



Vivekanand Education Society's

Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai,, Approved by AICTE & Recognized by Govt. of Maharashtra
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

A.Y. 2024-25

Advance DevOps Lab

Experiment 12

Aim: To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Roll No.	42
Name	Naikwadi Yash Shivdas
Class	D15B
Subject	Advance DevOps Lab
LO Mapped	LO1: To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements. LO6: To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework
Grade:	

AIM : To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

THEORY :

AWS Lambda and S3 Integration

Delving into the powerful integration between AWS Lambda and Amazon S3, a popular use case in serverless architecture. AWS Lambda enables automatic execution of code in response to events generated by other AWS services, such as the addition of an object to an S3 bucket.

Event-Driven Architecture

AWS Lambda operates on an event-driven model, where specific events trigger the execution of predefined functions. In this case, when an object like an image is uploaded to a specific S3 bucket, S3 generates an event notification. This event is sent to AWS Lambda, which then triggers the function you have created. The function can be designed to perform various tasks, such as processing the object, extracting metadata, or, as in this experiment, simply logging a message—“An Image has been added.”

Automation and Real-Time Processing

The integration of AWS Lambda with S3 is a prime example of automation in the cloud. It allows you to set up a real-time response system where actions are automatically triggered without manual intervention. This is particularly useful in scenarios where immediate processing or reaction is required, such as in applications that handle large volumes of incoming data or media files.

Scalability and Efficiency

One of the key advantages of using AWS Lambda with S3 is its scalability. Lambda functions can automatically scale to handle multiple events concurrently, ensuring that no matter how many objects are added to the S3 bucket, each will trigger the necessary function without delay. This setup is highly efficient, reducing operational overhead and ensuring that processes remain consistent and reliable, even under heavy workloads.

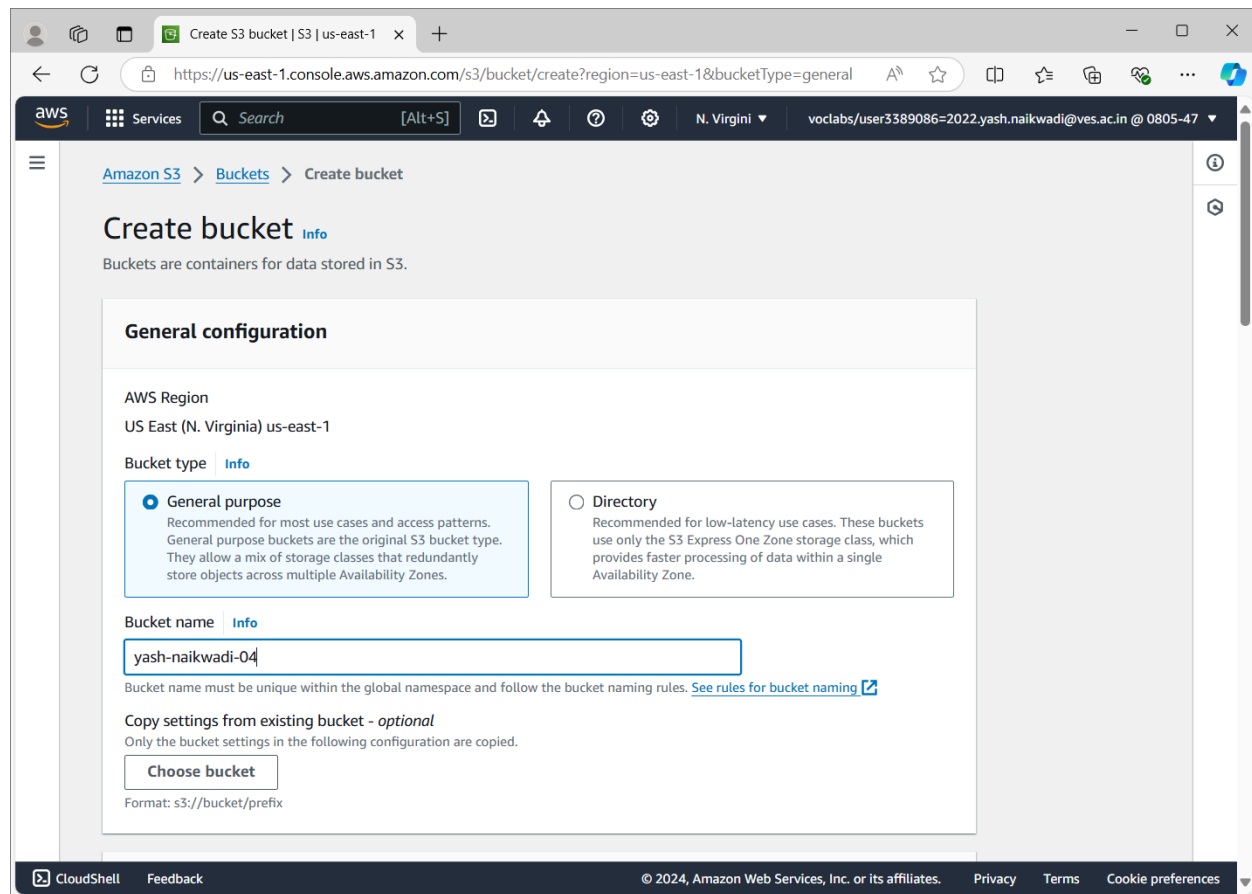
Use Cases and Practical Applications

This type of setup is widely used in various domains, including media processing, where images or videos uploaded to S3 can trigger Lambda functions to generate thumbnails or transcode files. It is also commonly used in data analytics, where data uploaded to S3 triggers processing functions that prepare the data for further analysis or storage. Another application is in backup systems, where new files added to S3 are immediately backed up or replicated in other locations.

Security Considerations

When working with AWS Lambda and S3, security is paramount. Each Lambda function interacts with S3 via permissions defined by AWS IAM roles and policies. These permissions must be carefully configured to ensure that the function has the necessary access rights—such as read and write permissions for the specific S3 bucket—without exposing unnecessary access to other resources. This ensures that the system remains secure while performing the required tasks efficiently.

Create an S3 Bucket. Enter a unique bucket name.



Amazon S3 > Buckets > Create bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3.

General configuration

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

Bucket type [Info](#)

- ☒ **General purpose**
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
- ☐ **Directory**
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

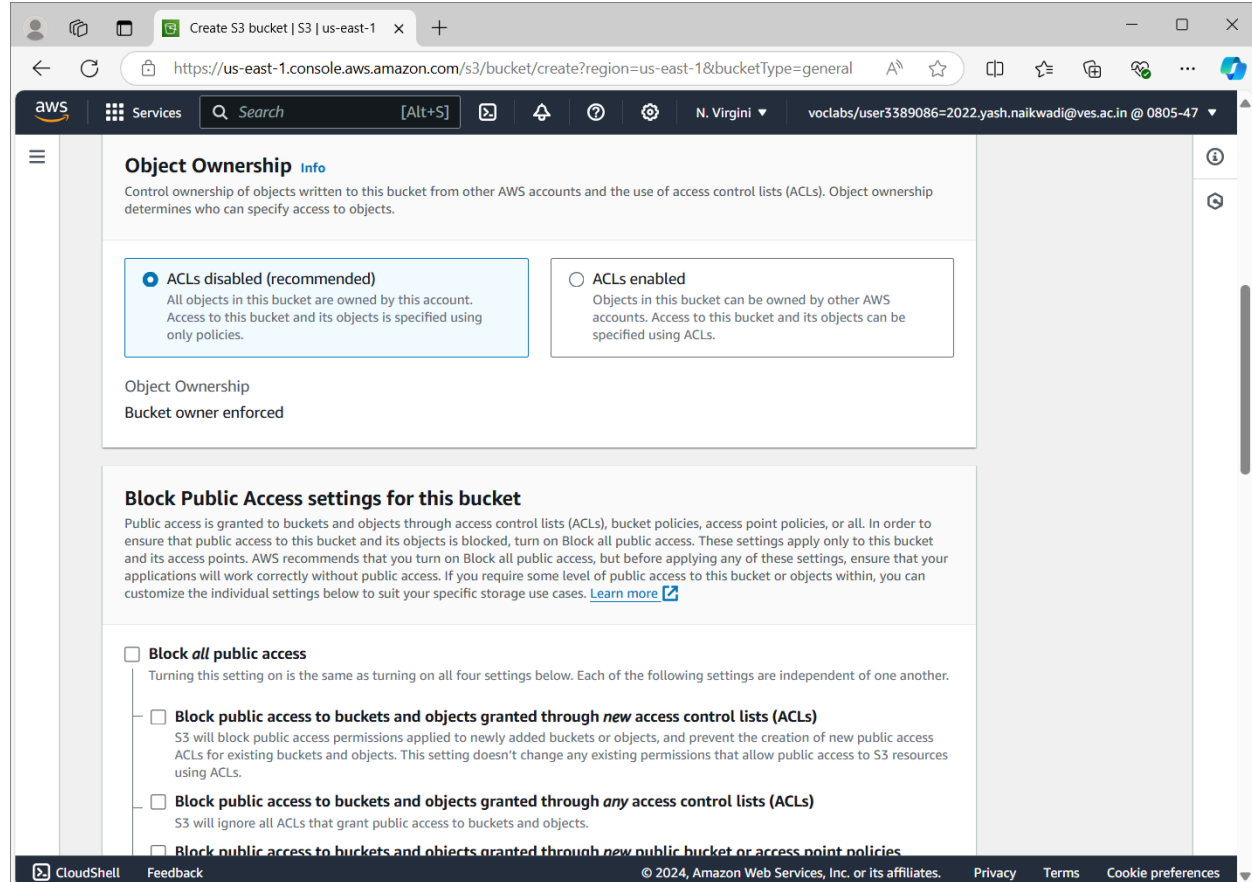
Bucket name [Info](#)

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

Format: s3://bucket/prefix

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- ☒ **ACLs disabled (recommended)**
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ☐ **ACLs enabled**
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

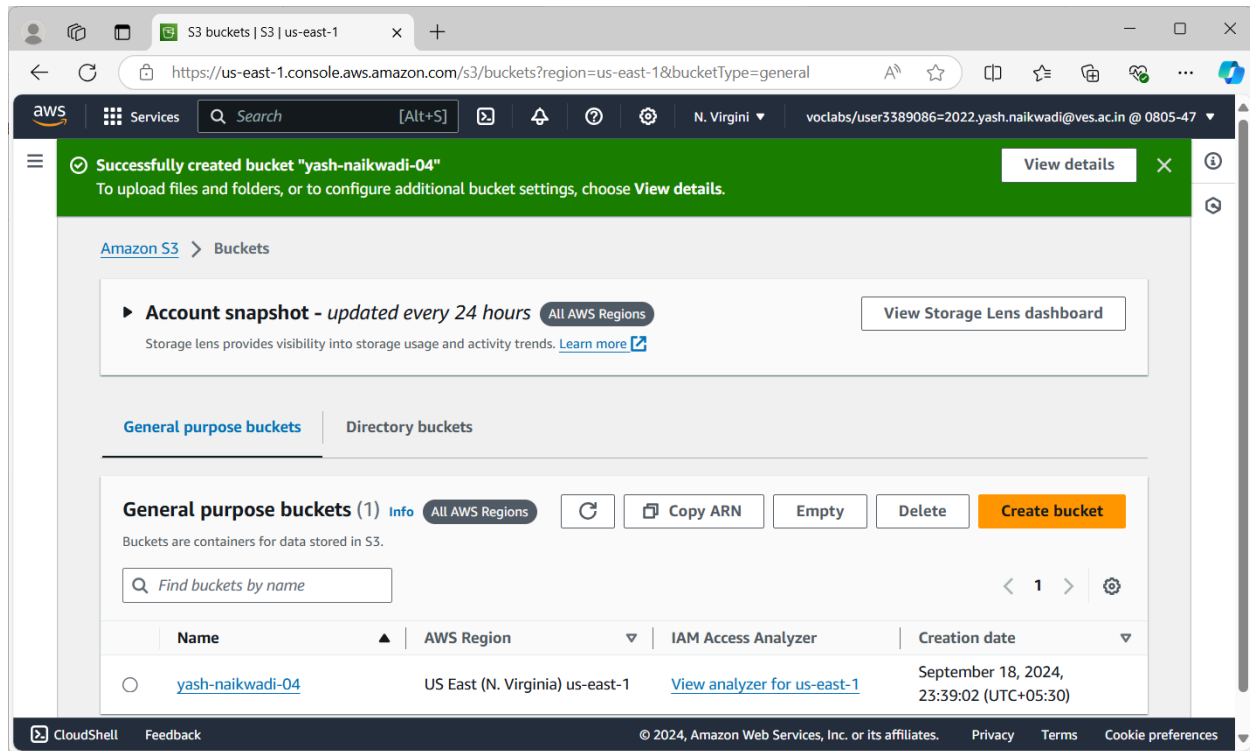
Object Ownership
Bucket owner enforced

Block Public Access settings for this bucket

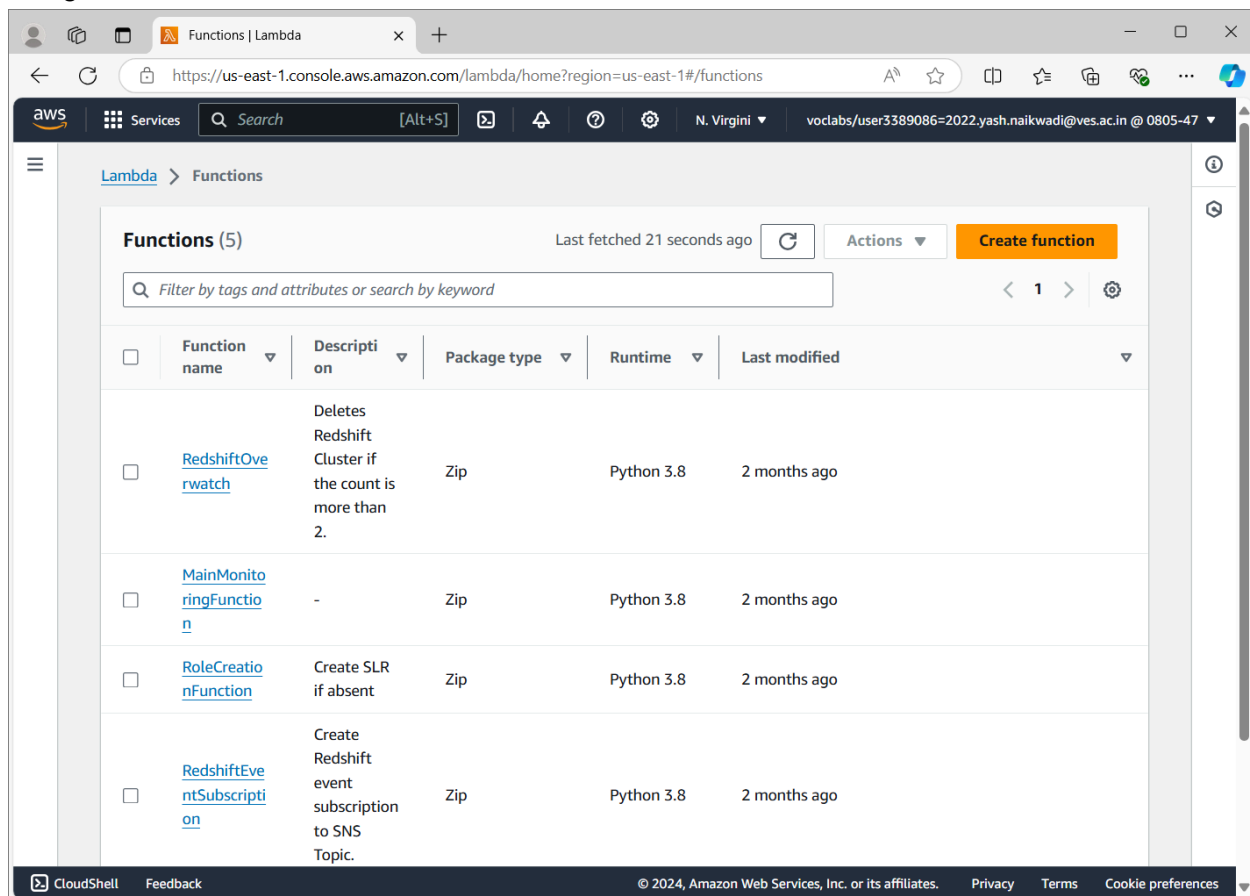
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access 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](#)

- ☒ **Block all public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings is independent of one another.
 - ☐ **Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
 - ☐ **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
 - ☐ **Block public access to buckets and objects granted through new public bucket or access point policies**

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Navigate to the Lambda service. Click on "Create function."



Choose "Author from scratch." Enter a name for your function. Select a runtime (Python 3.x).

The screenshot shows the 'Create function' page in the AWS console. The 'Author from scratch' option is selected. The function name is 'yash-naikwadi-04-lambda'. The runtime is set to 'Python 3.12'. The architecture is set to 'x86_64'.

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.

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.

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
☒ x86_64
☐ arm64

Permissions [Info](#)

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the 'Permissions' section of the 'Create function' page. The 'Use an existing role' option is selected. The existing role is 'LabRole'.

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

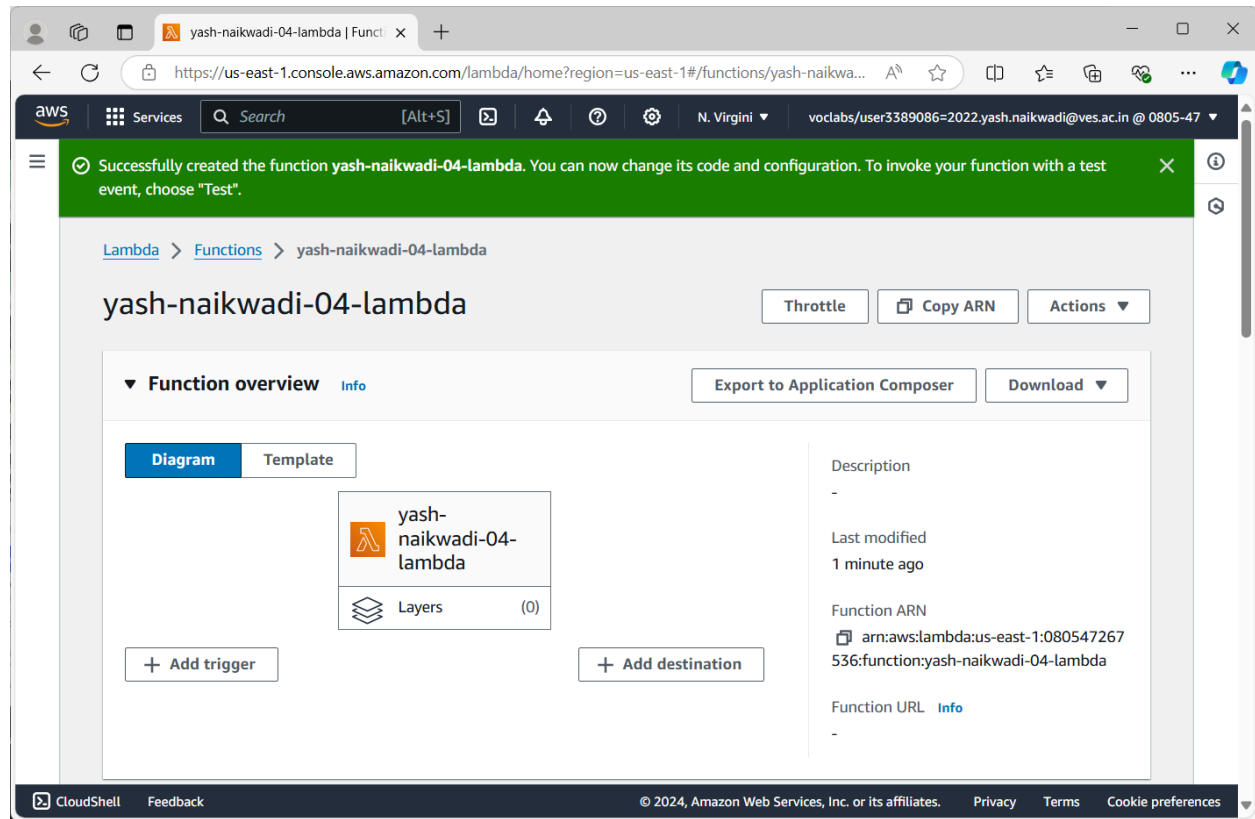
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

[View the LabRole role](#) on the IAM console.

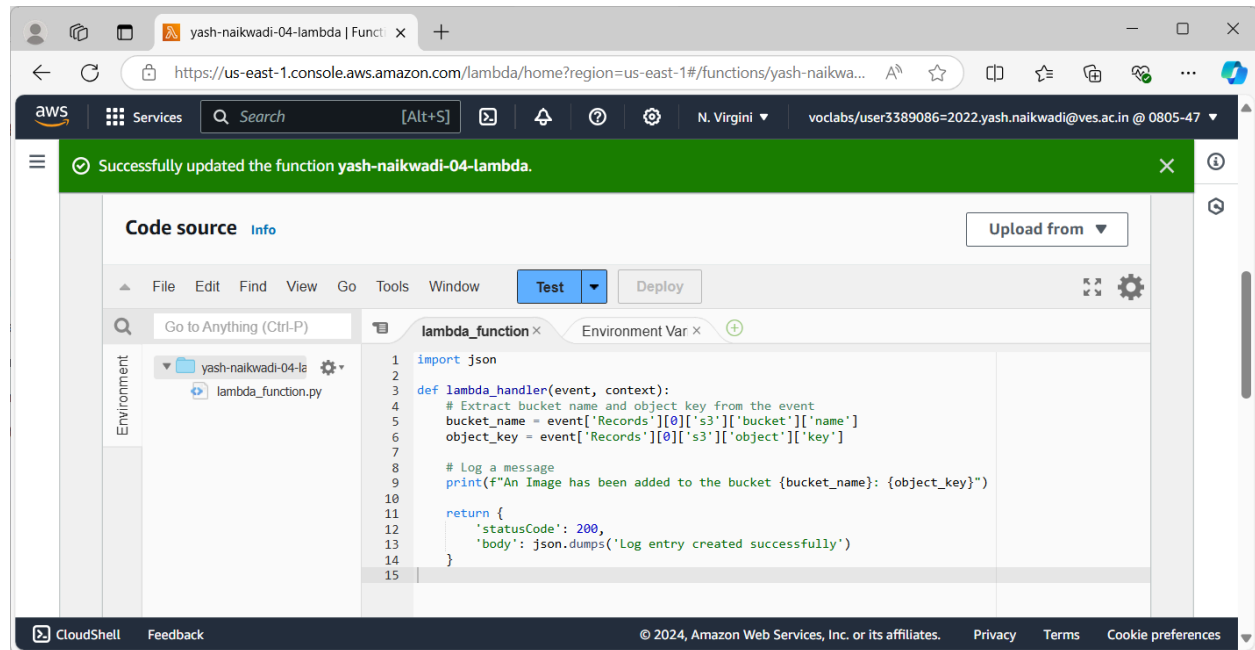
Advanced settings

Cancel

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



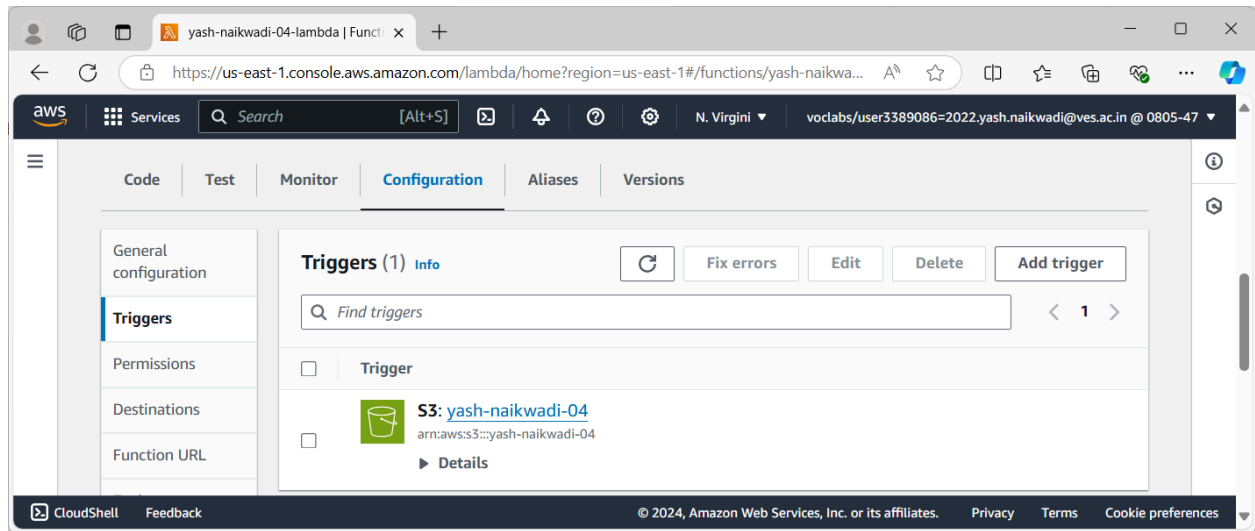
Write the Lambda Function Code. Click "Deploy" to save your changes.



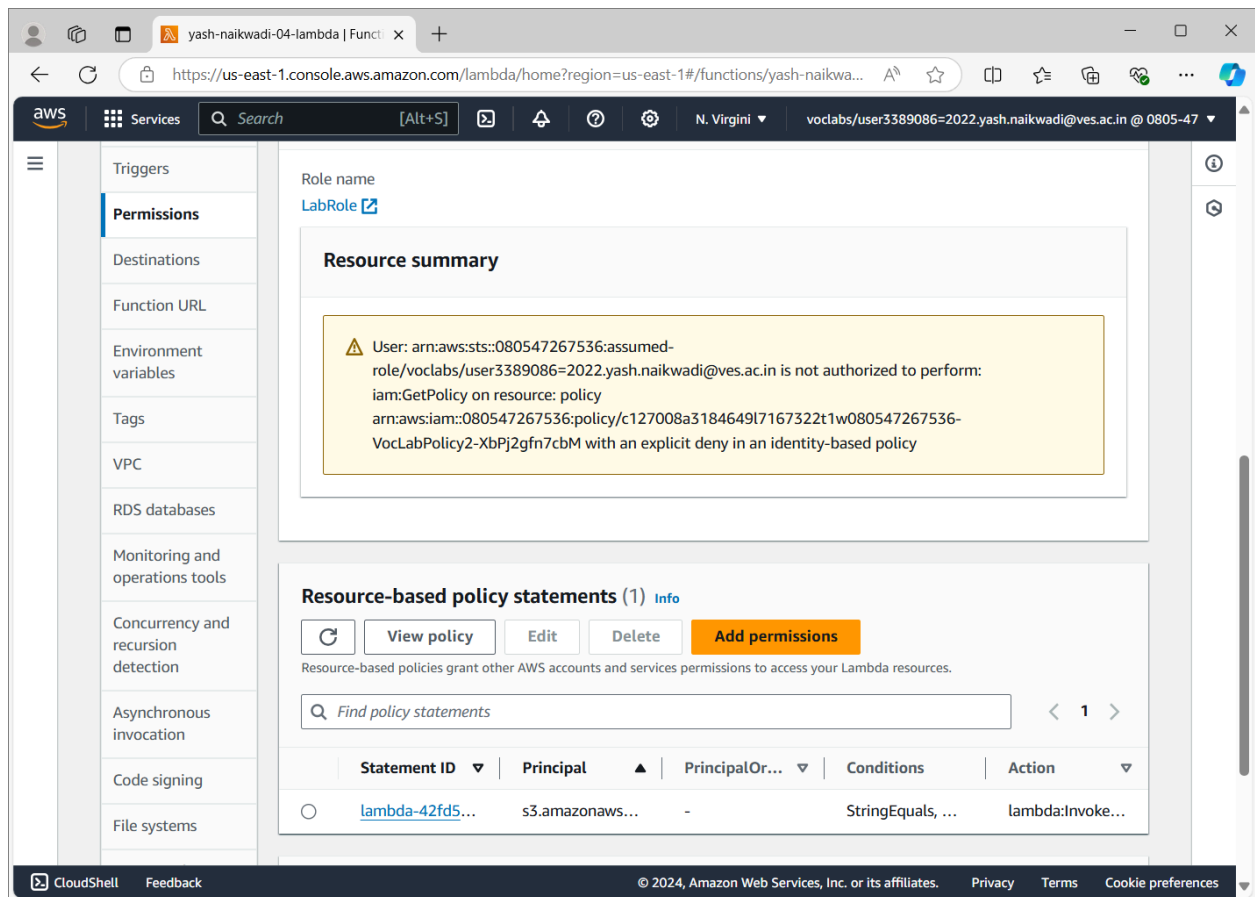
Select "S3" from the list of triggers. Choose the S3 bucket you created earlier. In the "Event type" dropdown, select "All object create events." Click "Add."

The screenshot shows the 'Add triggers' page in the AWS Lambda console. The 'Trigger configuration' section has 'S3' selected in the dropdown menu. Below it, the 'Bucket' field contains 's3/yash-naikwadi-04'. The 'Event types' dropdown is set to 'All object create events'. The page includes a search bar, navigation links, and a footer with copyright information.

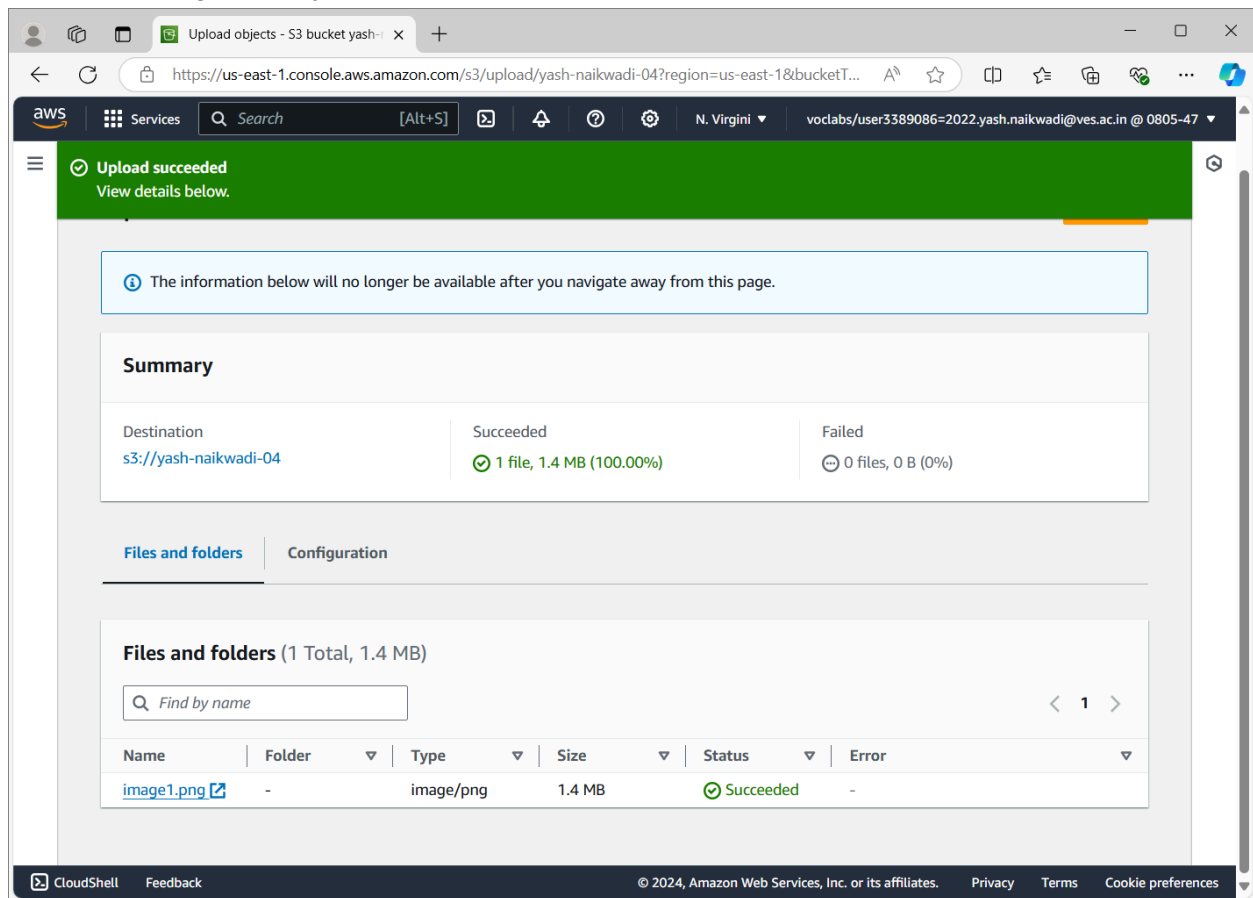
The screenshot shows the 'Function overview' page for the 'yash-naikwadi-04-lambda' function. A green notification banner at the top states: 'The trigger yash-naikwadi-04 was successfully added to function yash-naikwadi-04-lambda. The function is now receiving events from the trigger.' The 'Function overview' section includes a diagram showing the function connected to an S3 bucket. The right sidebar displays function details: Description, Last modified (28 minutes ago), Function ARN (arn:aws:lambda:us-east-1:080547267:536:function:yash-naikwadi-04-lambda), and Function URL.



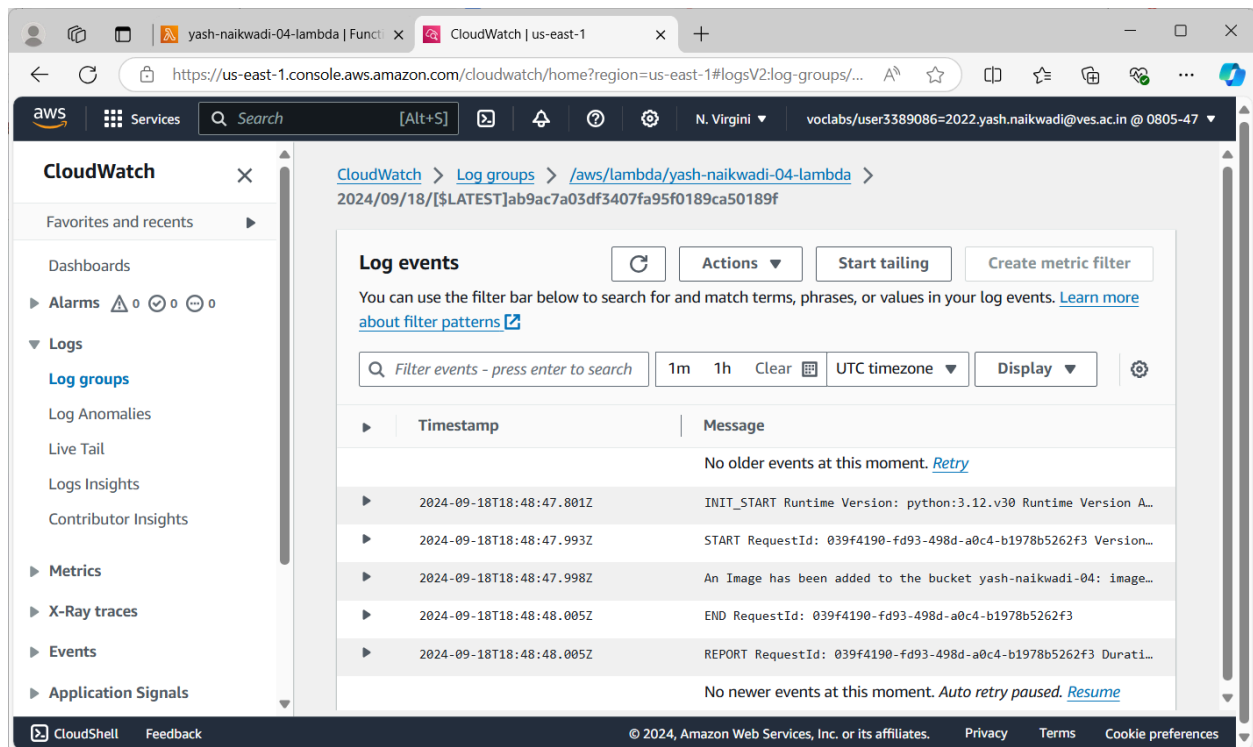
Navigate to the "Permissions" tab of your Lambda function. Ensure the Lambda function's execution role has the necessary permissions to access the S3 bucket.



Upload an image file to your S3 bucket.



Go to the "Monitoring" tab in your Lambda function to check the logs. Use CloudWatch Logs to view the output and confirm that the message "An Image has been added" has been logged.



CONCLUSION :

By integrating AWS Lambda with S3, you can create a seamless, automated process that responds instantly to changes in your S3 bucket. This not only improves efficiency but also allows for greater flexibility in handling data and media in real-time, making it a powerful tool in modern cloud-based applications.