PROJECT DOCUMENTATION

Overview

Create an AWS LAMBDA FUNCTION to backup files/folder from an EC2 Instance to S3 bucket.

The Lambda function should have the following features...

It should be able to copy/backup files from a folder in a RHEL EC2 Instance to an S3 bucket.

it should create a date wise backup folder

It should maintain the folder structure in S3 bucket

It should run at scheduled frequency.

It should maintain a log of all the file/folder transfers.

Use of Lambda Functions in AWS

AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources, making it the fastest way to turn an idea into a modern, production, serverless applications.

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) applications, and only pay for what you use.

For reference, refer to the official docs: <https://aws.amazon.com/pm/lambda/?gclid=CjwKCAjwps-zBhAiEiwALwsVYaMVZzUhQin5HxRABXPv3hvyrtGSecD8gETEd4Dx_VJqBFdEOkzZEBoCK20QAvD_BwE&trk=5cc83e4b-8a6e-4976-92ff-7a6198f2fe76&sc_channel=ps&ef_id=CjwKCAjwps-zBhAiEiwALwsVYaMVZzUhQin5HxRABXPv3hvyrtGSecD8gETEd4Dx_VJqBFdEOkzZEBoCK20QAvD_BwE:G:s&s_kwcid=AL!4422!3>!

Prerequisites

* An AWS account
* An EC2 instance running on RHEL.
* An S3 bucket to store the backups.
* Necessary IAM roles and policies.

Approach Used

The following function has been performed by using Amazon eventbridge scheduler with Lambda function. The scheduler is set to trigger the lambda function at a scheduled frequency. This initiates the lambda function which contains the code . The code connects with the EC2 instance and the s3 bucket with the help of the required policies attached to it using IAM roles. It then carries out a backup mechanism. The algorithm regularly maintains a last-backup.txt file which stores the last time when the backup was performed . This helps the function to backup new files every time the function is triggered. The process is simple. There is a file in EC2 instance let’s say ‘filepath.txt’ which contains the paths of the folders which need to be backed up in S3 bucket. (Make sure the paths are separated by a new line). The function will read this file, run a loop and copy all the new files to S3 bucket. The function is also responsible for making and appending log records inside the same S3 bucket.

Following steps were taken in the process-

Step 1: Creating an EC2 instance.

Search for the EC2 service in AWS console.

->Launch an EC2 instance.  
-> Select RHEL in amazon machine image and the instance type.

-> Generate a key pair and save.

Step 2: Create an S3 bucket.

->Search for the S3 service in AWS console.

-> Give the required name and click on create bucket.

The bucket and the instance are now ready. You can control the EC2 instance state by either starting, stopping or terminating the instance.

Step 3: Configure IAM roles

We require the creation of two IAM roles and a custom policy.

Lambda function IAM Role

Following policies will be attached to the Lambda function.

-> AmazonS3FullAccess

->AmazonSSMFullAccess

-> CloudWatchLogsReadOnlyAccess

IAM role for EC2 instance

->AmazonSSMFullAccess

->AmazonS3FullAccess (List, read,write)

Step 4: Install aws cli on RHEL instance.

To install aws cli on RHEL instance, run the following commands inside the instance

sudo yum update -y

sudo yum install -y unzip curl

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

Verify the installation using aws –-version on the terminal.

Step 5: Install SSM agent on RHEL .

* Use PuTTY or any other SSH client to connect to your RHEL EC2 instance.
* Switch to the Root User.

sudo su –

* Install the SSM Agent Using DNF.

dnf install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux\_amd64/amazon-ssm-agent.rpm

* Start and Enable the SSM Agent.

systemctl start amazon-ssm-agent

* Enable the SSM agent to start on boot.

systemctl enable amazon-ssm-agent

* Verify the SSM Agent Status

systemctl status amazon-ssm-agent

Step 6: Connecting EC2 instance using ssh(for adding files to EC2).

->Open the command line interface in the directory where ec2 keypair is stored. Connect the ec2 instance using ssh command given in the connect option in ec2 console. Alternatively it can be opened using PUTTY.

->Add the ‘filepath.txt’ to the ec2-user.

Step 5: Verify and Enable AWS-RunShellScript Document

* Go to the AWS Management Console.
* Open the Systems Manager service.
* Under Shared Resources, select Documents.
* Search for AWS-RunShellScript.
* If it is not enabled, enable the Document:
* If the document is not available, ensure that you have the necessary permissions and contact AWS Support if needed.

Step 6: Create and deploy the Lambda Function.

Step 7: Setting up the event trigger in amazon eventbridge scheduler

->Open the scheduler in amazon eventbridge

->Create a topic an choose AWS lambda as the target. Select the required lambda function.

->Add the cron expression for the schedule. This will be the time period in which the lambda function will get triggered.

Step 8: Test the Backup Process

->Invoke the Lambda Function:

->Manually trigger the Lambda function from the AWS Lambda console .

->Monitor Execution

->Check the AWS CloudWatch logs to monitor the execution and debug any issues.

Creating the Lambda Function

Create a Lambda function with the IAM role attached. Give necessary environment variables.

Here is the full code:

import boto3

import os

from datetime import datetime

import time

# Initialize clients

ssm = boto3.client('ssm')

s3 = boto3.client('s3')

# Retrieve environment variables

instance\_id = os.getenv('INSTANCE\_ID')

s3\_bucket\_name = os.getenv('S3\_BUCKET\_NAME')

filepath = os.getenv('FILEPATH')

# Get current date

date = datetime.now()

year = date.year

month = f"{date.month:02d}"

day = f"{date.day:02d}"

s3\_key\_prefix = f"{year}/{month}/{day}"

log\_s3\_key = f"logs/log.txt"

def lambda\_handler(event, context):

    final\_log\_message = ''

    found\_new\_files = False

    try:

        # Step 1: Get the last backup time

        last\_backup\_time = ''

        directories\_file\_path = f"cat /{filepath}"

        file\_path\_content = execute\_shell\_command(instance\_id, directories\_file\_path)

        backup\_directories = file\_path\_content.strip().split('\n')

        try:

            last\_backup\_time\_command = 'cat /home/ec2-user/last-backup.txt'

            last\_backup\_time = execute\_shell\_command(instance\_id, last\_backup\_time\_command)

        except Exception:

            print('No last backup time found, assuming first backup.')

        for backup\_directory in backup\_directories:

            # Step 2: Find new files

            find\_new\_files\_command = (

                f"find {backup\_directory} -type f -newermt \"{last\_backup\_time.strip()}\""

                if last\_backup\_time else f"find {backup\_directory} -type f"

            )

            new\_files\_list = execute\_shell\_command(instance\_id, find\_new\_files\_command)

            new\_files = list(filter(None, new\_files\_list.strip().split('\n')))

            if not new\_files:

                final\_log\_message += f"Backup from {backup\_directory} on {year}-{month}-{day} at {datetime.now().strftime('%H:%M:%S')}: No new files found\n"

                continue

            found\_new\_files = True

            final\_log\_message += f"Backup from {backup\_directory} on {year}-{month}-{day}: {len(new\_files)} new files backed up\n"

            # Step 3: Backup new files to S3 and get file sizes

            for file in new\_files:

                file\_size = get\_file\_size(instance\_id, file)

                s3\_key = f"{s3\_key\_prefix}/{backup\_directory.replace('/', '\_')}/{file.replace(f'{backup\_directory}/', '')}"

                s3\_copy\_command = f"aws s3 cp {file} s3://{s3\_bucket\_name}/{s3\_key}"

                send\_shell\_command(instance\_id, s3\_copy\_command)

                final\_log\_message += f"{file.replace(f'{backup\_directory}/', '')} (Size: {file\_size} bytes) backed up at {datetime.now().strftime('%H:%M:%S')}\n"

        # Step 4: Update last backup time

        update\_last\_backup\_time\_command = f"date +\"%Y-%m-%dT%H:%M:%S\" > /home/ec2-user/last-backup.txt"

        send\_shell\_command(instance\_id, update\_last\_backup\_time\_command)

        # Step 5: Update log file in S3

        update\_log(final\_log\_message)

        return {

            'statusCode': 200,

            'body': {

                'message': 'Backup successful',

                's3Bucket': s3\_bucket\_name,

                's3KeyPrefix': s3\_key\_prefix,

                'logFilePath': log\_s3\_key

            }

        }

    except Exception as err:

        print(f"Error: {err}")

        return {

            'statusCode': 500,

            'body': {

                'error': 'Backup failed',

                'message': str(err)

            }

        }

def execute\_shell\_command(instance\_id, command):

    send\_command\_params = {

        'InstanceIds': [instance\_id],

        'DocumentName': 'AWS-RunShellScript',

        'Parameters': {

            'commands': [command]

        }

    }

    response = ssm.send\_command(\*\*send\_command\_params)

    command\_id = response['Command']['CommandId']

    print(f"Command sent: {command\_id}")

    return wait\_for\_command\_completion(command\_id, instance\_id)

def wait\_for\_command\_completion(command\_id, instance\_id):

    status = 'InProgress'

    while status == 'InProgress':

        time.sleep(1)

        result = ssm.get\_command\_invocation(CommandId=command\_id, InstanceId=instance\_id)

        status = result['Status']

        print(f"Command status: {status}")

        if status == 'Success':

            return result['StandardOutputContent']

        elif status == 'Failed':

            raise Exception(result['StandardErrorContent'])

def send\_shell\_command(instance\_id, command):

    return execute\_shell\_command(instance\_id, command)

def get\_file\_size(instance\_id, file\_path):

    size\_command = f"stat -c %s {file\_path}"

    file\_size = execute\_shell\_command(instance\_id, size\_command)

    return file\_size.strip()

def update\_log(log\_message):

    # Fetch the existing log file from S3

    existing\_log\_content = ''

    try:

        data = s3.get\_object(Bucket=s3\_bucket\_name, Key=log\_s3\_key)

        existing\_log\_content = data['Body'].read().decode('utf-8')

    except s3.exceptions.NoSuchKey:

        print('No existing log file found, creating a new one.')

    # Append new log message to the existing log content

    if log\_message not in existing\_log\_content:

        updated\_log\_content = existing\_log\_content + log\_message

        upload\_params = {

            'Bucket': s3\_bucket\_name,

            'Key': log\_s3\_key,

            'Body': updated\_log\_content,

            'ContentType': 'text/plain'

        }

        s3.put\_object(\*\*upload\_params)

Code explanation

AWS Clients: Initializes the AWS Systems Manager (SSM) and S3 clients.

Environment Variables: Retrieves environment variables for the EC2 instance ID (INSTANCE\_ID), S3 bucket name (S3\_BUCKET\_NAME), and the file path containing directories to back up (FILEPATH).

Date and S3 Key Prefix: Gets the current date to create a directory structure in the S3 bucket (s3\_key\_prefix).

**Lambda Handler**: The lambda\_handler function is the entry point for the Lambda function. It processes events and context, performing the backup operations.

**Retrieve Last Backup Time**:

* Executes a shell command on the EC2 instance to read the last backup time from a file (/home/ec2-user/last-backup.txt).
* If the file does not exist, assumes this is the first backup.

**Read Directories to Backup**:

* Executes a shell command on the EC2 instance to read the content of the file specified by the FILEPATH environment variable, which contains the directories to back up.
* Splits the content by newline to get individual directory paths.

**Find New Files**:

* For each directory, finds new files that have been modified since the last backup time (if available) using the find command.
* If there are new files, logs the number of files found.

**Backup New Files to S3**:

* Iterates over the new files and:
  + Gets the file size using the stat command.
  + Copies the file to the S3 bucket using the aws s3 cp command.
  + Logs the backup operation with the file size and timestamp.

**Update Last Backup Time**:

* Updates the last backup time file on the EC2 instance to the current time.

**Update Log File in S3**:

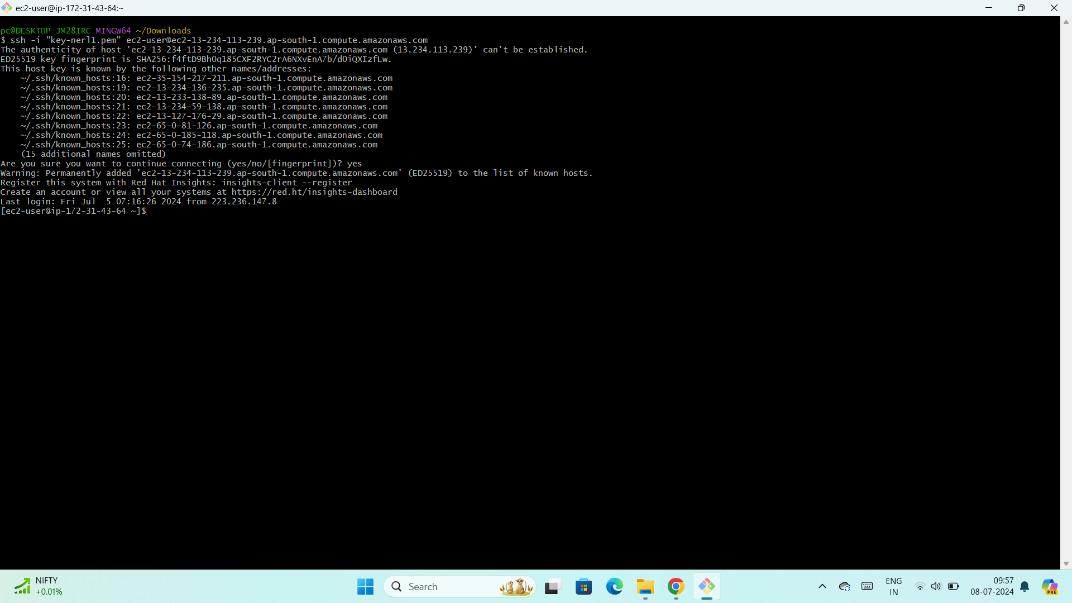
* Fetches the existing log file from S3 (if it exists) and appends the new log messages.
* Uploads the updated log file back to S3.
*  **execute\_shell\_command**: Sends a shell command to the EC2 instance using SSM and waits for the command to complete.
*  **wait\_for\_command\_completion**: Polls the status of the SSM command until it completes (either successfully or with failure).
*  **send\_shell\_command**: Wrapper around execute\_shell\_command for sending commands.
*  **get\_file\_size**: Executes a shell command to get the size of a file.
*  **update\_log**: Fetches the existing log file from S3, appends new log messages, and uploads the updated log file.

The Lambda function has error handling to catch and log any exceptions that occur during the backup process, returning a 500 status code in case of failure.

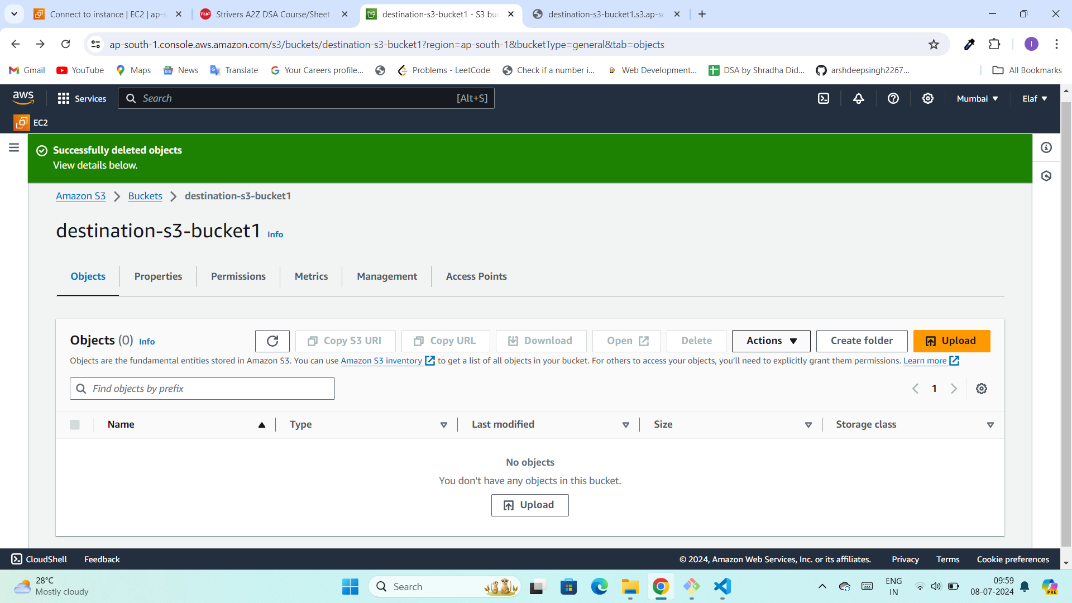
Result:

Let’s perform a demo on the described lambda function.

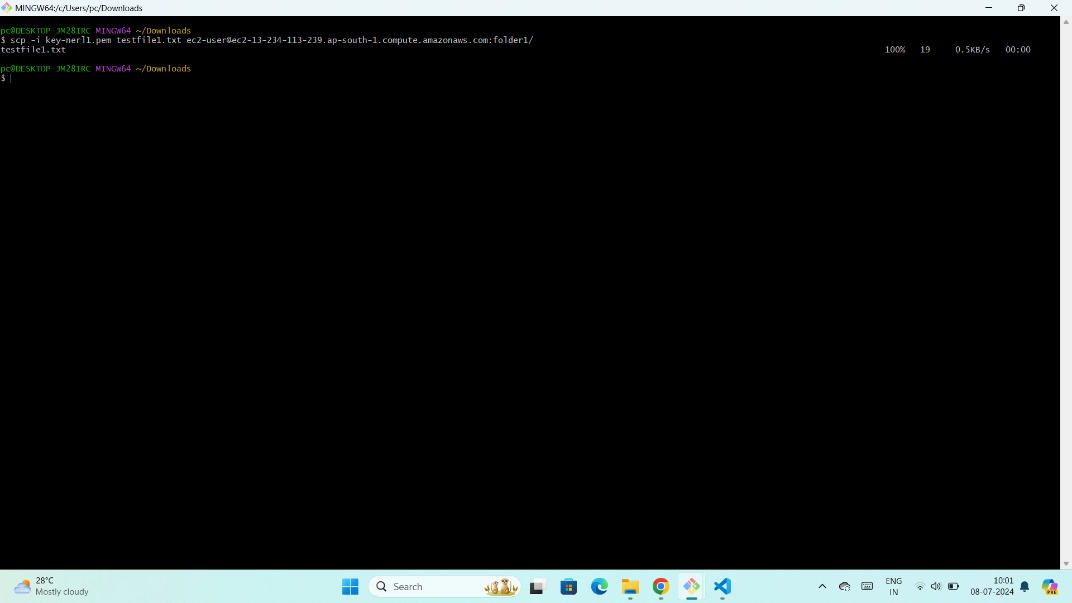
1. Start by connecting to the EC2 instance.



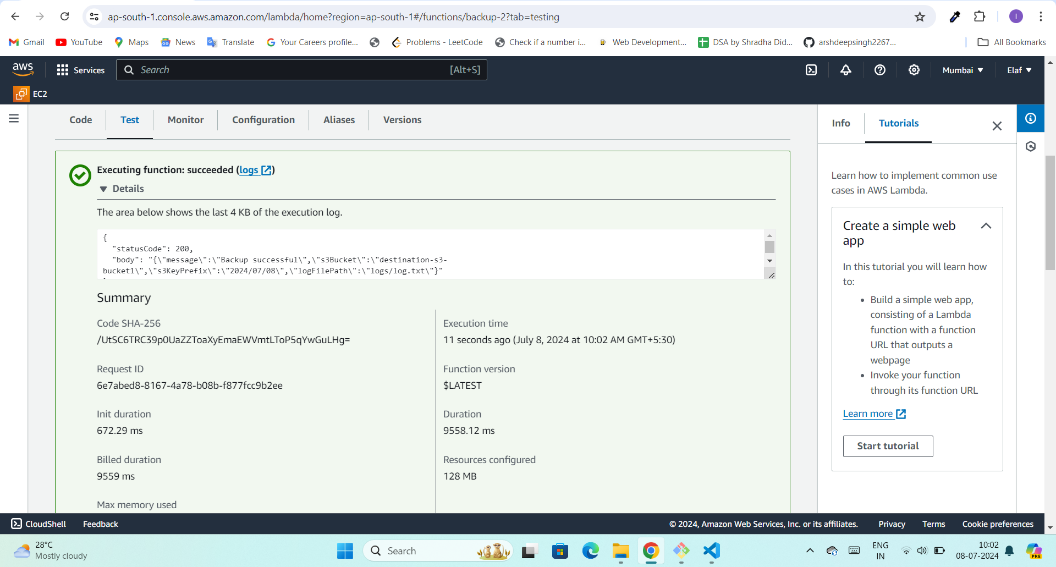
1. Cd inside your desired folder.
2. At this point my S3 bucket is empty.



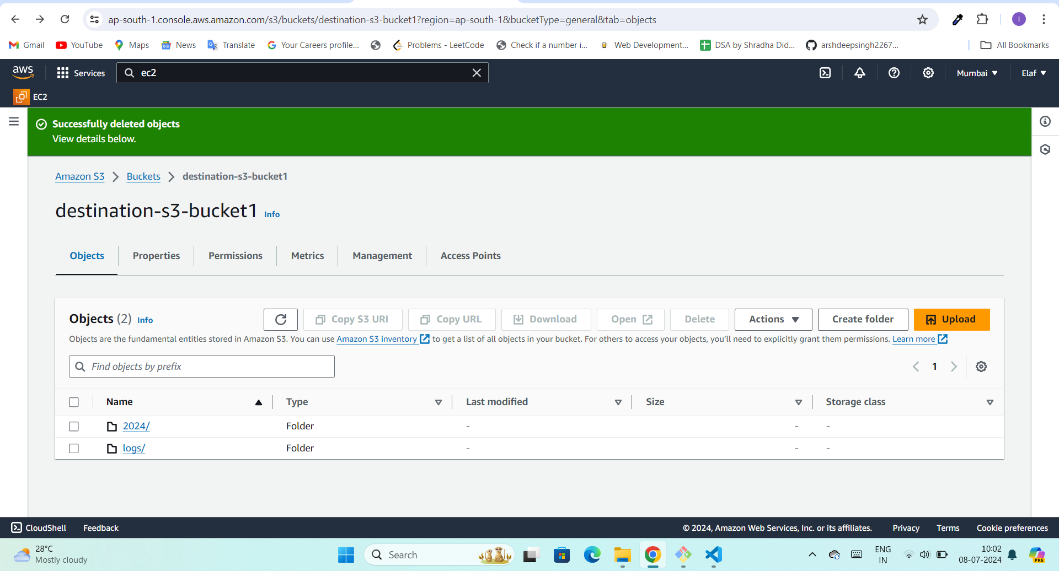
1. Let’s add a file to the EC2 instance.



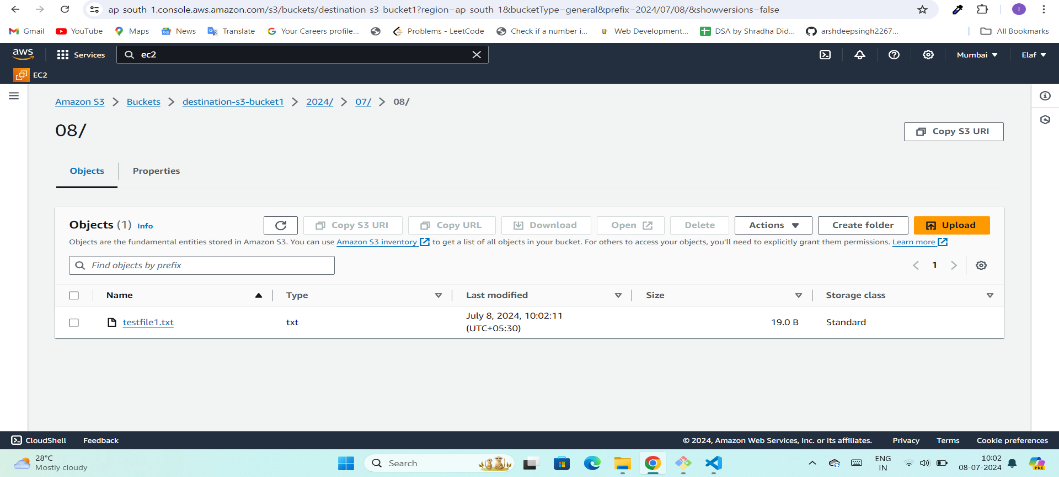
1. Now we’ll manually trigger the lambda function.



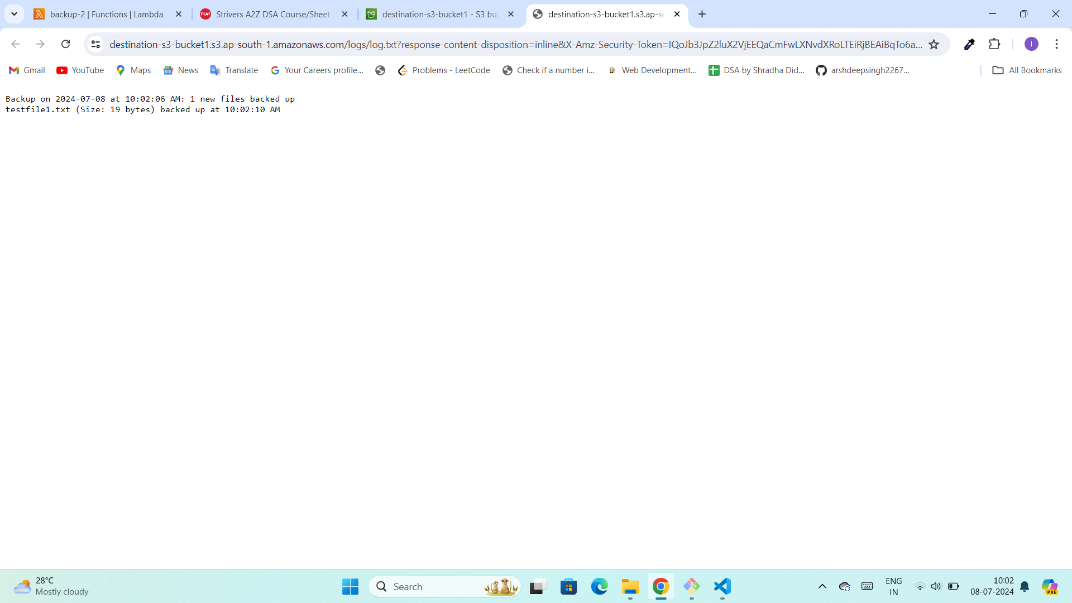
1. The function is successfully executed with the log message backup successful. Now we can view the S3 bucket.



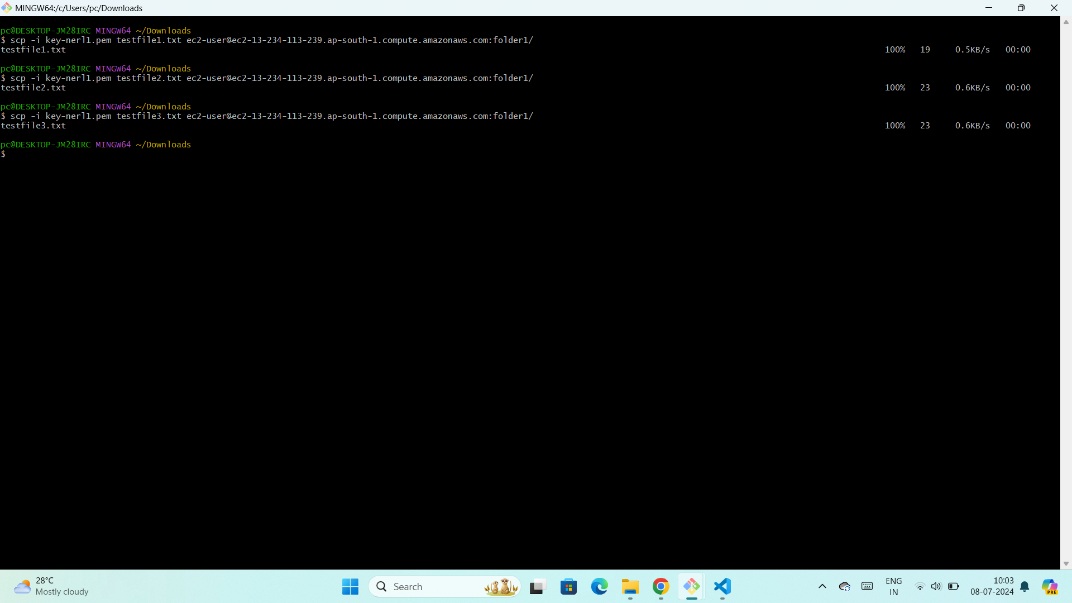
7.Let’s check into the 2024 backup directory .



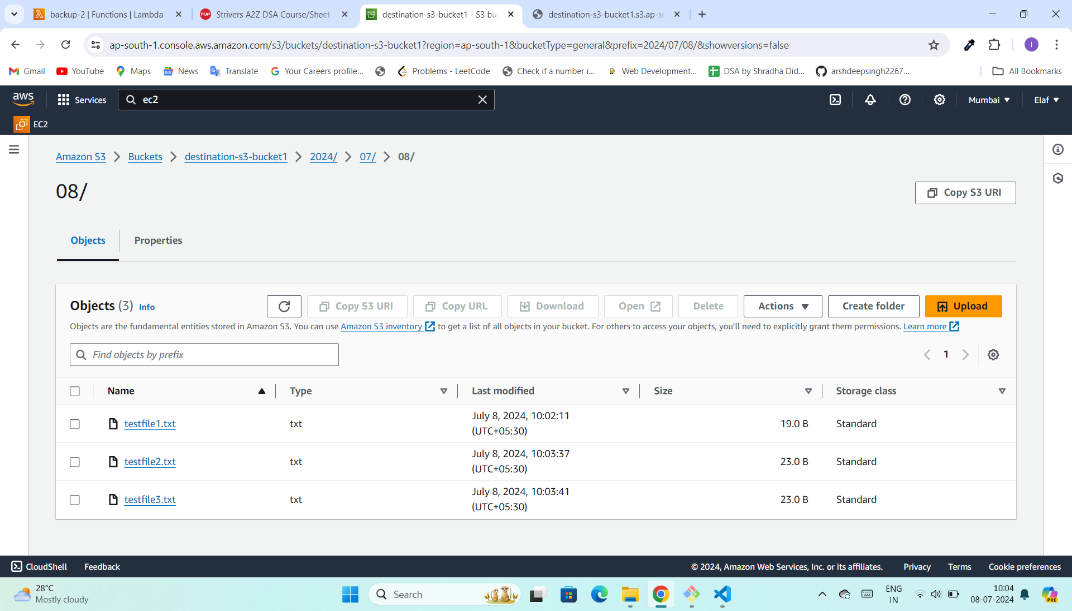
8. This is how the logs.txt file will look.



9. Let’s add some more files in the EC2 instance folder.



10. After repeating the same process, the S3 bucket will get updated with new files and logs.



11. The logs file will get updated with the same.

