console** in AWS.
- Click on **Roles** and choose **Create Role**.
- Select **AWS service** as the trusted entity type, and choose **Lambda** as the use case.
- Attach policies such as:
    - **AmazonS3FullAccess**
    - **AmazonCodeBuildAdminAcess**
    - **AmazonCodePipeline_FullAccess**
    - **AWSLambda_FullAccess**
    - **CloudWatchLogsFullAccess**
- Give the role a name: **AWS_Case_Study** and create it.
- In Trust Realtionship/Policy attach below policy in Statement.

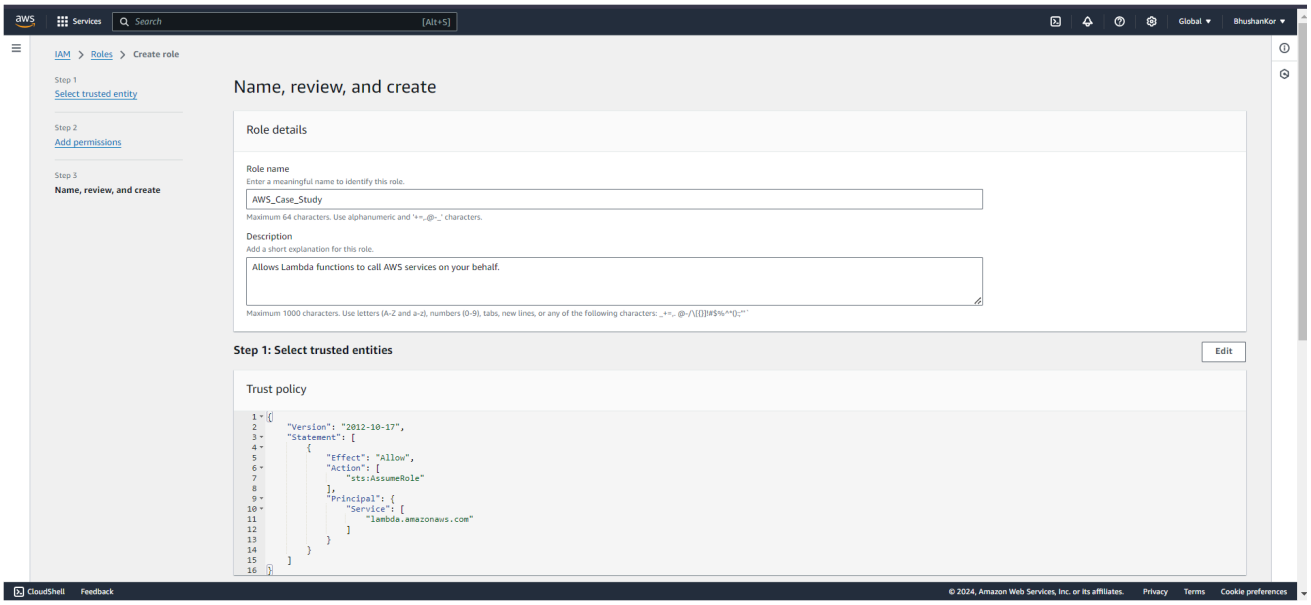
```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "codepipeline.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
},
{
    "Effect": "Allow",
    "Principal": {
        "Service": "codebuild.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
}
```

Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

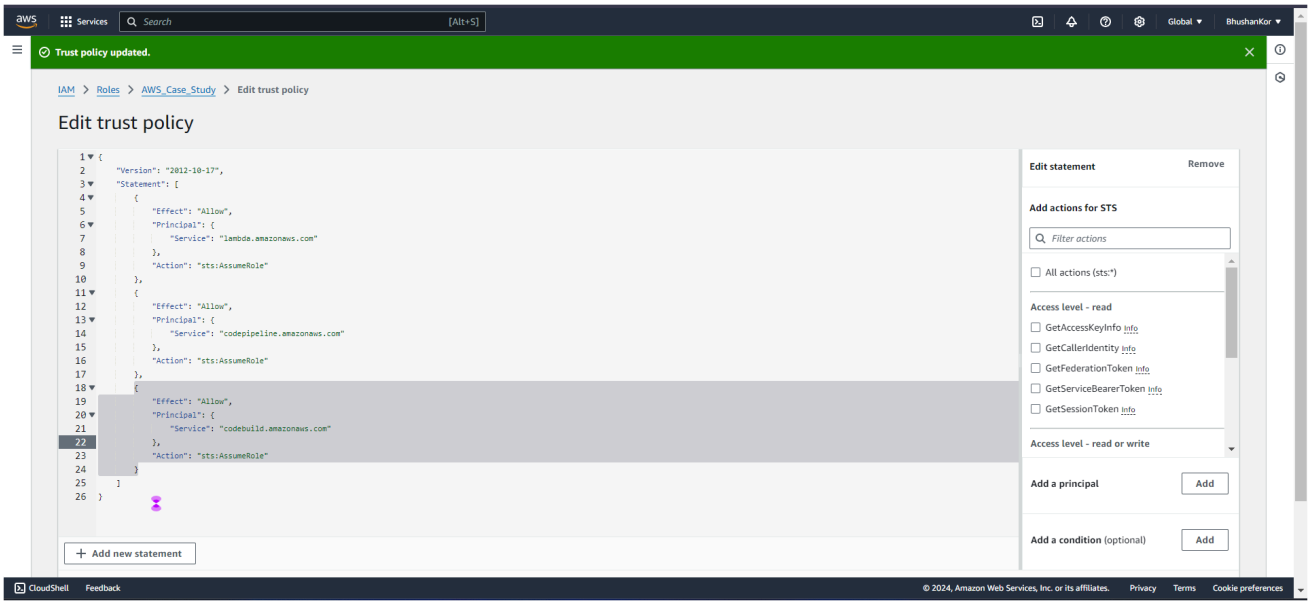


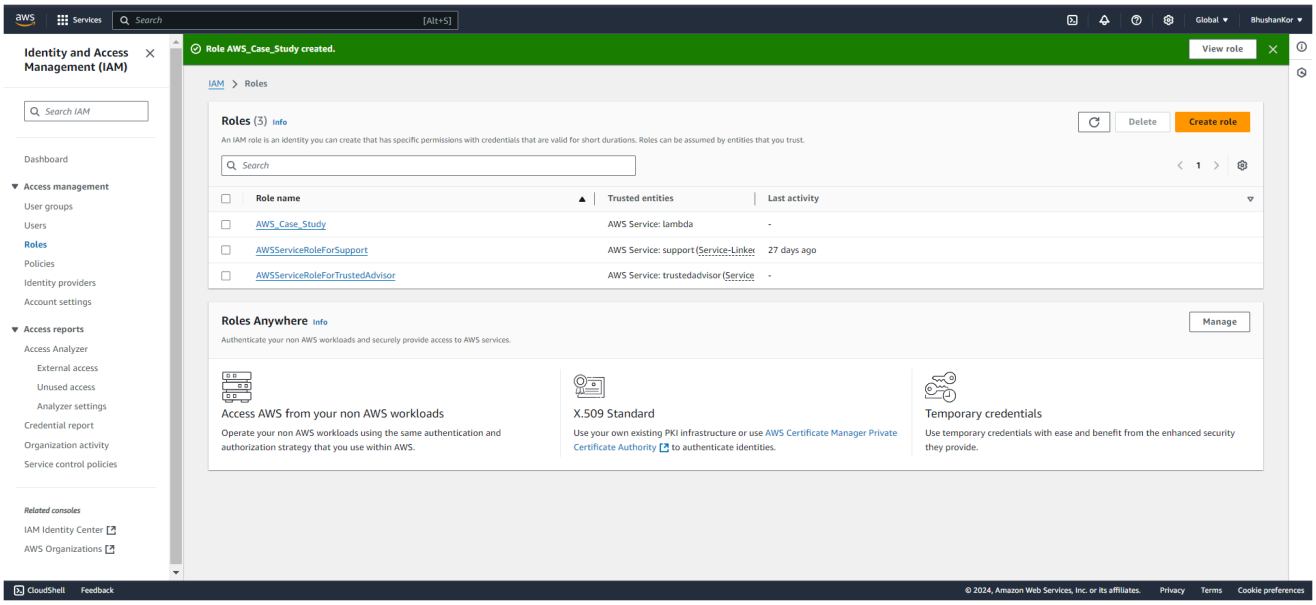




Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**





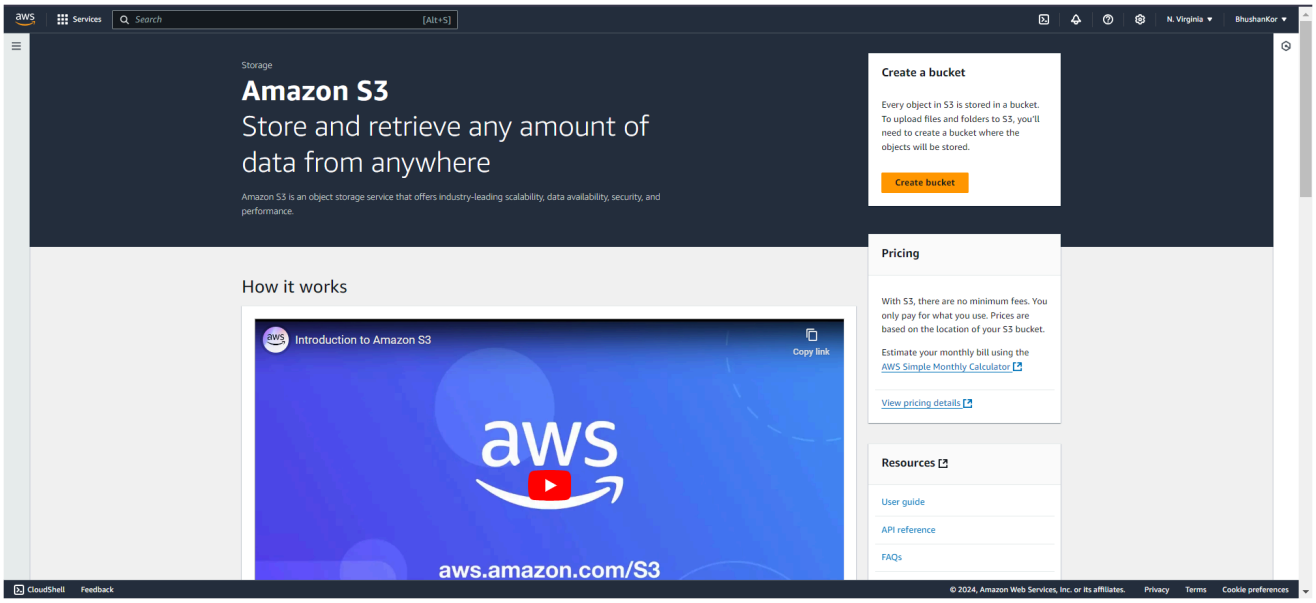
2. Create an S3 Bucket:

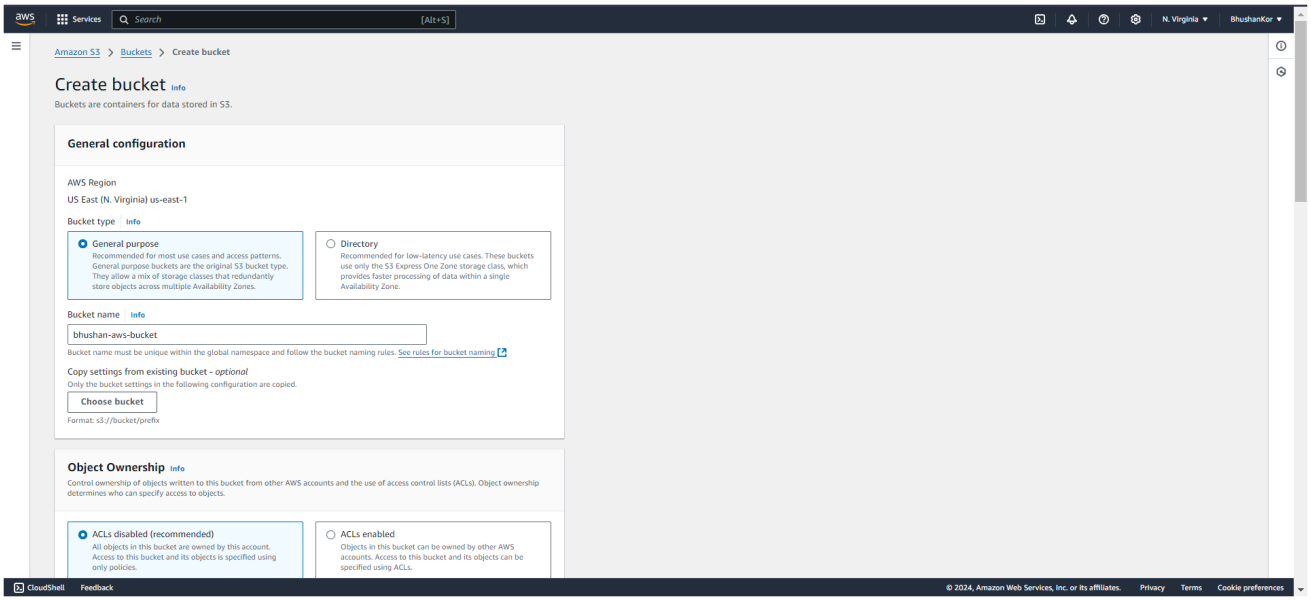
- Navigate to the **S3** service.
- Click on **Create Bucket**.
  - Bucket Type: **General Purpose.**
  - Provide a unique bucket name. **(bhushan-aws-bucket)**
  - **Uncheck the Block all public access option.**
  - Keep Rest of the things to default.

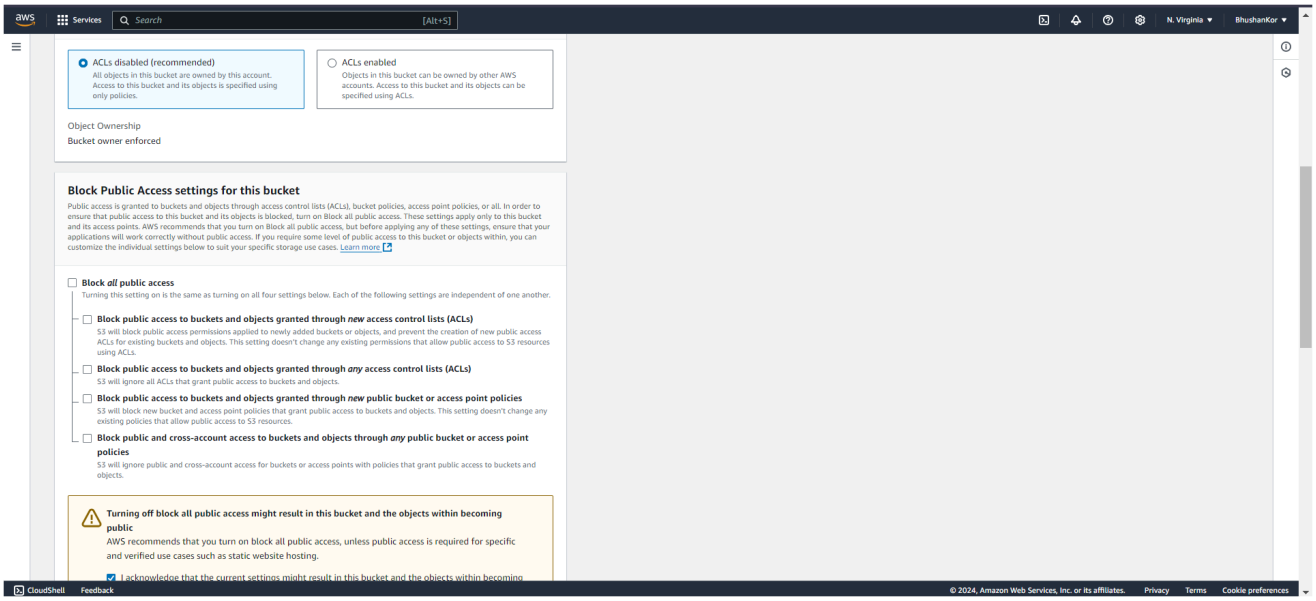
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

- You can upload a test image to ensure the bucket is working properly.

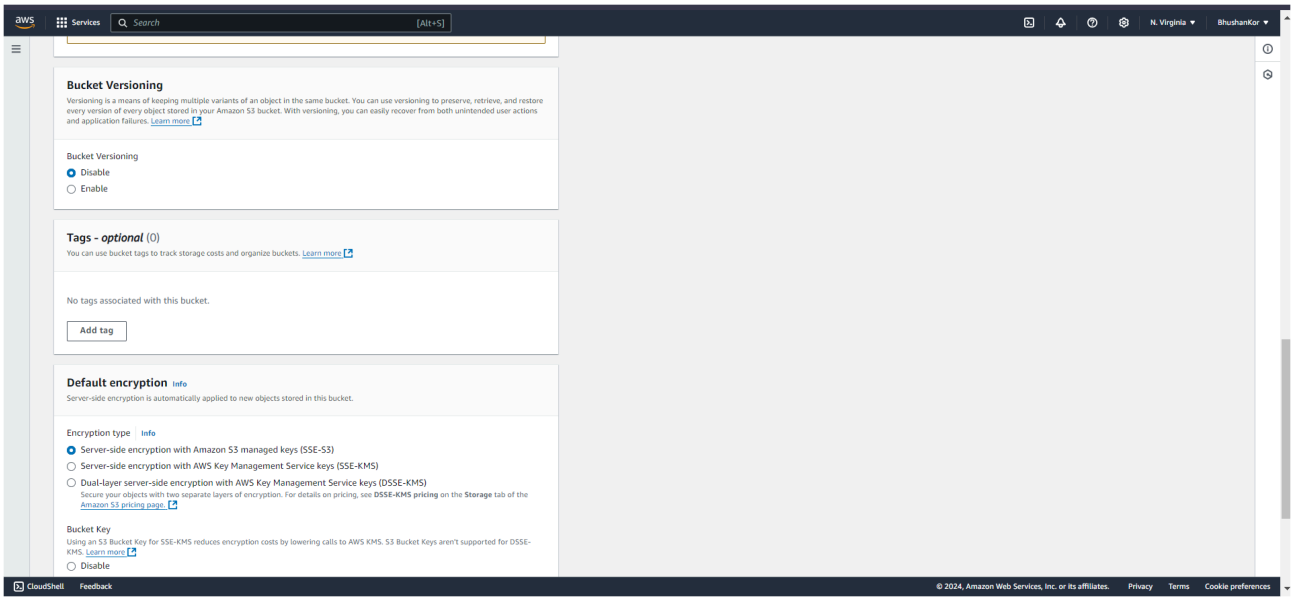


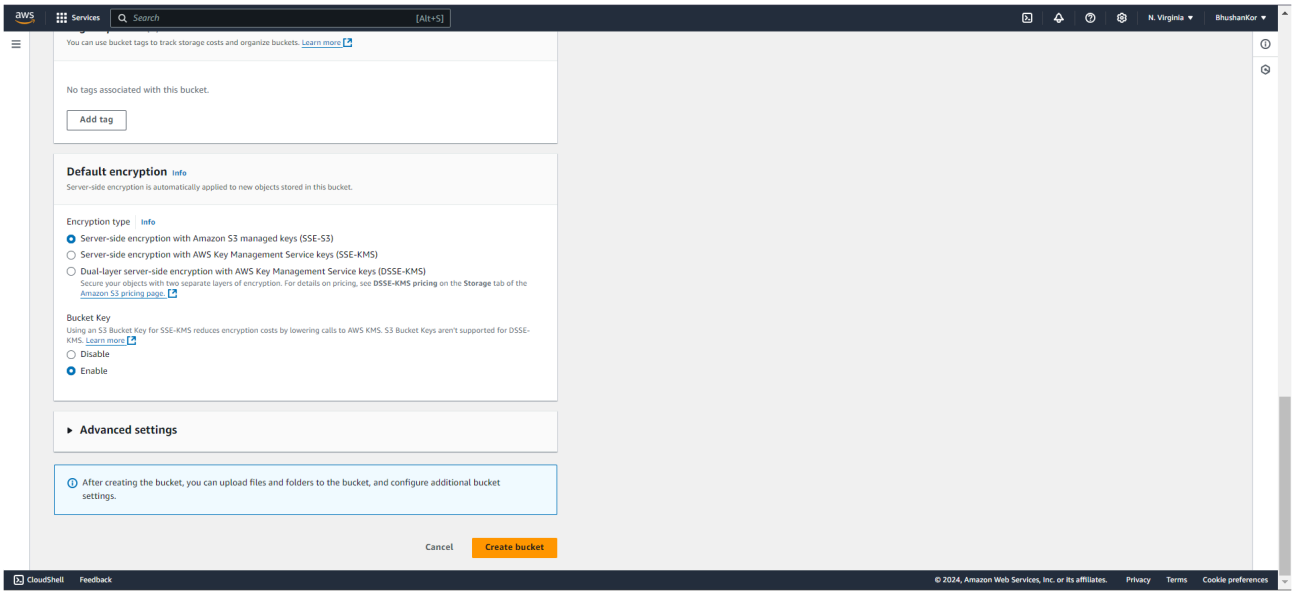


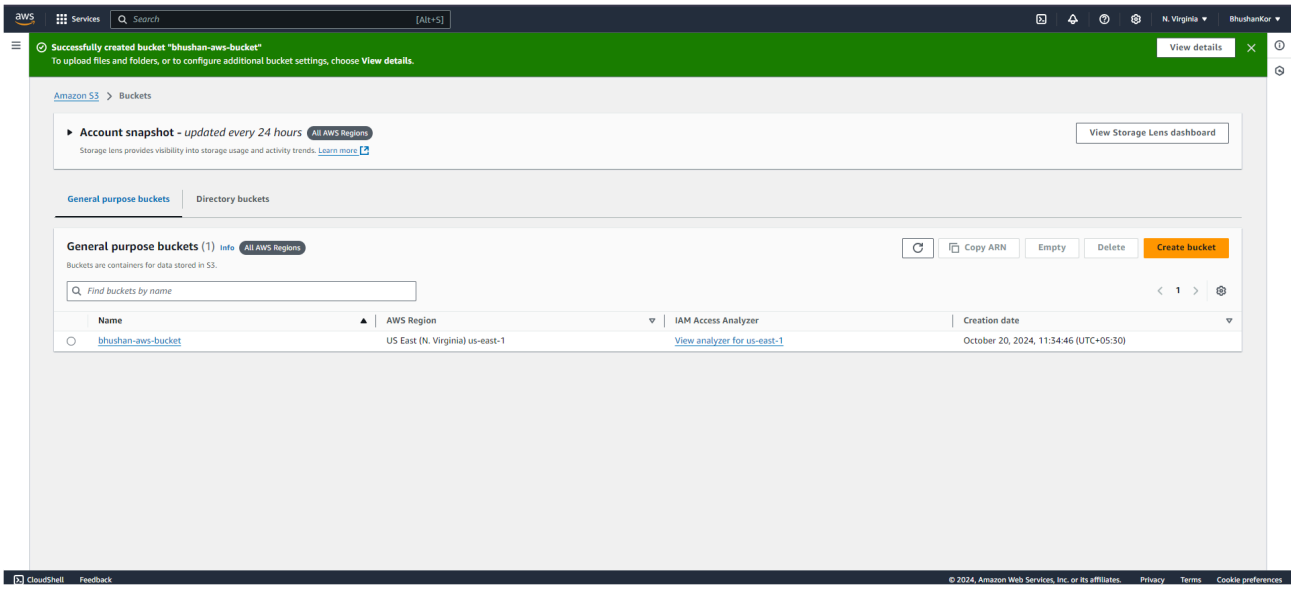


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**







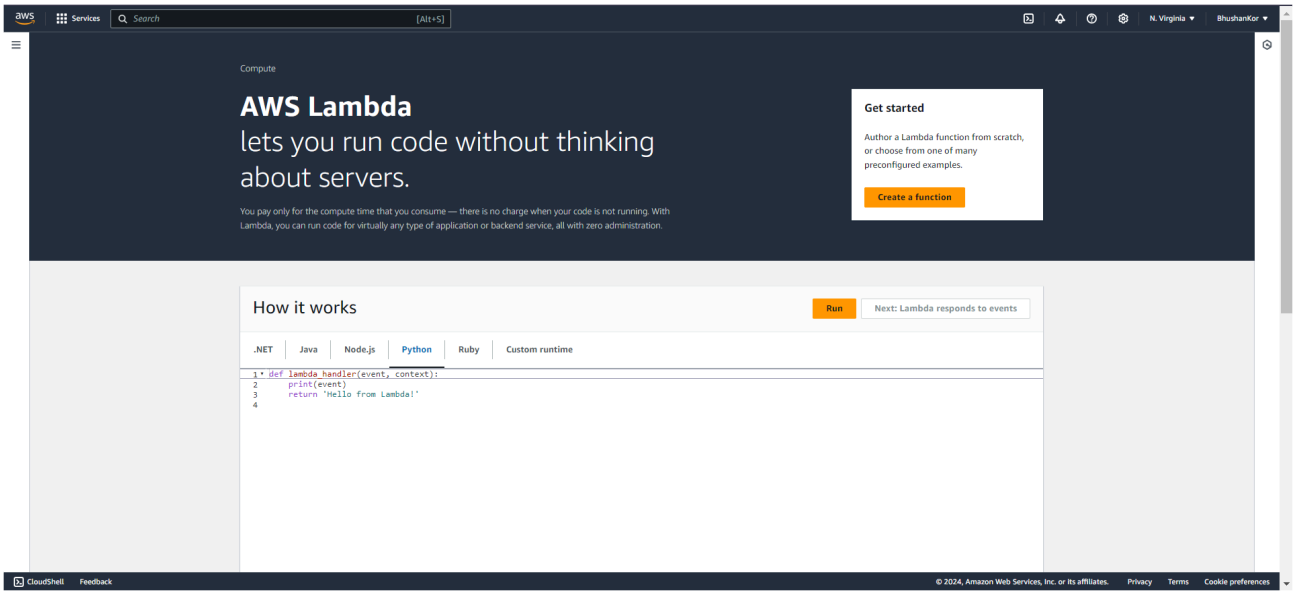
3. Create a Lambda Function:

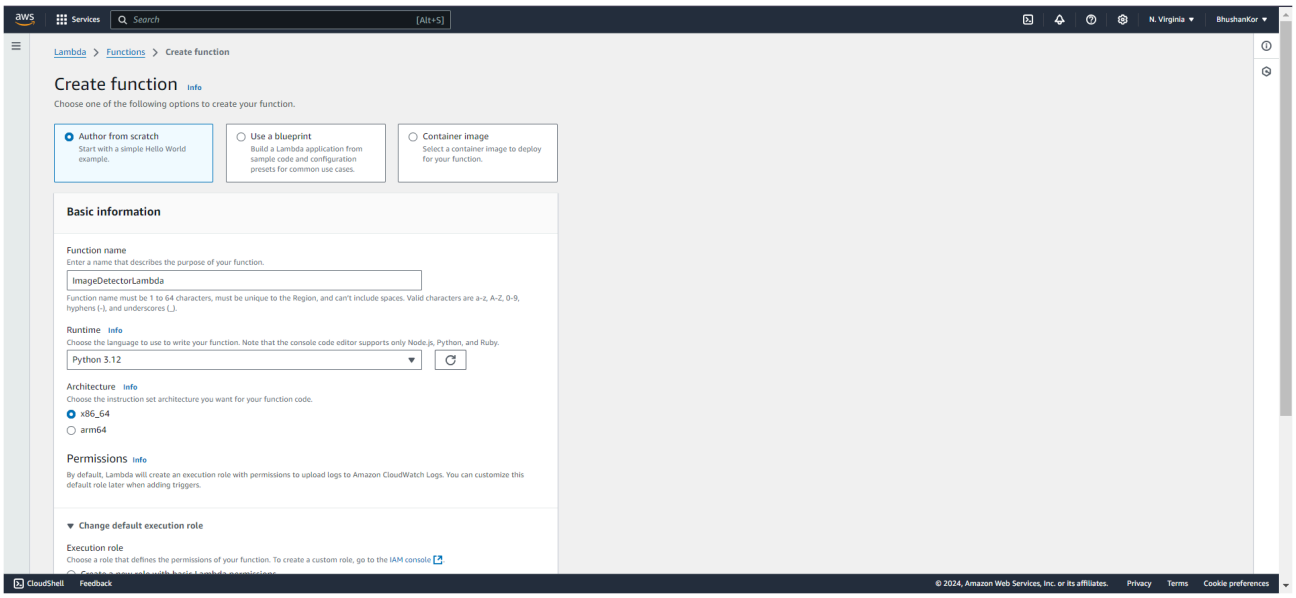
- Navigate to the **Lambda Console**.
- Click **Create Function**.
  - Choose **Author from scratch**.
  - Provide a function name **ImageDetectorLambda**.
  - Choose a free-tier eligible runtime **Python 3.12**.
  - Assign the **IAM role** created earlier **AWS_Case_Study**.
  - Keep Rest of things to default.
- In the **Function Code** section, add the logic to log when the image is uploaded to S3.

  Code:

```
import json

import logging


logger = logging.getLogger()

logger.setLevel(logging.INFO)


def lambda_handler(event, context):

    # Log the event details

    logger.info(f"Received event: {json.dumps(event)}")

    # Extract bucket name and object key (file name)

    bucket_name = event['Records'][0]['s3']['bucket']['name']

    object_key = event['Records'][0]['s3']['object']['key']

    logger.info(f"New image added: {object_key} in bucket: {bucket_name}")

    return {

        'statusCode': 200,

        'body': json.dumps('Image processed successfully.')

    }
```

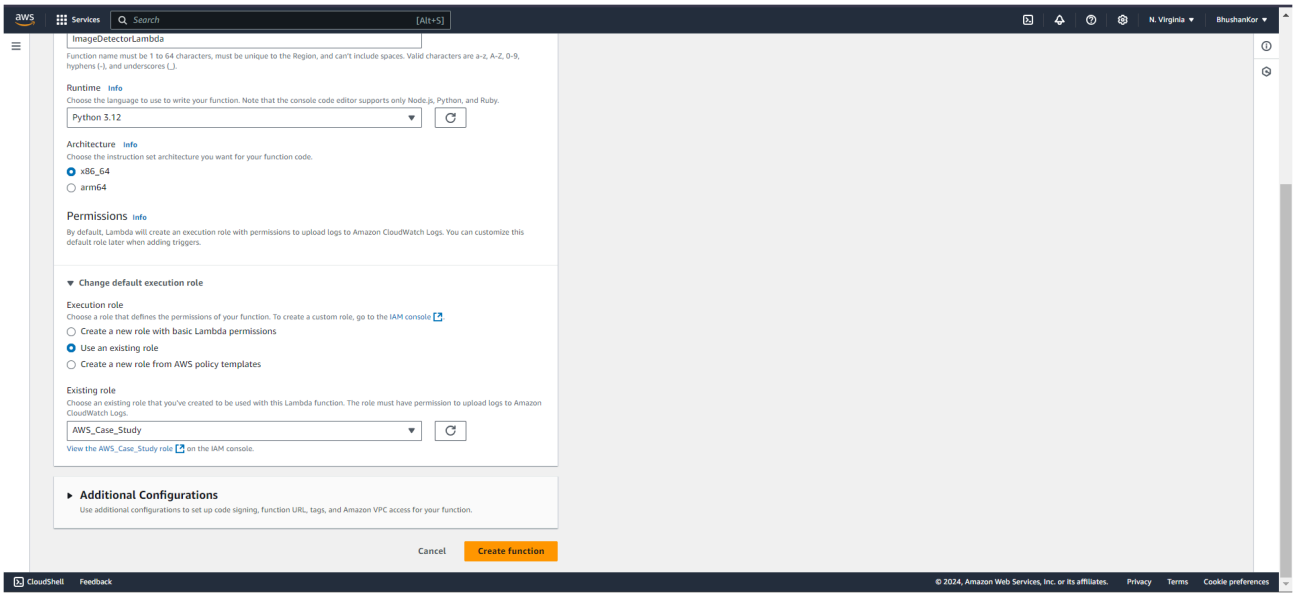
- Now Add the **Trigger** (Click on Trigger).
  - In Trigger Configuration Select S3.
  - Select Bucket which we have created.**(bhushan-aws-bucket)**
  - Select Event Types
    - All object create events
    - PUT
    - POST
    - COPY
    - Multipart upload completed

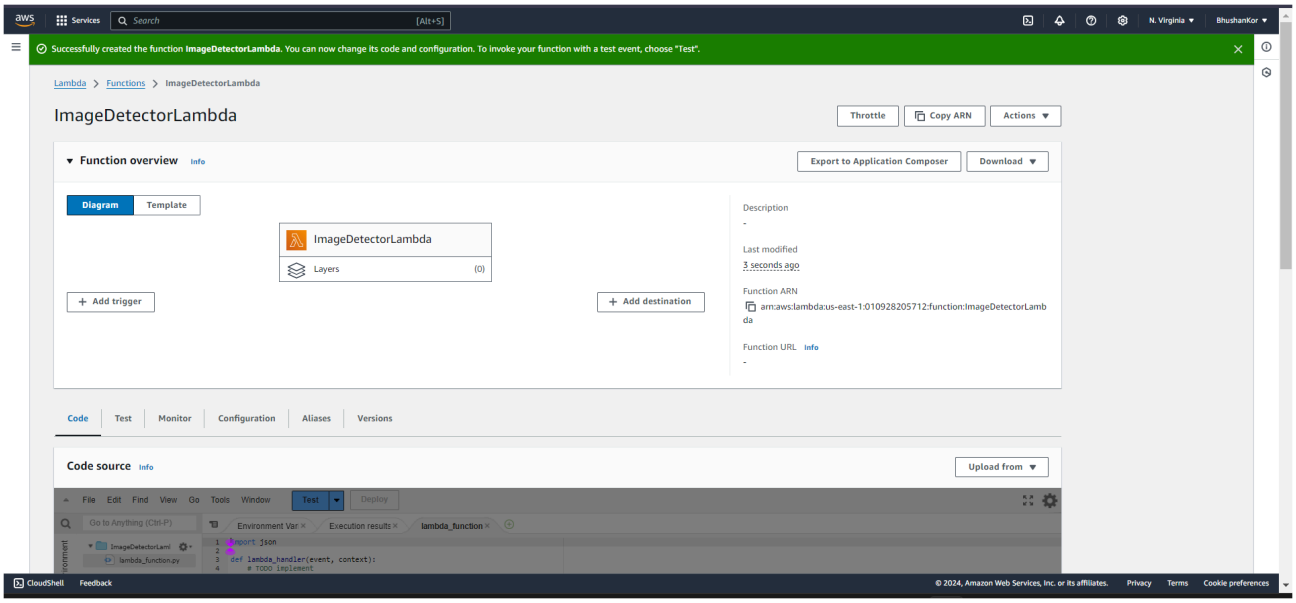


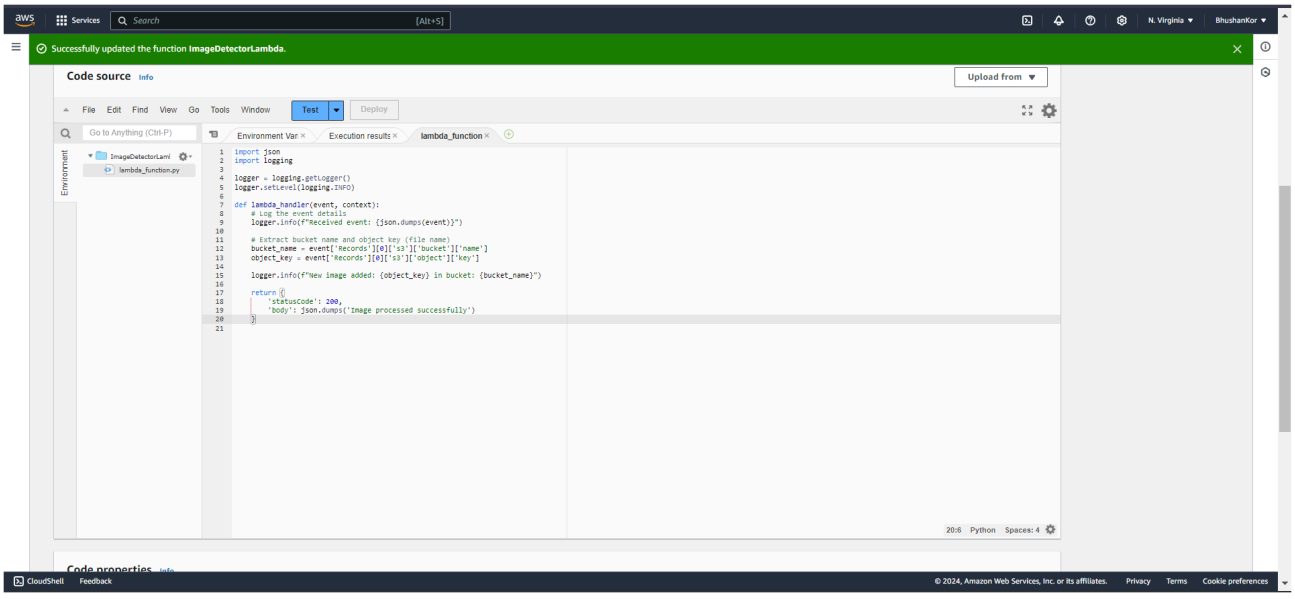


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

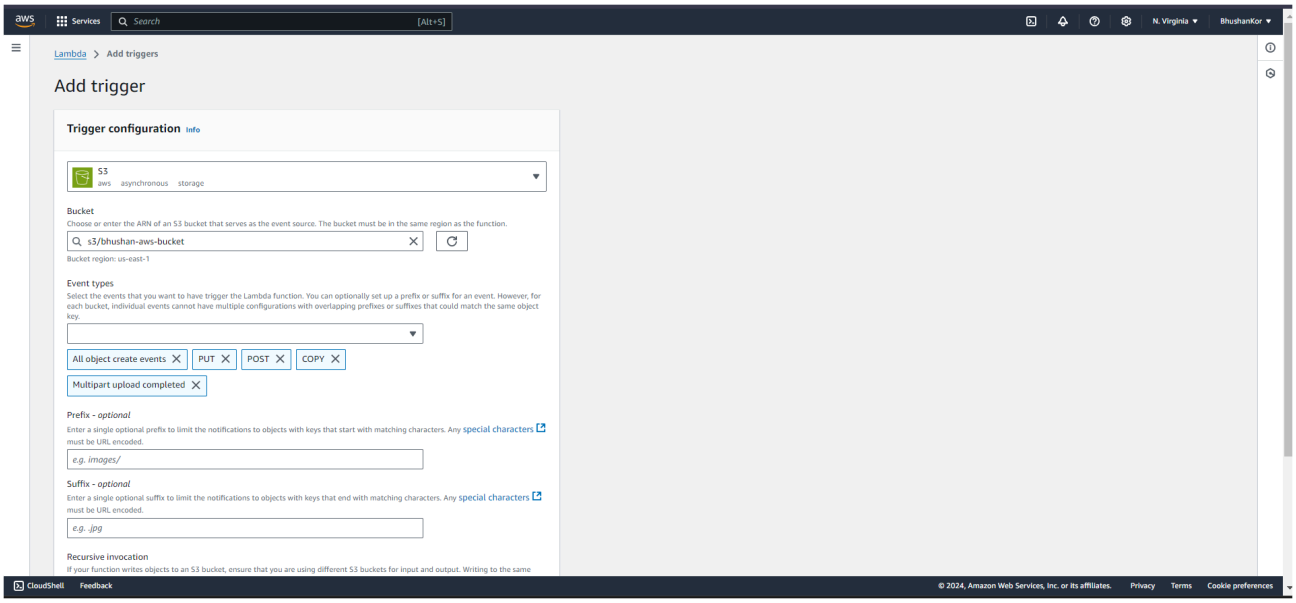


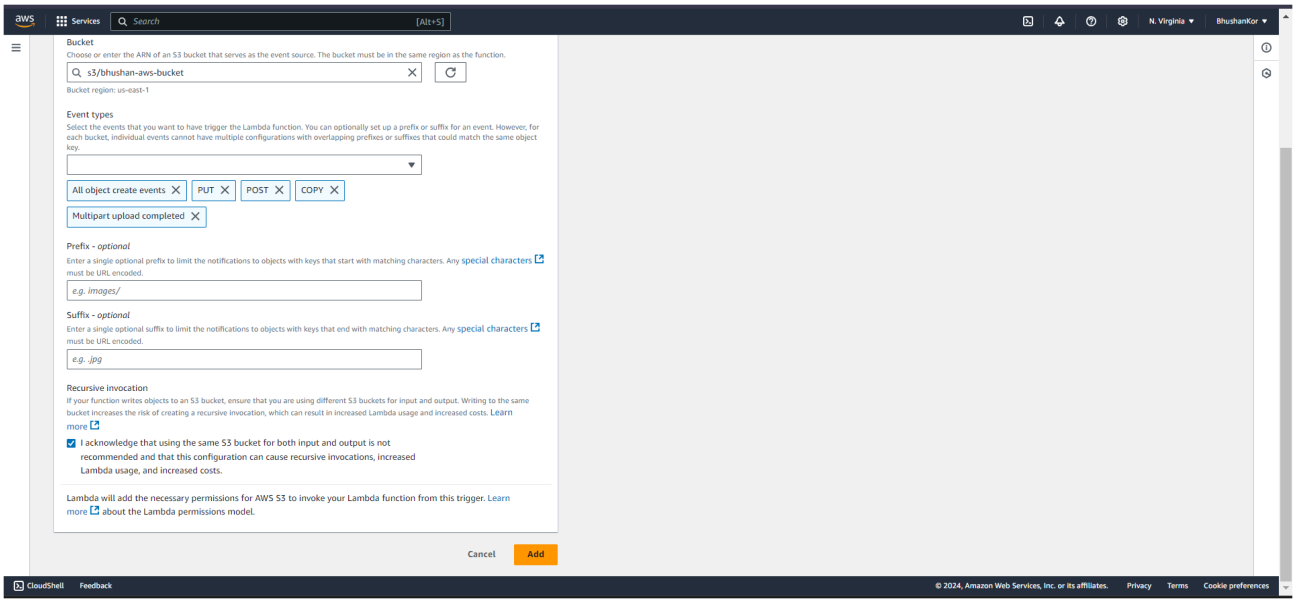


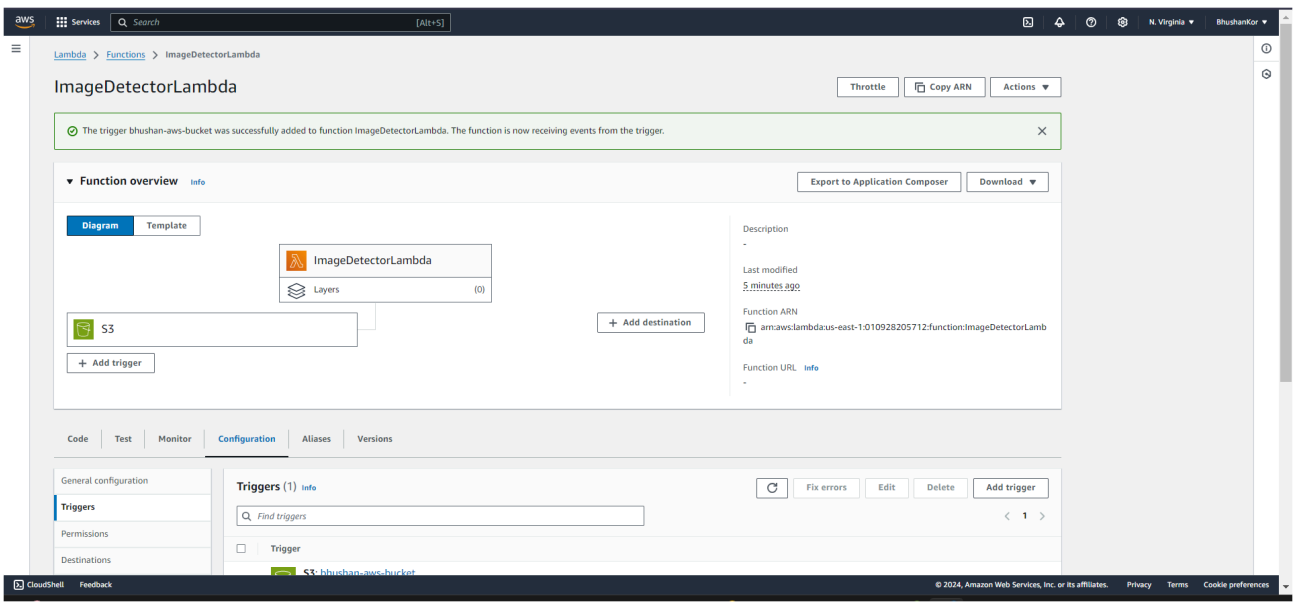


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**



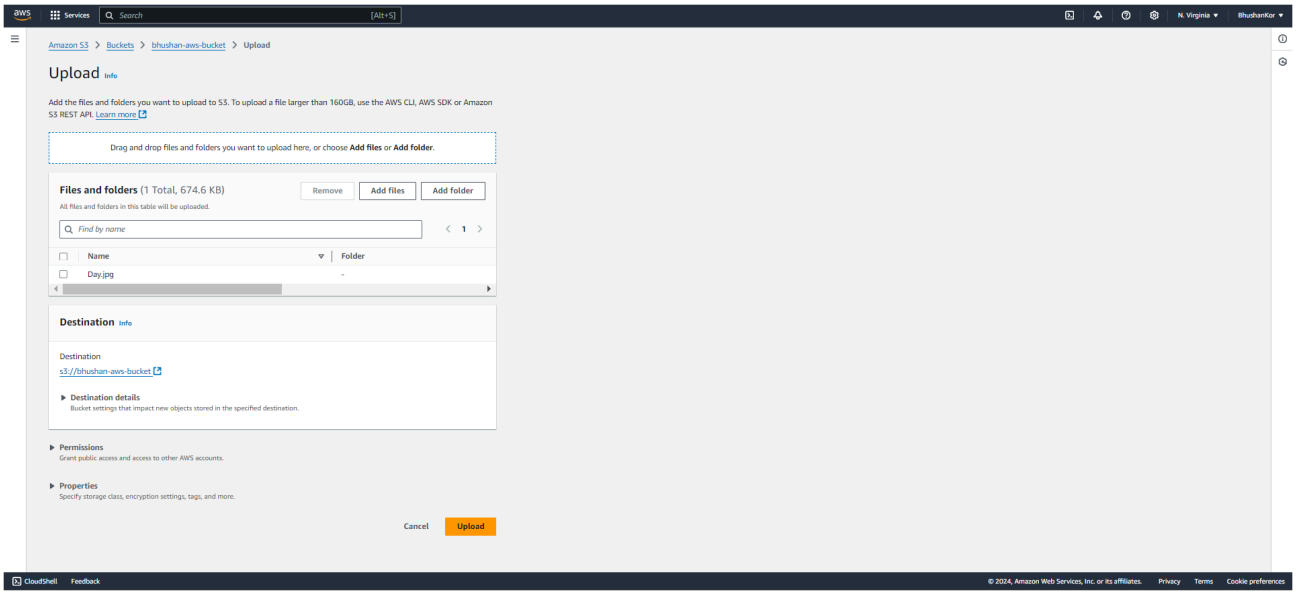


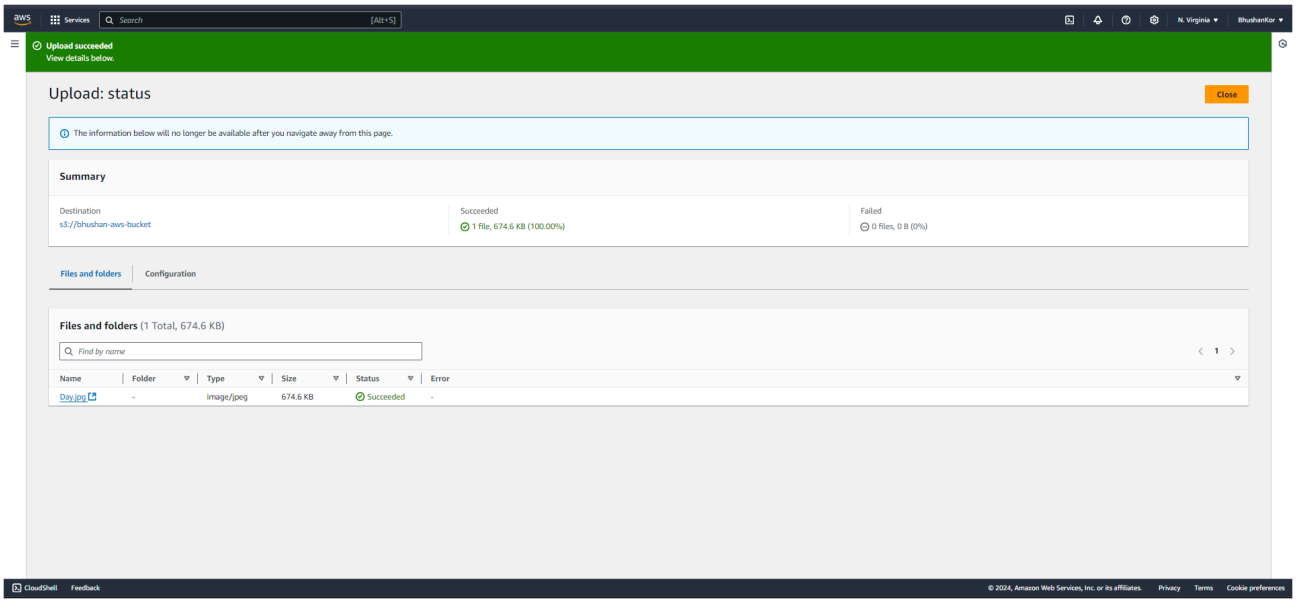


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

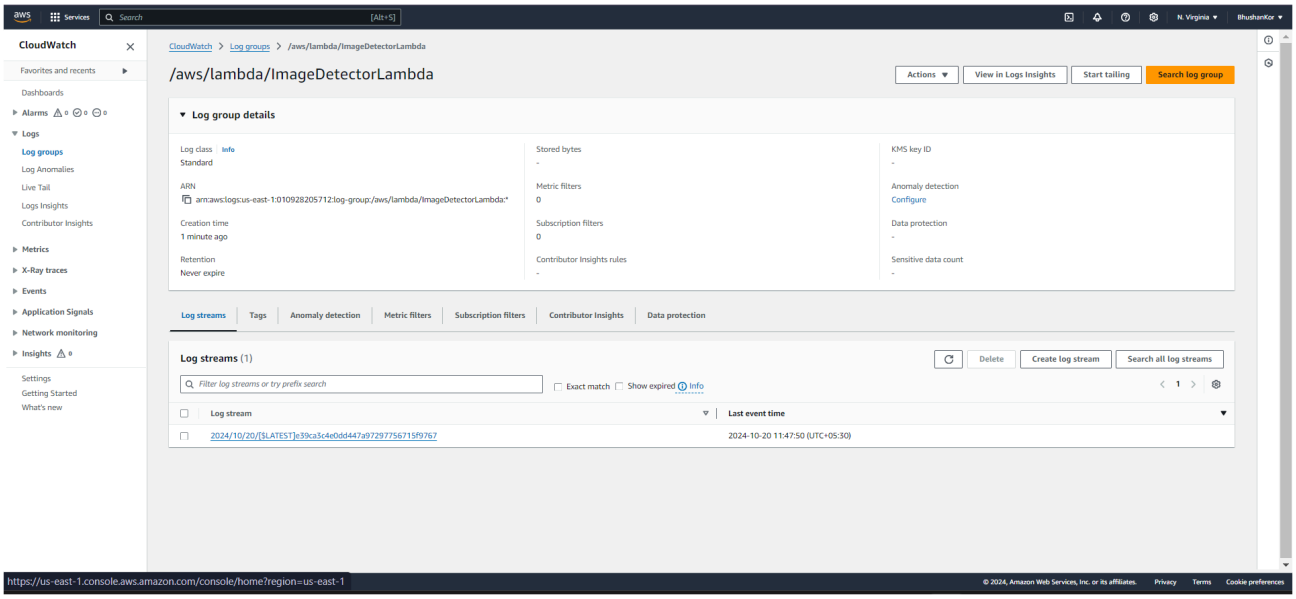
- Now Let's test that Lambda is properly working by uploading image and checking logs.

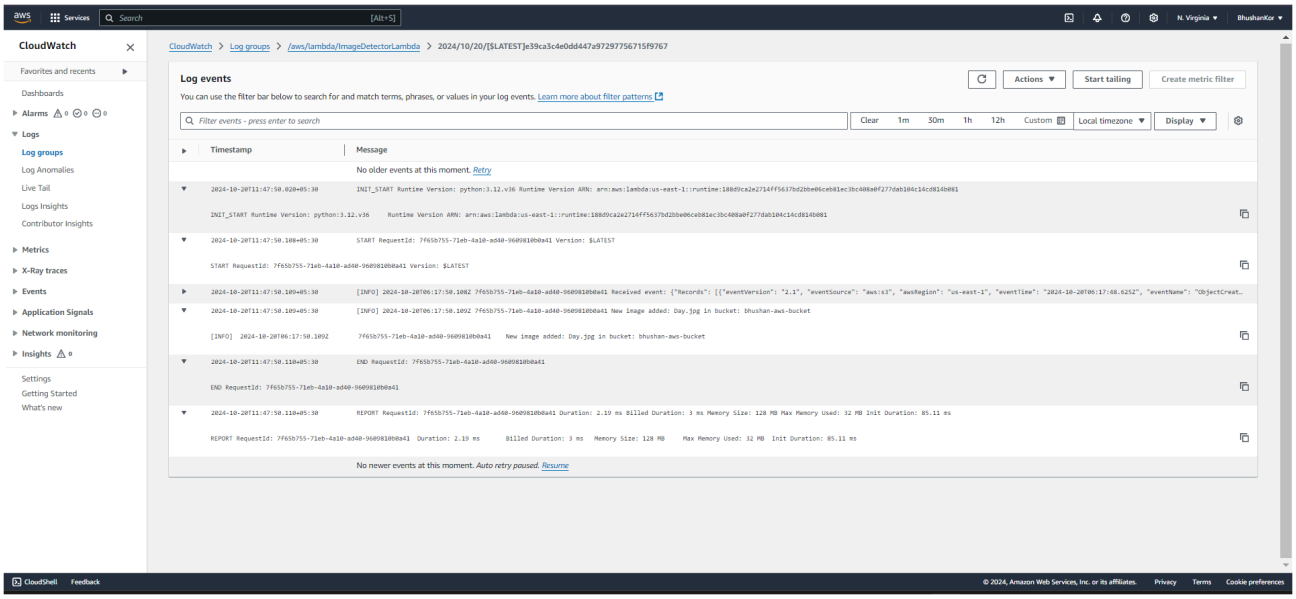




Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**





Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

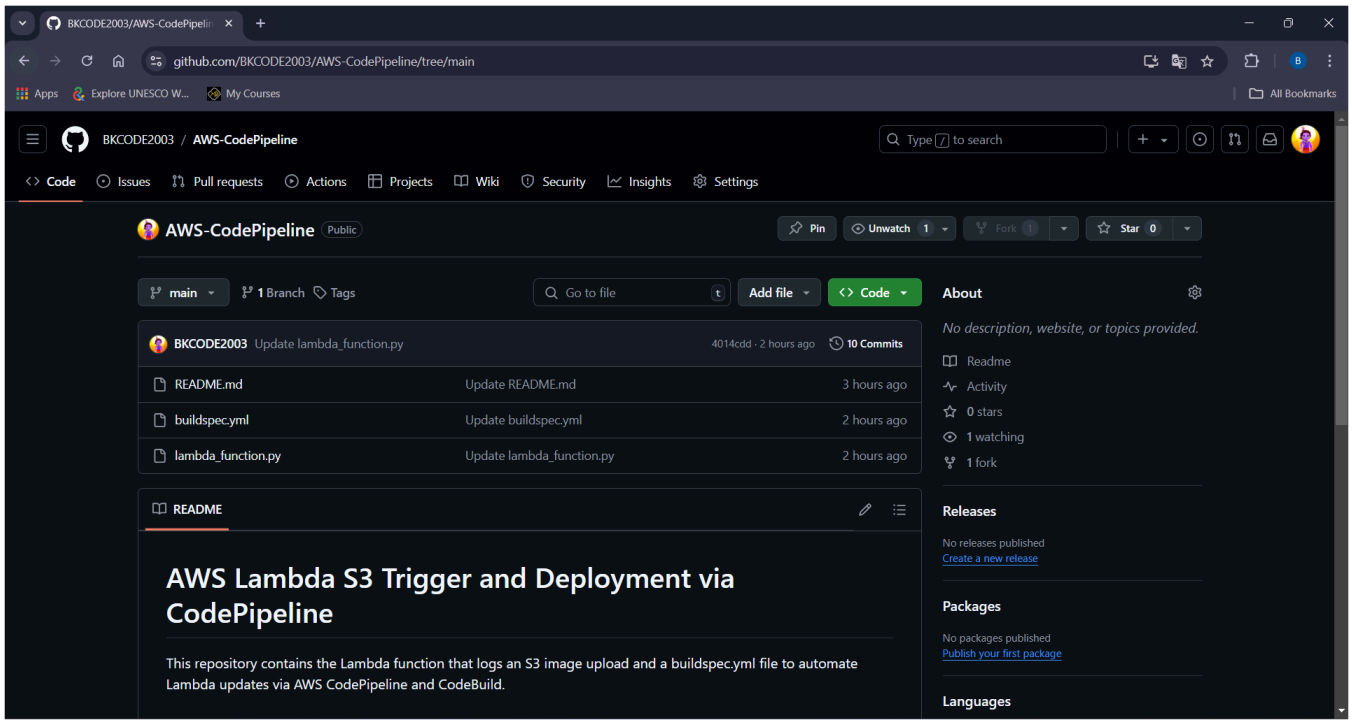
4. Create a Github Repository:

- Create Repositorywith name **AWS-CodePipeline.**
- Add the file **buildspec.yml** and **lambda_function.py**.

buidspec.yml

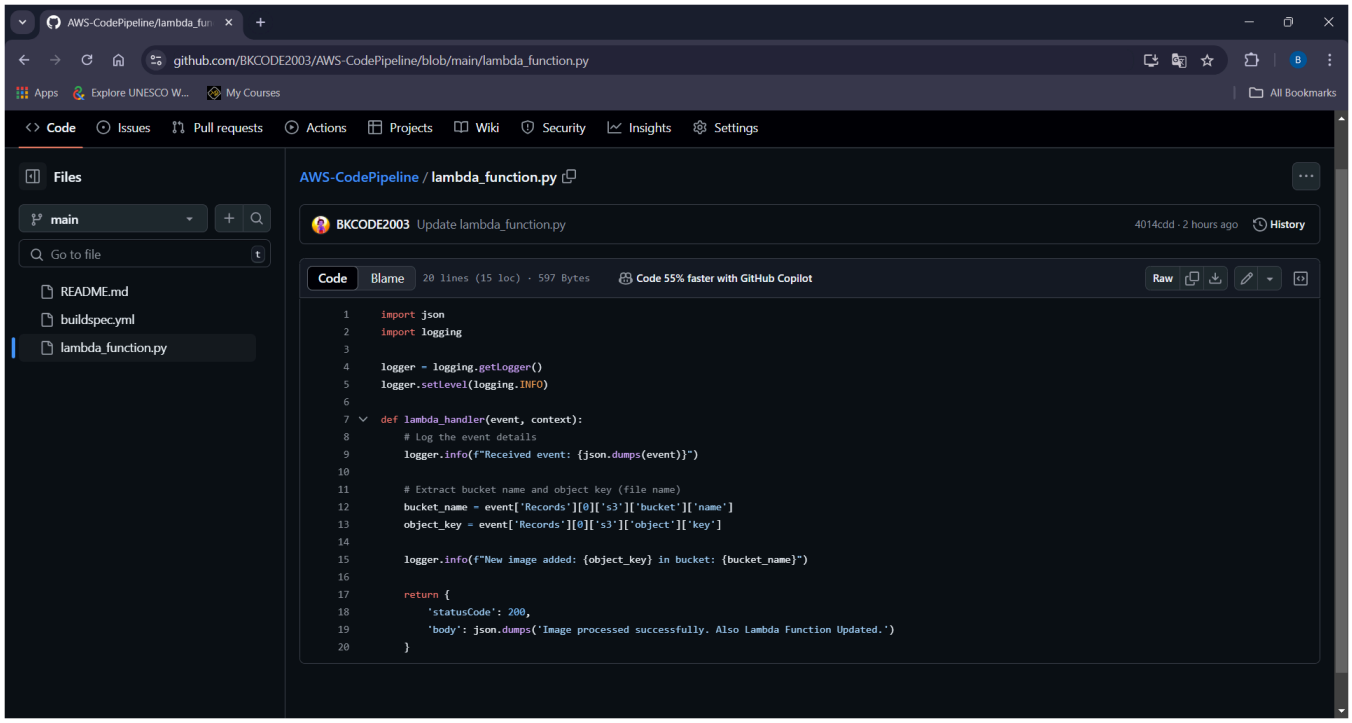
```
version: 0.2

phases:

  install:

    commands:

      - pip install awscli  # Ensure AWS CLI is available

  build:

    commands:

      - echo "Packaging Lambda function..."

      - zip -r lambda_function.zip .

      - echo "Updating Lambda function in AWS..."

      - aws lambda update-function-code --function-name ImageDetectorLambda --zip-file fileb://lambda_function.zip
```

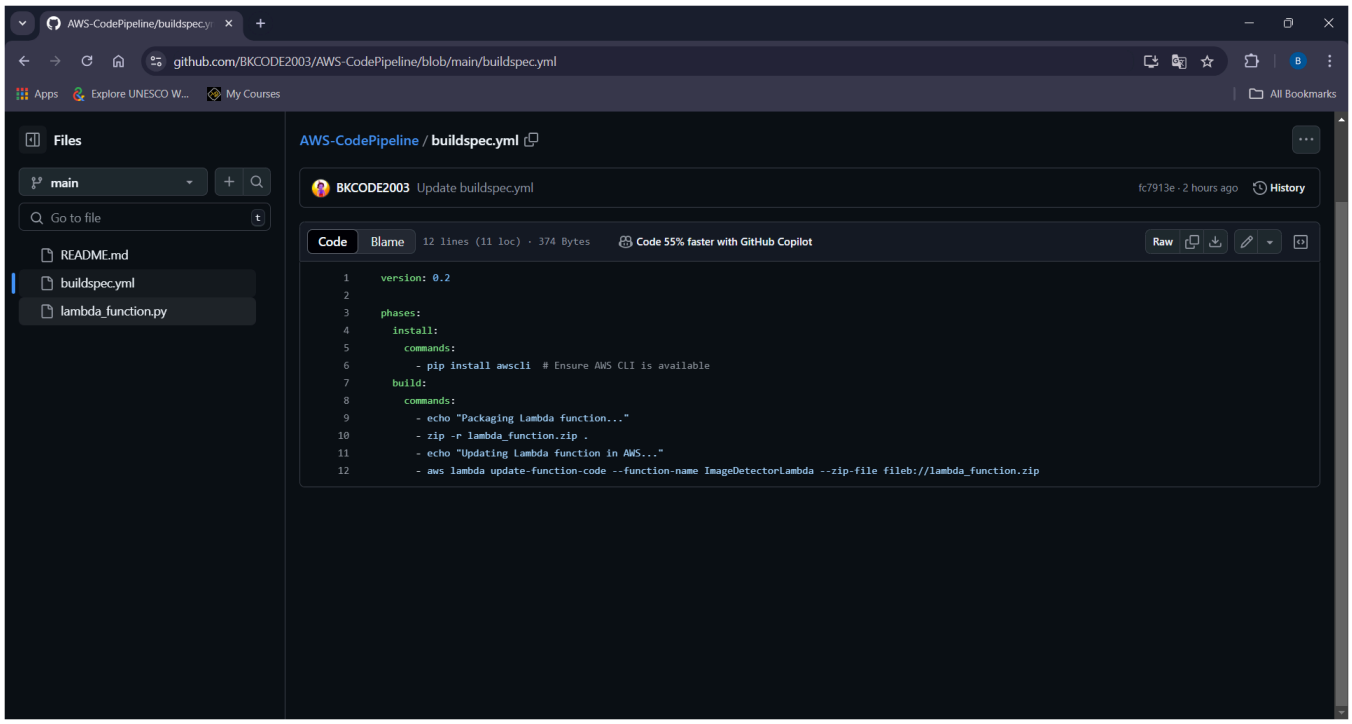
lambda_function.py Same as in step 3



Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**





Name:Bhushan Mukund Kor **Academic Year:2024-2025**

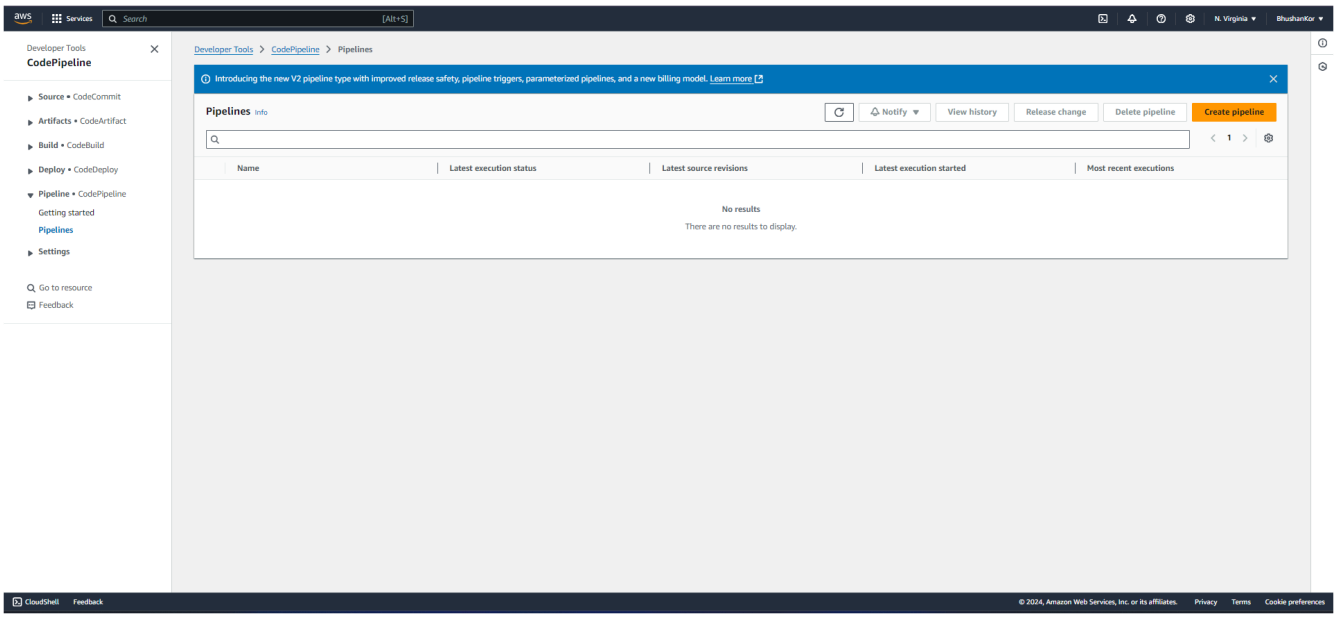
Division: D15C **Roll No: 28**

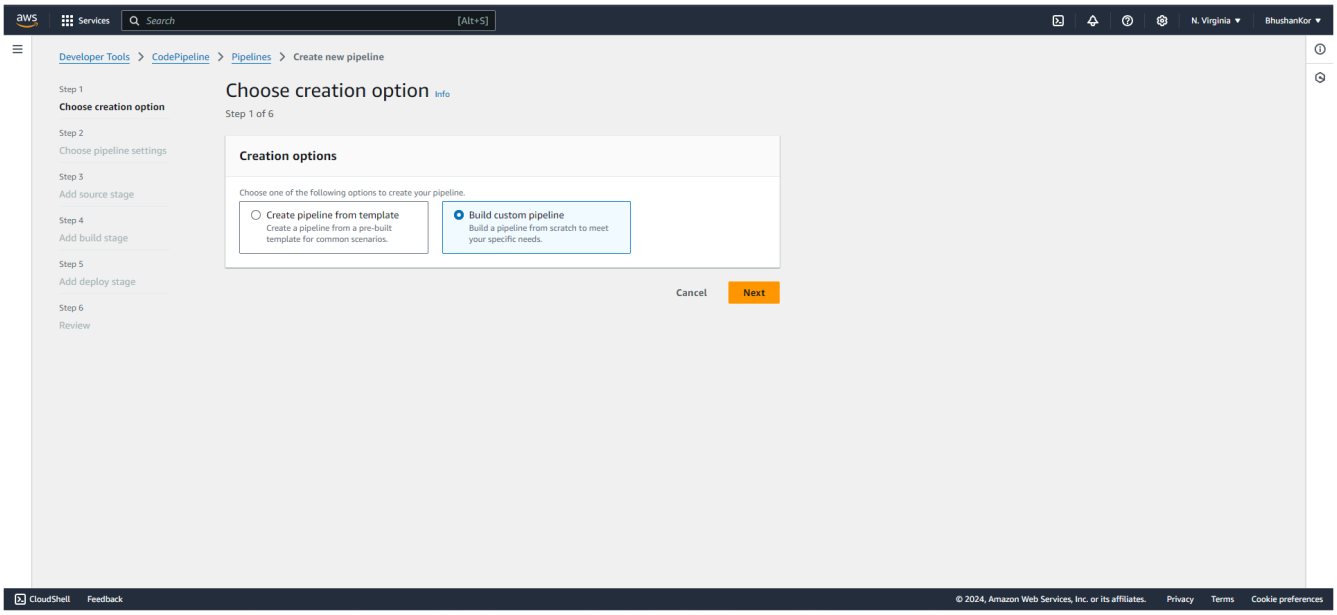
5. Create a CodePipeline:

- Go to **AWS CodePipeline** in the AWS Management Console and create a new pipeline.

Step 1:Choose Creation option.

- In creation option Select **Build Custom pipeline.**



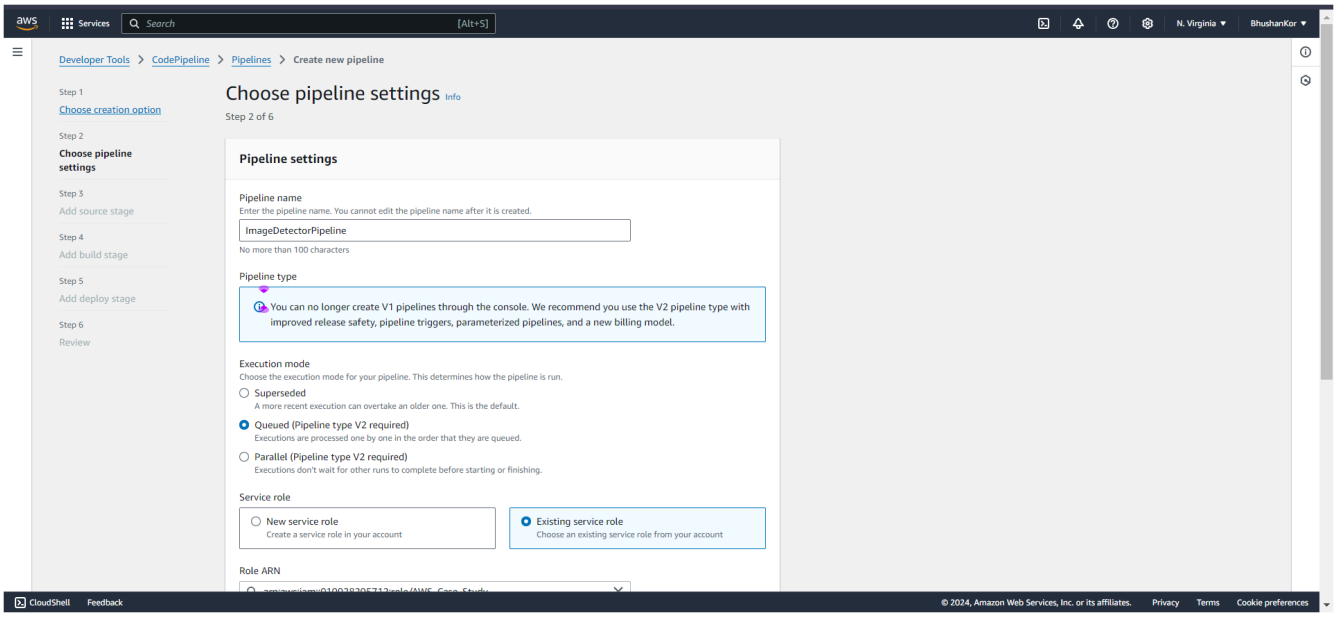


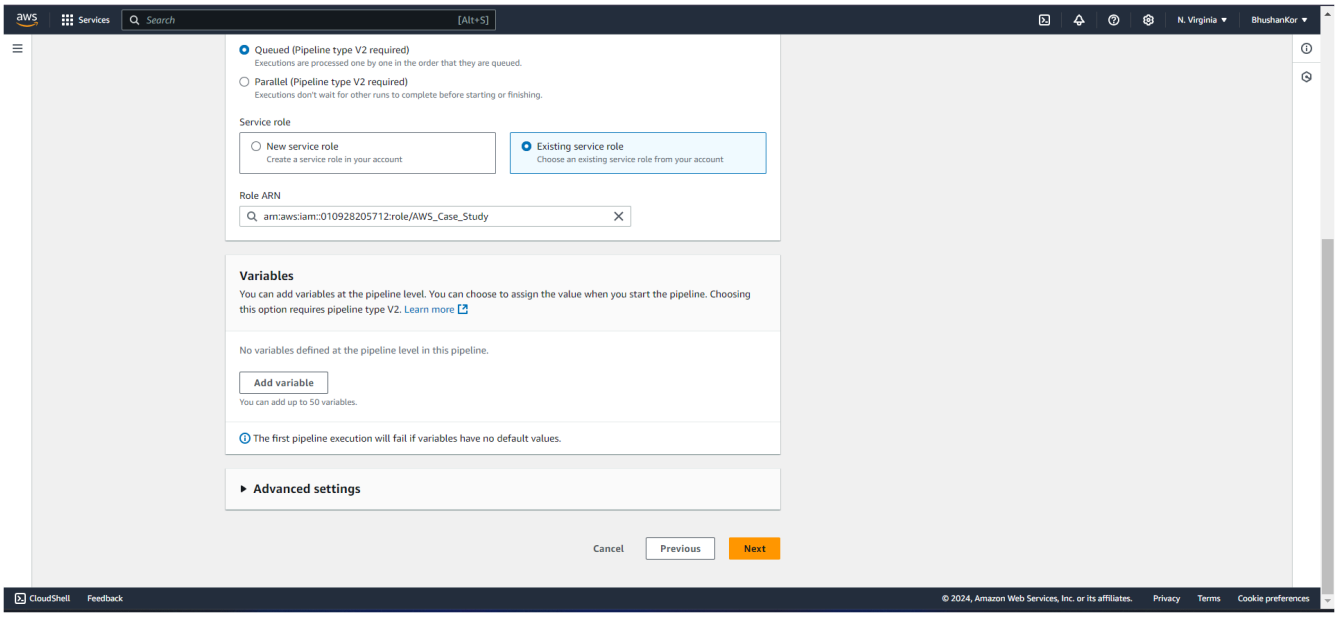
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Step 2:Choose Pipeline Settings.

- Pipeline Name: **ImageDetectorPipeline** .
- Select Existing Role : **AWS_Case_Study** .
- Keep Rest all to default.



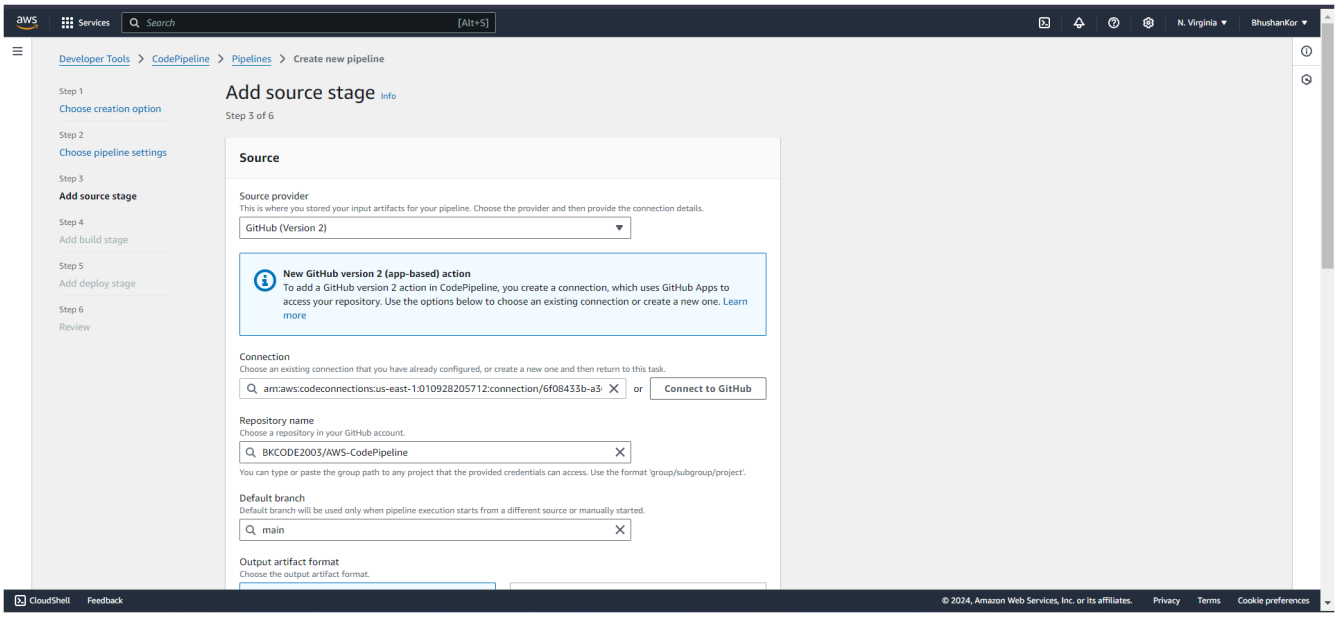


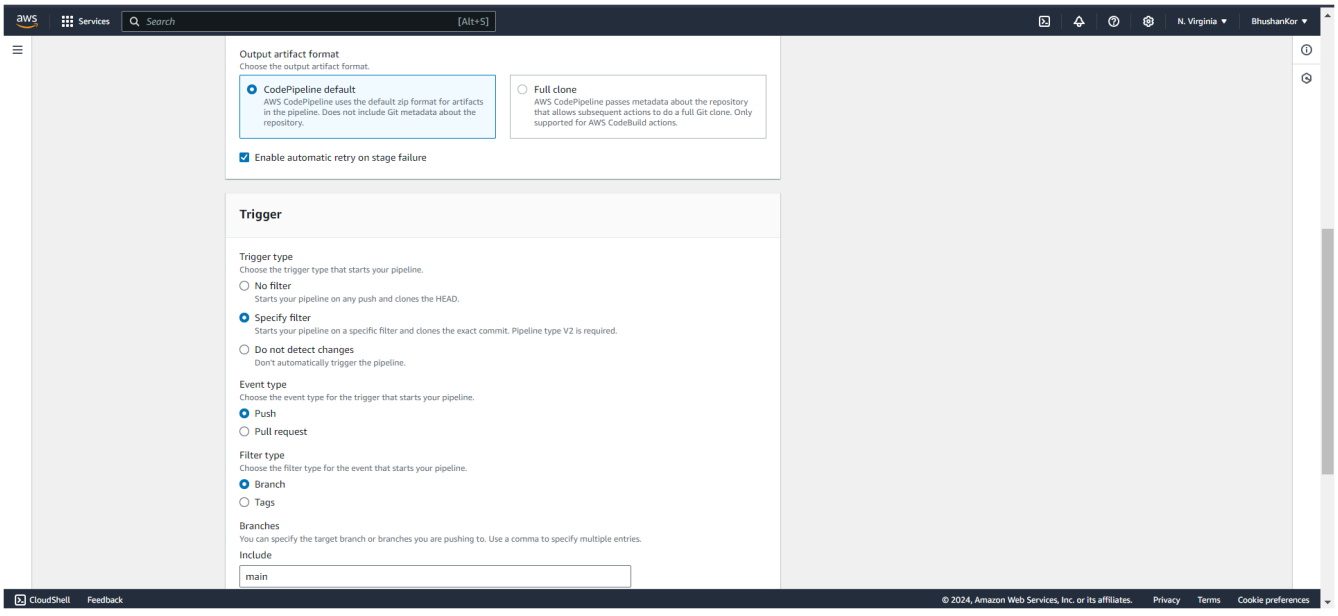
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Step 3: Add Source Storage.

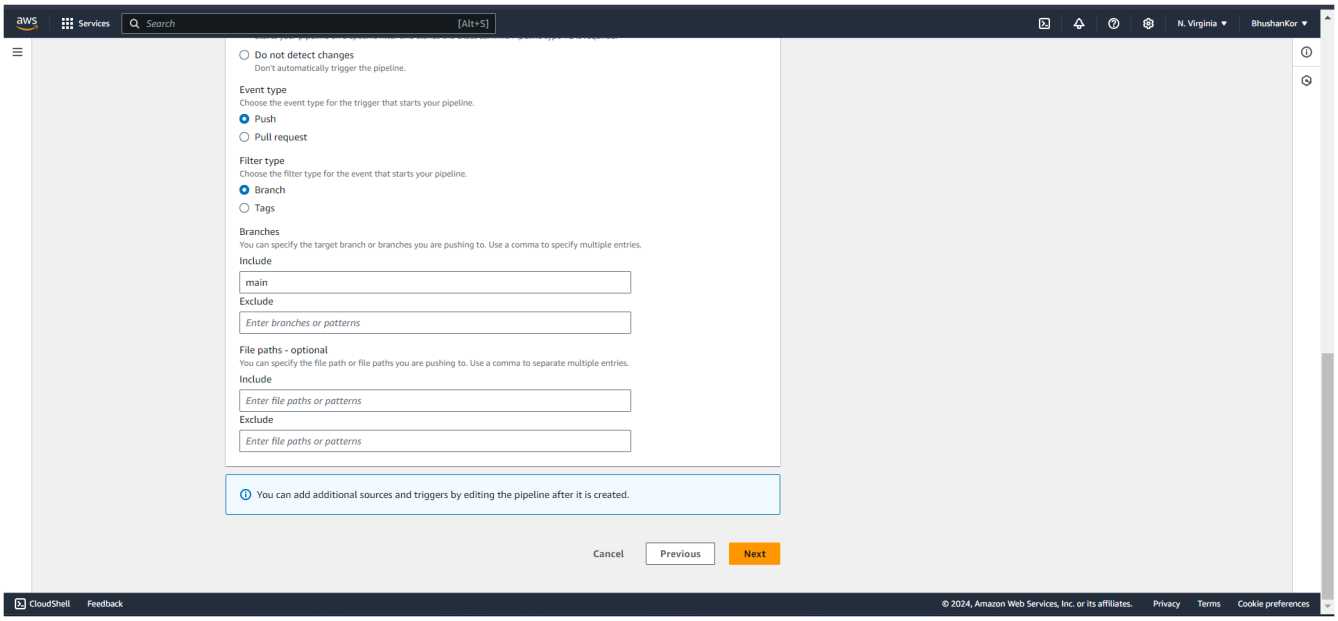
- Source provider: **GitHub (Version 2)** .
- Connect Your account where you have created the repository. **(BKCODE2003)**
- Select Repository. **(BKCODE2003/AWS-CodePipeline)**
- Select Branch to **main**.
- In Trigger Just add **main** in include.
- Keep Rest all to default.





Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

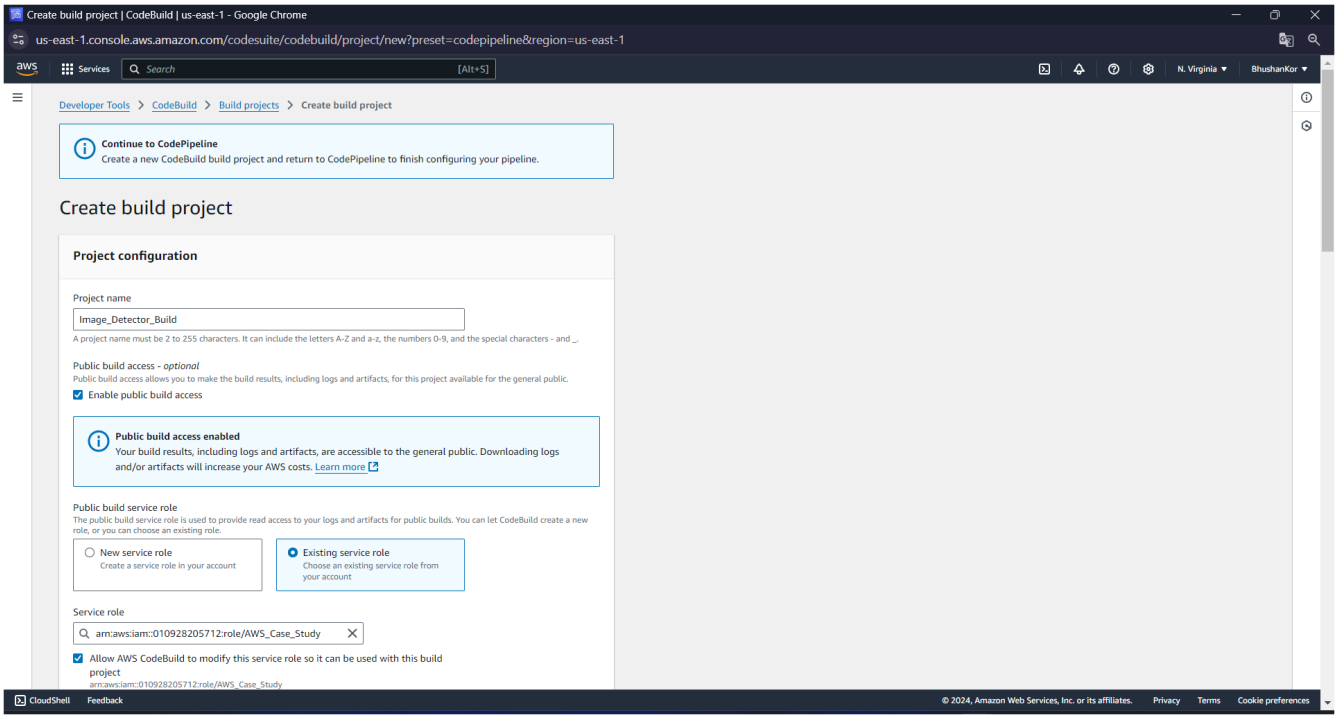


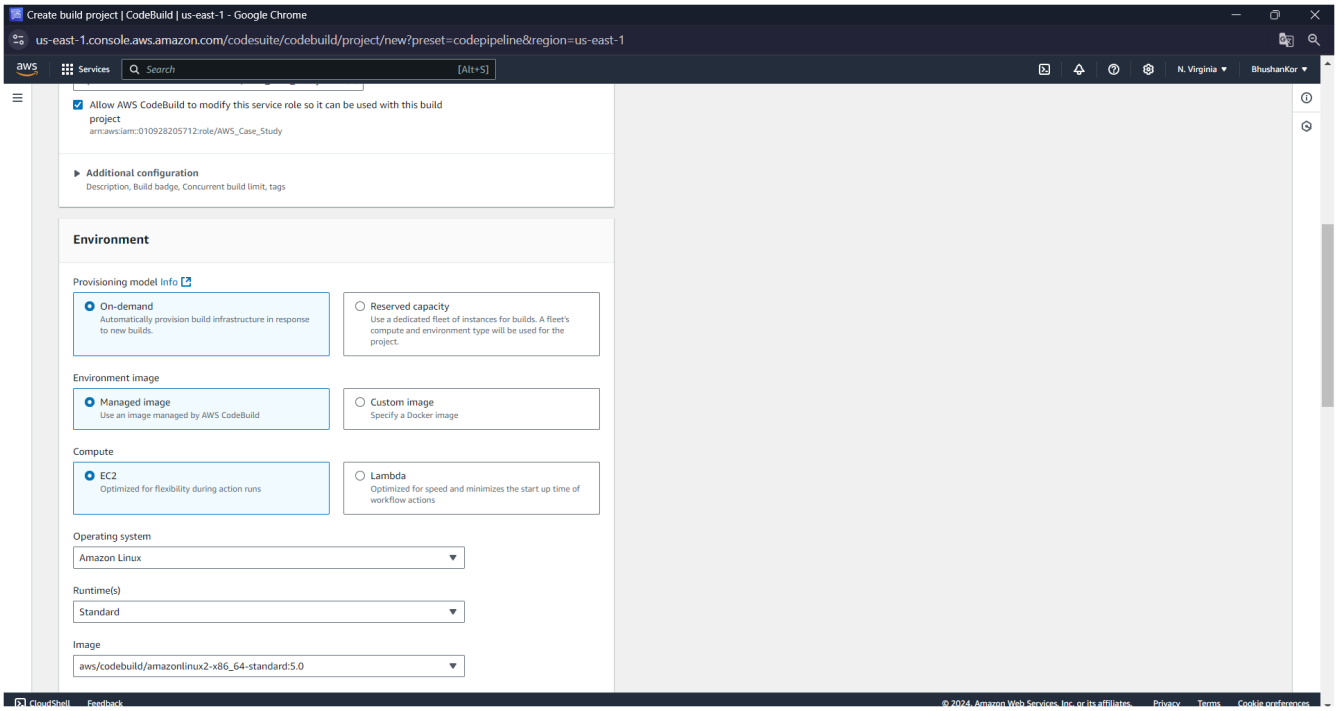
Step 4: Add Build Stage.

- Select **Other build Provider**.
- Select **AWS CodeBuild**.
    - Now Click on **Create Project**.
    - Project name: **Image_Detector_Build**
    - Enable public access.
    - Select Existing Role : **AWS_Case_Study** On 2 Places.
    - Buildspec: **Use a buildespec file** .
    - Buildspec name: **buildspec.yml** .
    - Keep Rest all to Default.
- Select Created Project.
- Keep Rest all to default.

Name:Bhushan Mukund Kor **Academic Year:2024-2025**

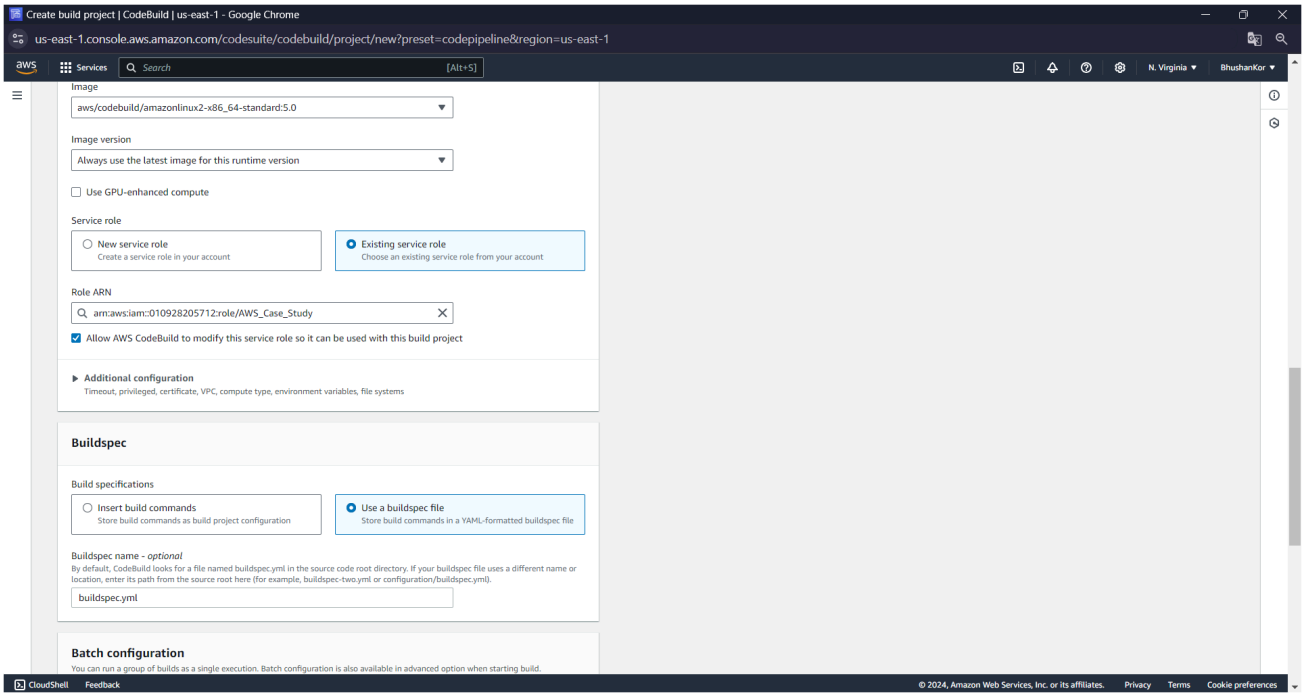
Division: D15C **Roll No: 28**

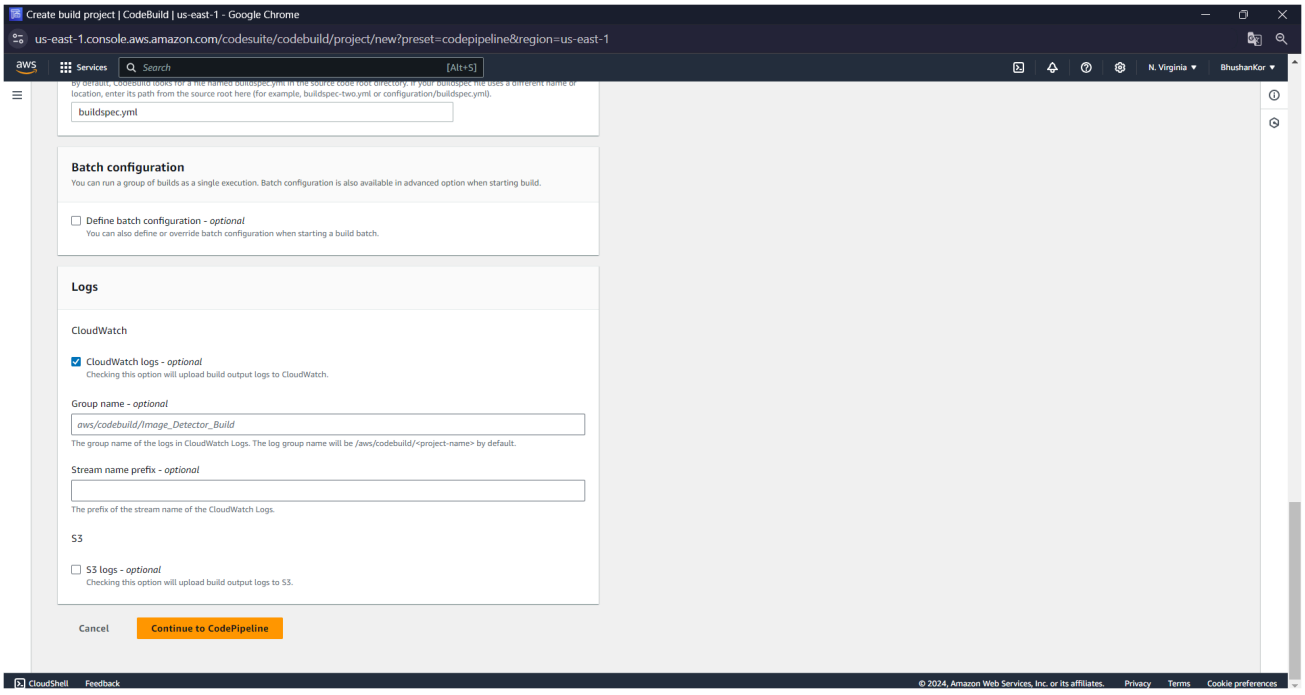




Name:Bhushan Mukund Kor **Academic Year:2024-2025**

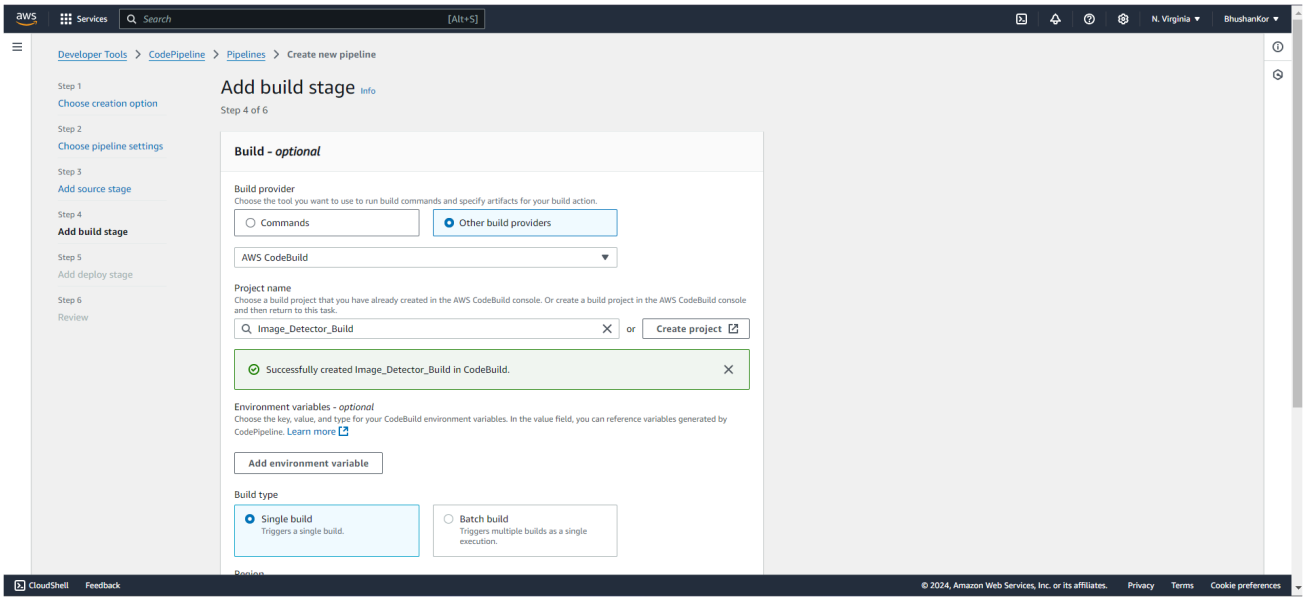
Division: D15C **Roll No: 28**

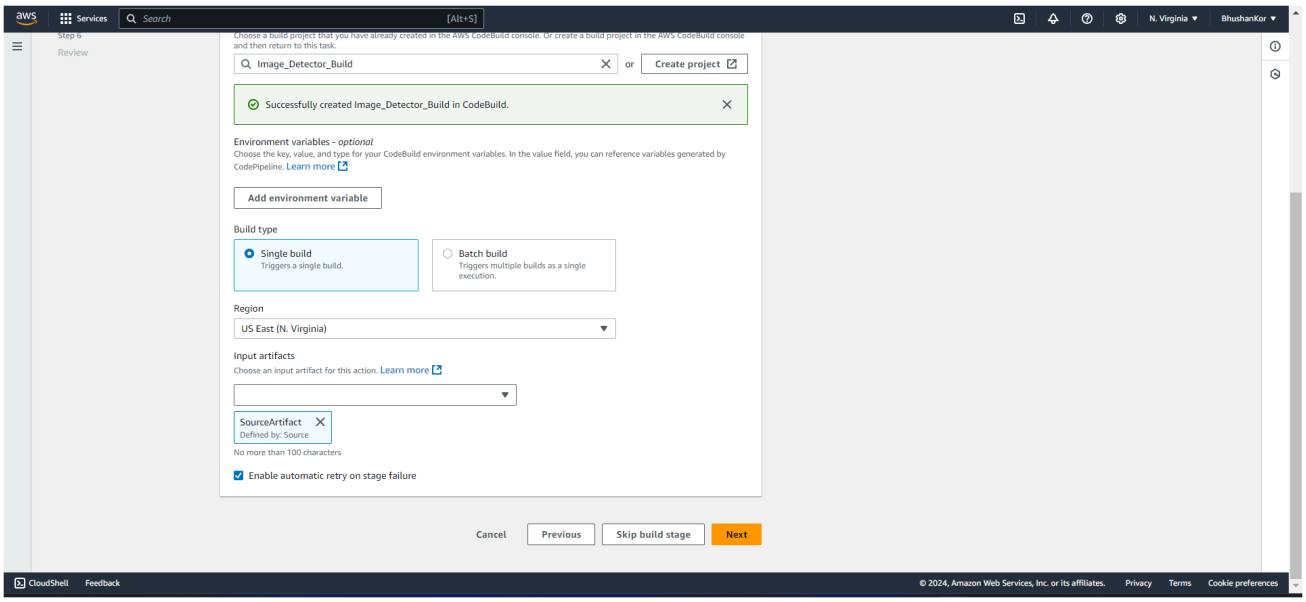




Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**



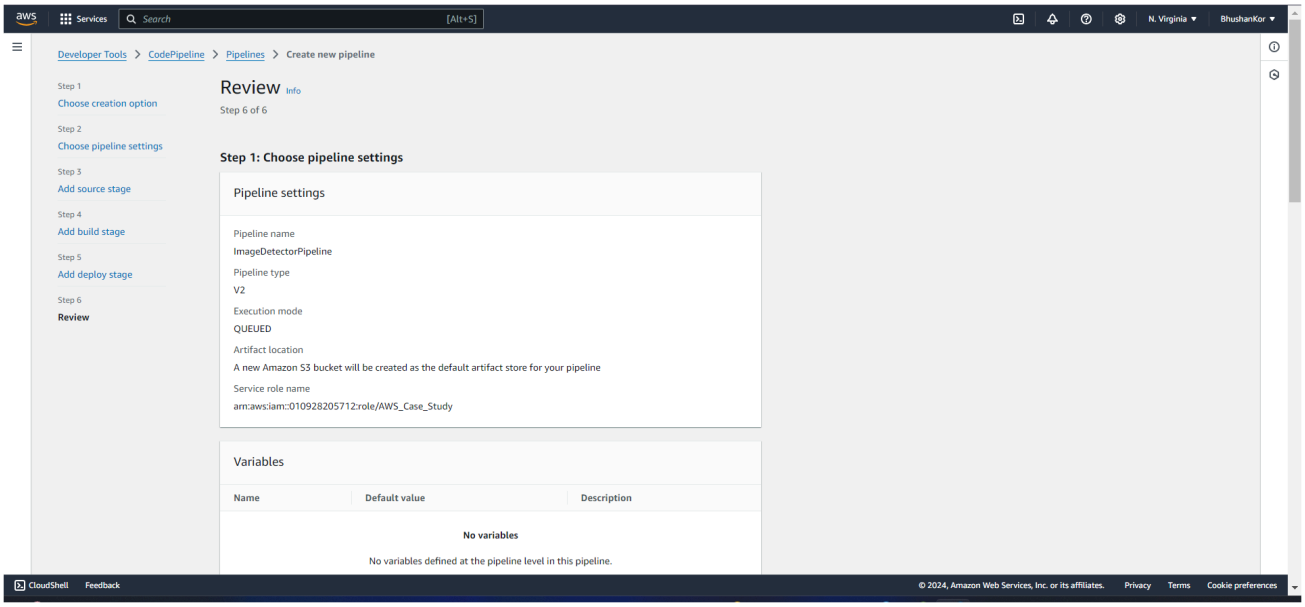


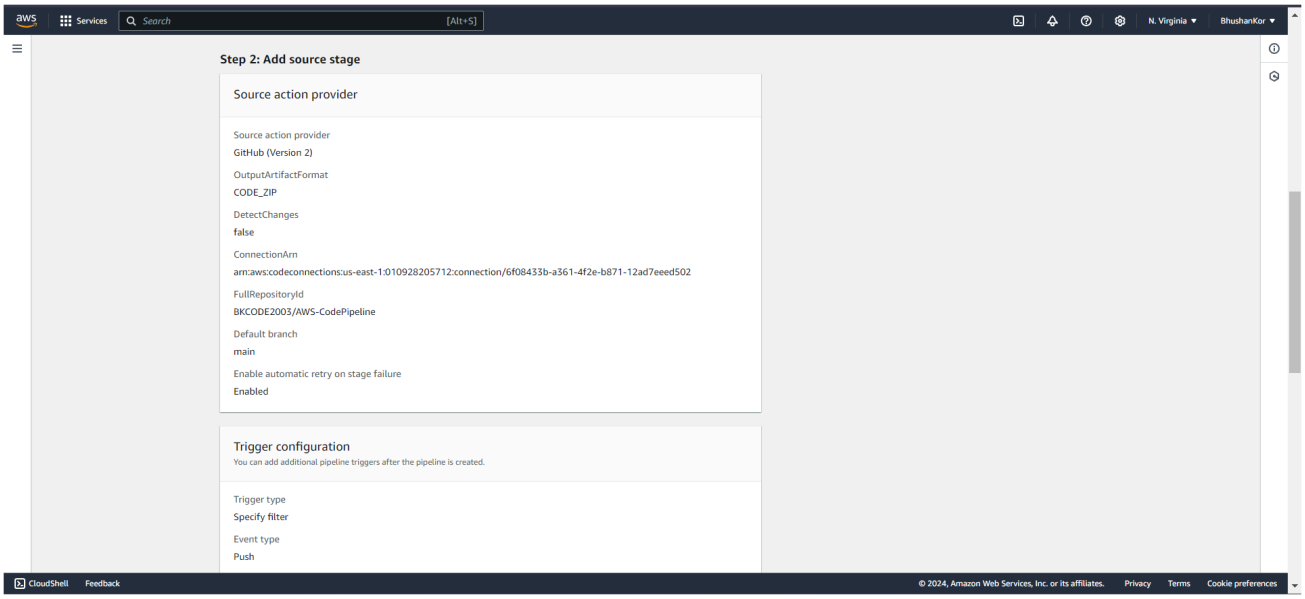
Step 5: Skip The Deploy Stage.

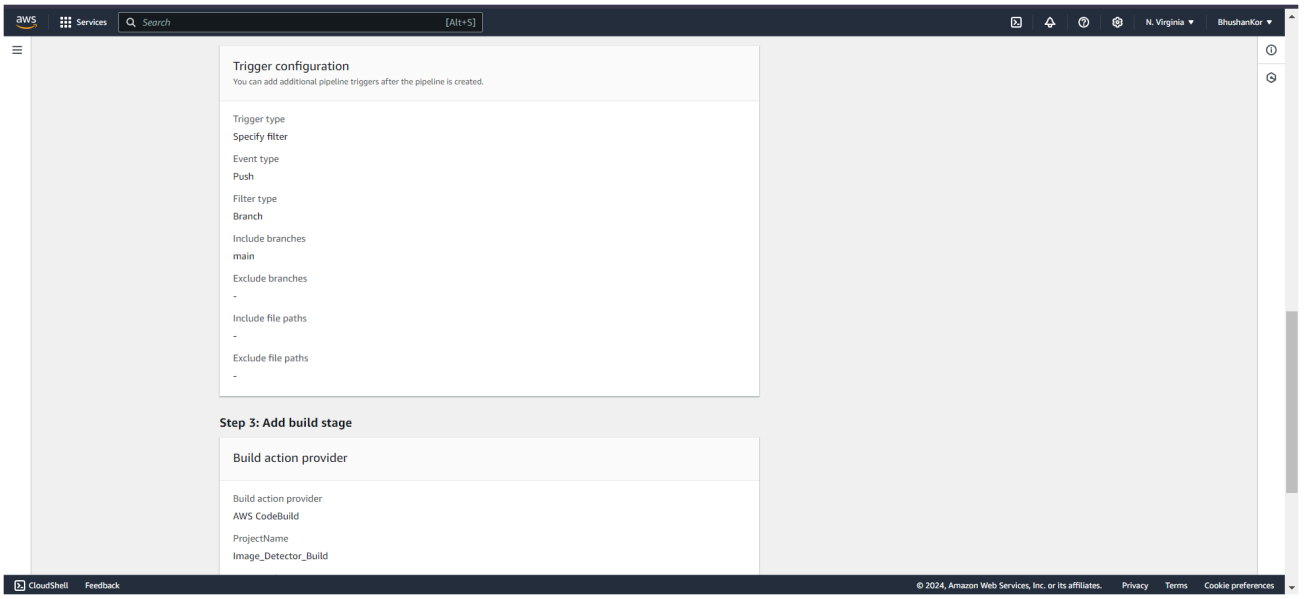
Step 6: Review the Pipeline and Click on Create.

Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

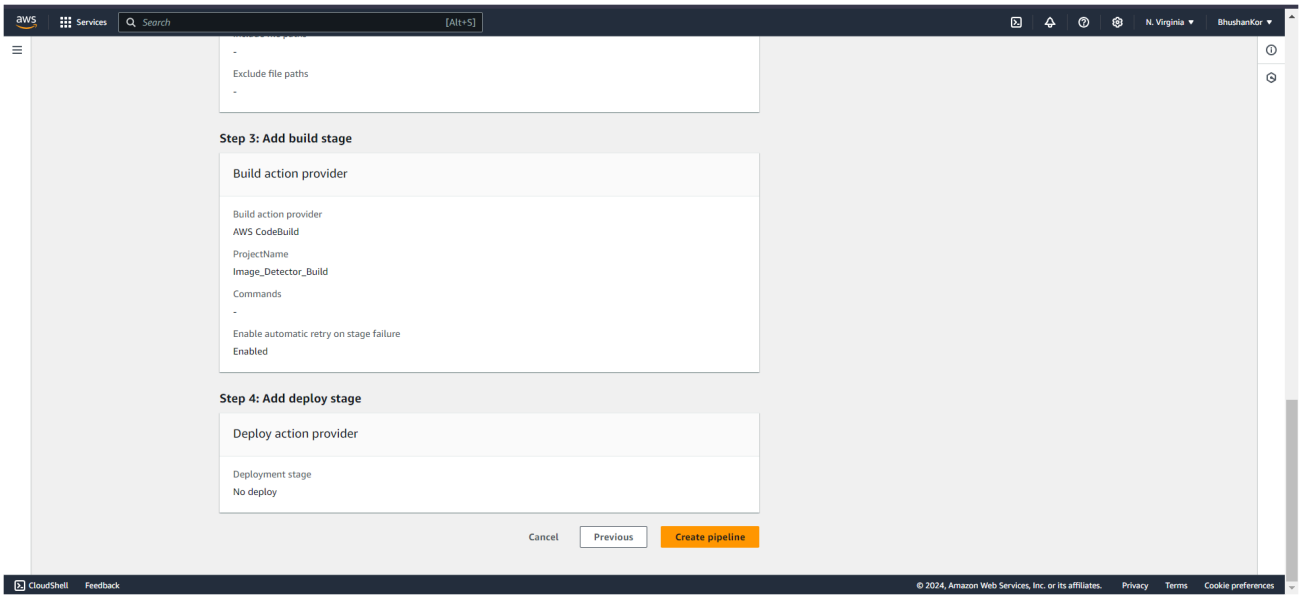


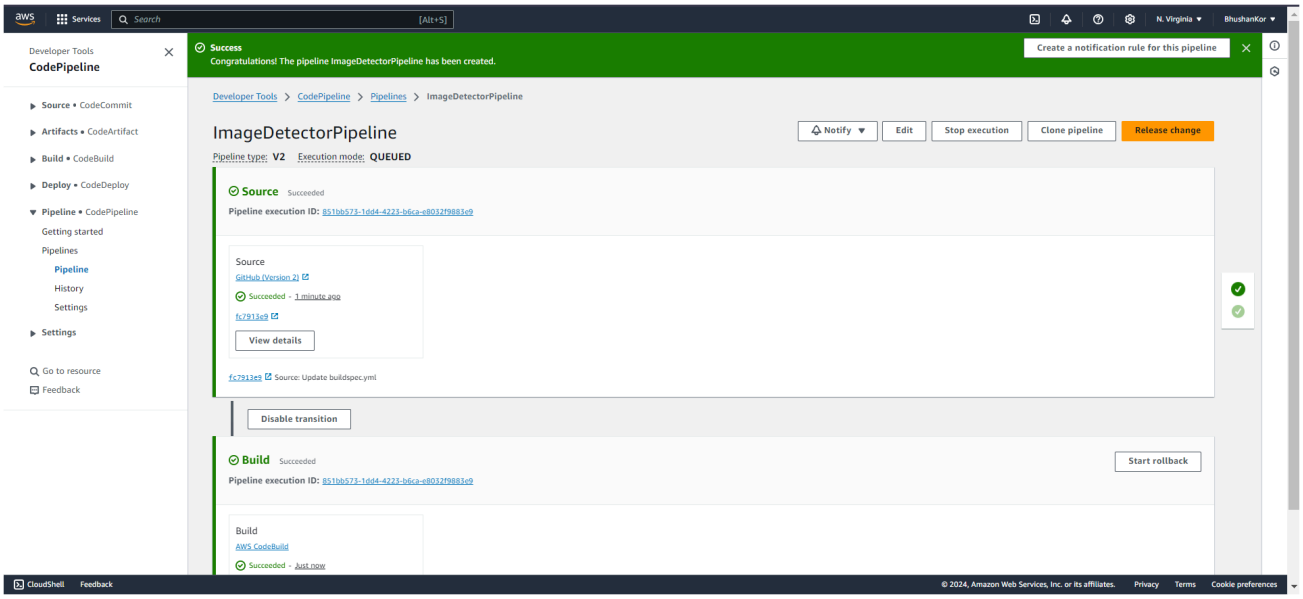


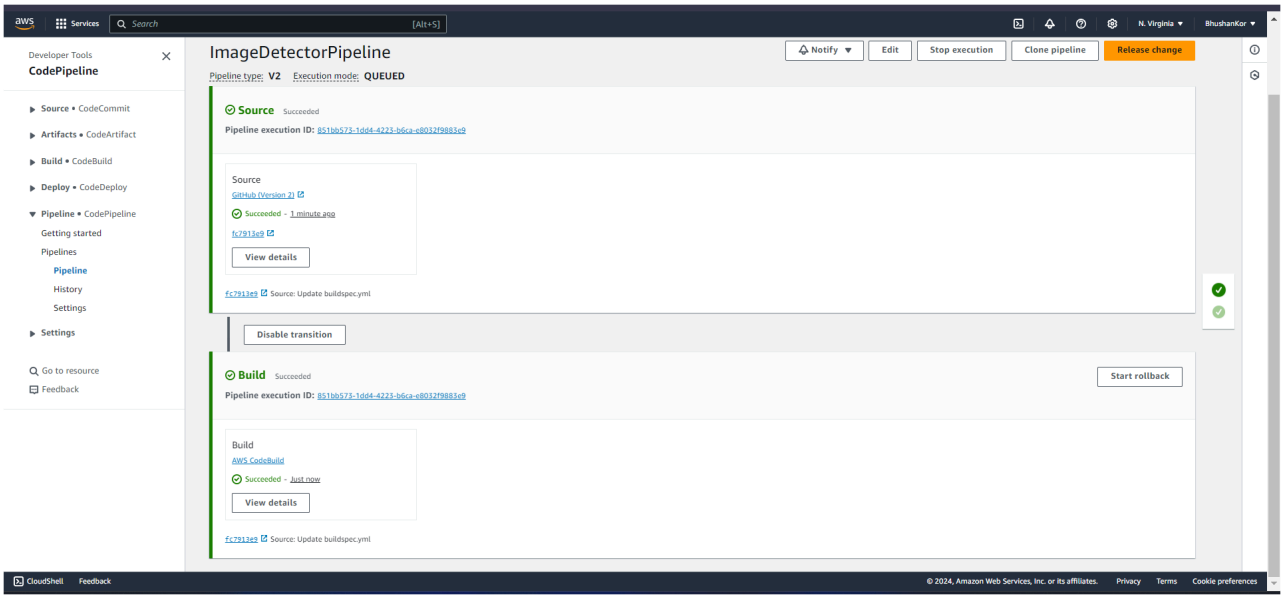


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**







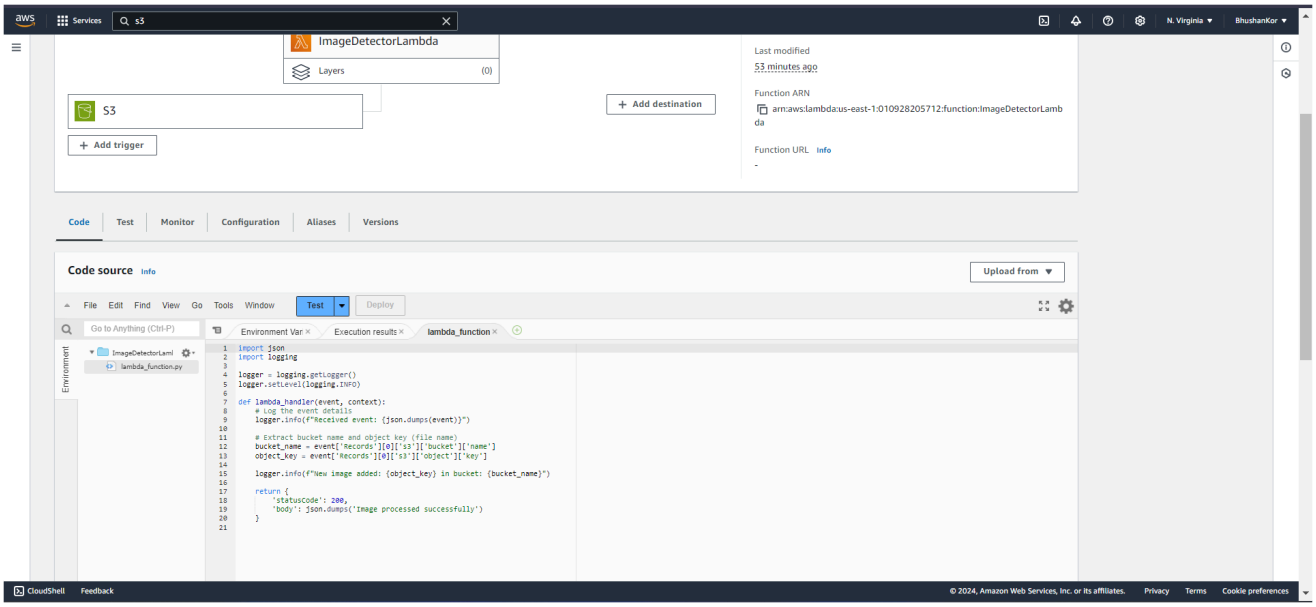
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

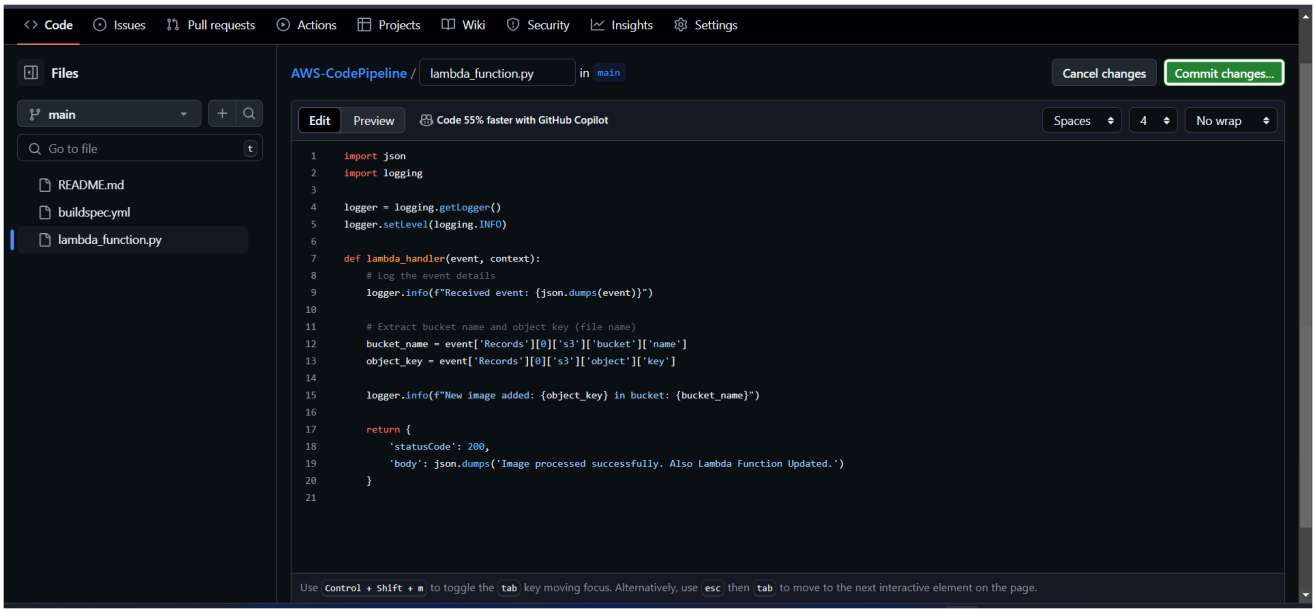
6.Test the Pipeline:

- Push/Update code to your GitHub repository.
- CodePipeline will trigger the CodeBuild project, which will package the code and update the Lambda function using the AWS CLI command aws lambda update-function-code.

Lambda Before push of new code in repository



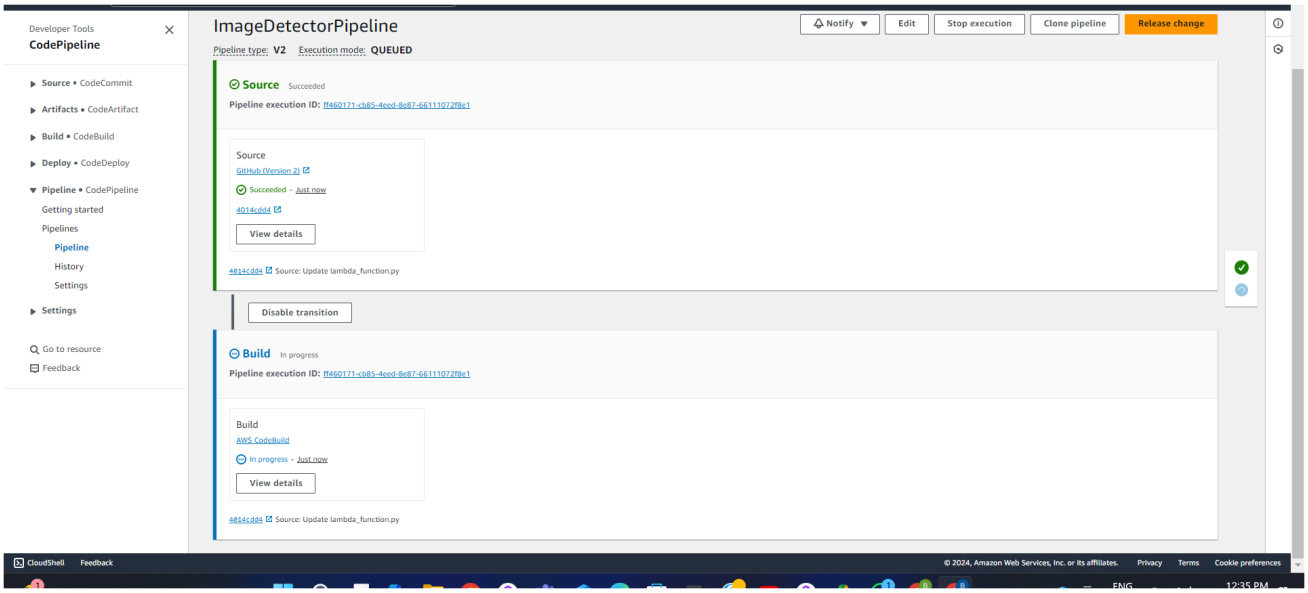
Code Change in Repository



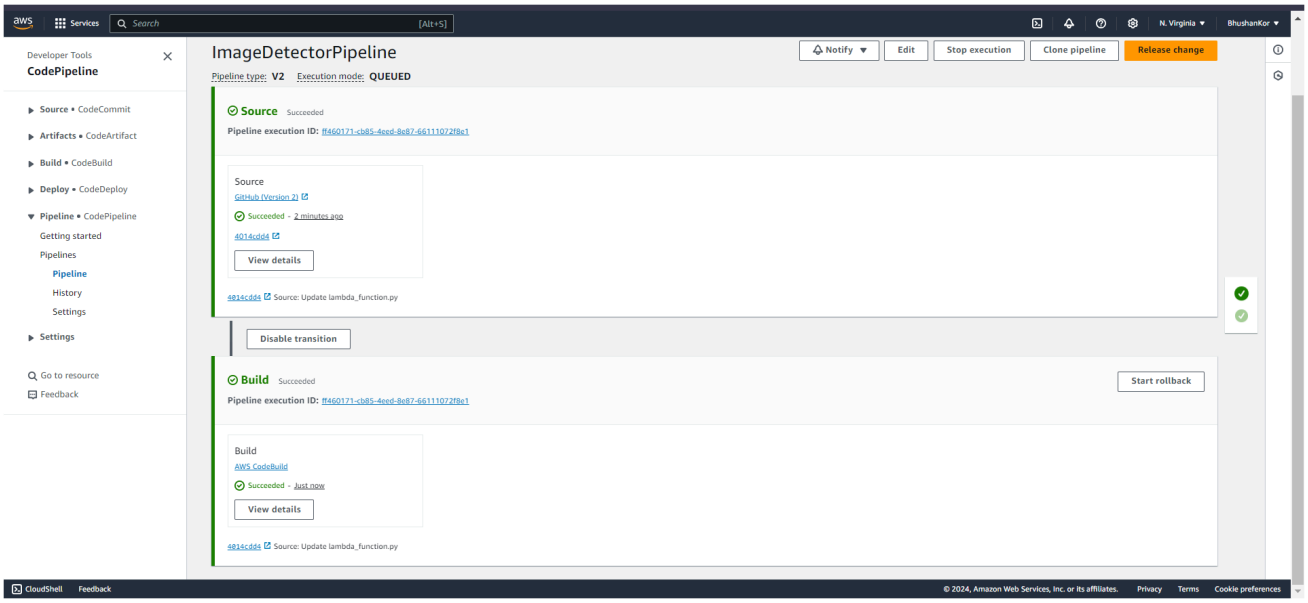
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Auto Triggered Pipeline



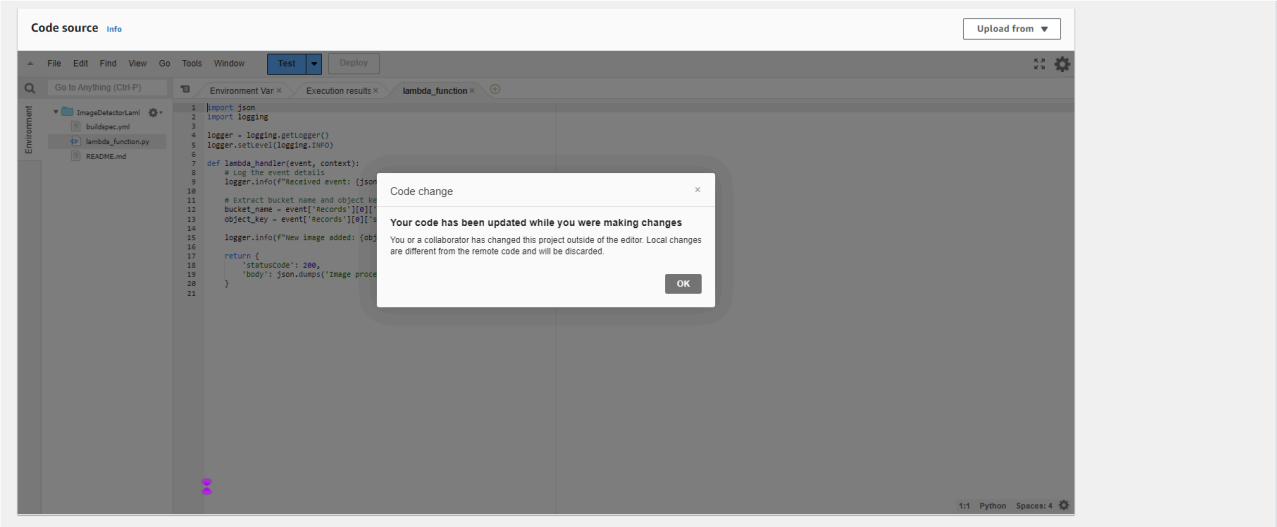
Successful Build.

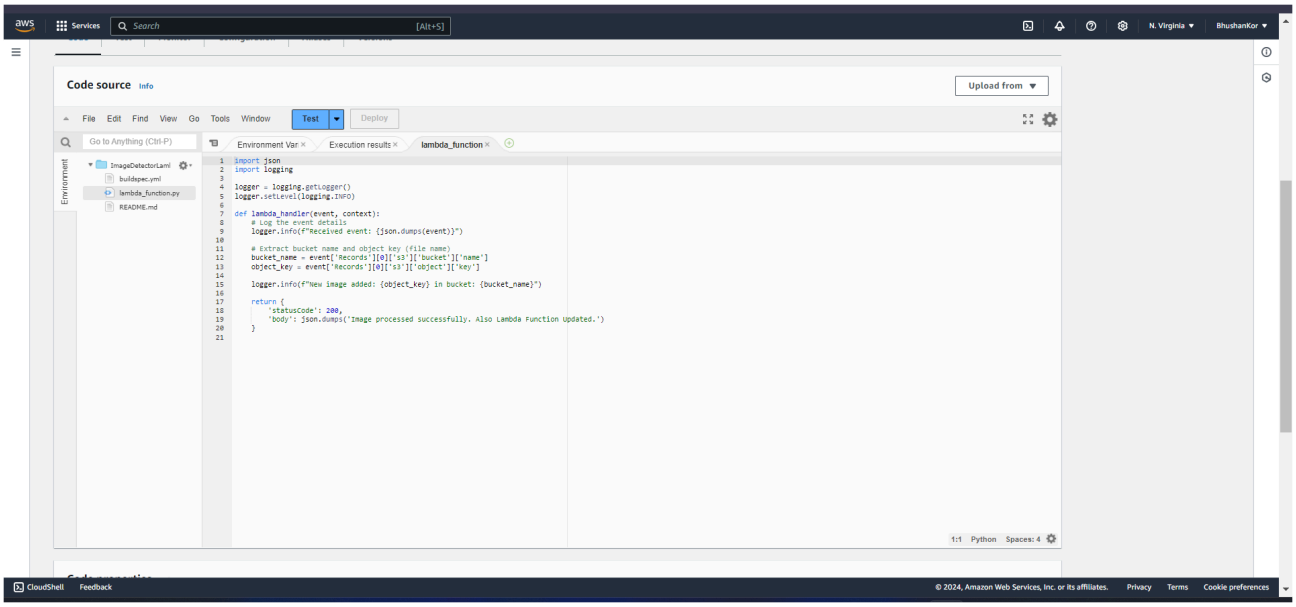


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Lambda After push of new code in repository .





Guidelines:

- **Use AWS Personal.**
- **IAM Role Creation**:
    - **Why It's Important**: The IAM role is critical as it grants the necessary permissions for Lambda, CodeBuild, and CodePipeline to interact with each other and with other AWS services like S3. Without proper permissions, the workflow will not function, and components will fail to trigger or deploy.
    - **Guidelines**:

- Attach policies such as **`AWSLambda_FullAccess`**, **`AmazonS3FullAccess`**, **`AmazonCodeBuildAdminAccess`**, **`AmazonCodePipeline_FullAccess`**, and **`CloudWatchLogsFullAccess`** to allow access to Lambda, S3, CodeBuild, and CodePipeline.
- Ensure that the trust relationship includes both **`codepipeline.amazonaws.com`** and **`codebuild.amazonaws.com`** services to allow these services to assume the role.

Conclusion:

In this case study, we successfully implemented a serverless image processing workflow using AWS Lambda, S3, and CodePipeline. The solution demonstrated how Lambda functions can be triggered by S3 events, allowing real-time image processing without manual intervention. By leveraging CodePipeline, we automated the deployment of updates to the Lambda function, ensuring a streamlined CI/CD process. Testing confirmed that the workflow functions as expected, with images uploaded to S3 triggering the Lambda function, which logs the event and processes the image.