# Task 1:

Creating an EC2 Instance and Managing Files (Folder Creation, Zipping and Unzipping)

# Aim:

To launch an Amazon EC2 instance, create a folder and files inside it, and perform file compression and decompression operations (zip and unzip) to understand basic Linux commands and AWS instance operations.

# Algorithm:

Step 1: Launch EC2 Instance:

* Log in to the AWS Management Console.
* Navigate to EC2 and launch a new instance using Amazon Linux 2023.
* Configure instance details, key pair, and security group.
* Connect to the instance using SSH. Step 2: Create a Folder and Files:
* Use the mkdir command to create a new folder.
* Navigate into the folder using cd.
* Create three text files using the touch or echo command. Step 3: Verify Files:
* Use ls to list and verify the created files. Step 4: Create a ZIP File:
* Install the zip utility if not available (sudo dnf install zip -y).
* Compress folder or files using:

zip -r project\_files.zip foldername

* Compress folder or files using:

zip -r project\_files.zip foldername Step5: Unzip the Files:

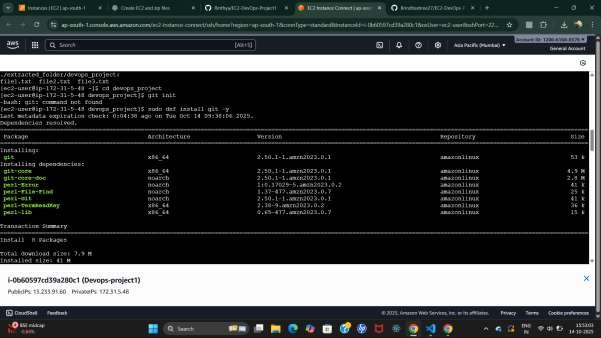
* Install the unzip utility if not available (sudo dnf install unzip -y).
* Extract files using:

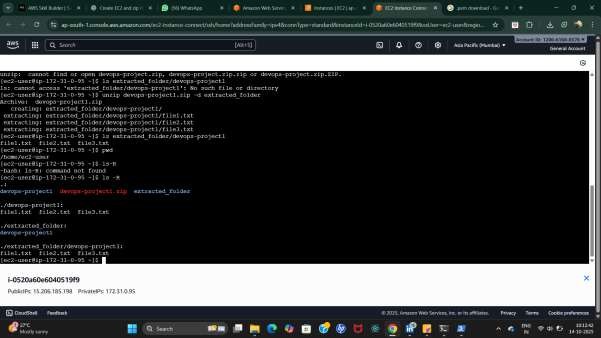
unzip project\_files.zip

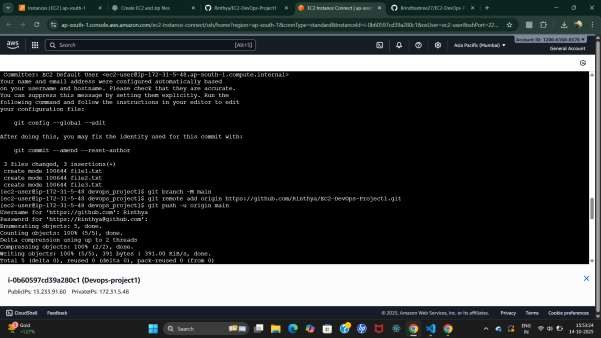
Step6: Verify Output:

* Use ls to confirm the files have been extracted successfully.

# Output:

****

****



**Github** : https://github.com/SujithaSadasivan/project

# Result:

An EC2 instance was successfully created and accessed using SSH. A folder containing three files was created, compressed into a ZIP file, and then extracted using Linux commands. The operations validated successful file handling, compression, and decompression within the AWS EC2 environment.

# Task 2:

Create an S3 bucket, upload new files into the bucket, enable versioning, and generate a pre-signed URL for secure file access.

# AIM:

To create a web-based interface using HTML that allows users to upload files to an Amazon S3 bucket, enable versioning for maintaining multiple versions of files, and generate secure, time-limited pre-signed URLs for controlled access to the uploaded files.

**ALGORITHM:**

1. **Start**
2. **AWS S3 Bucket Setup**
   * Log in to the AWS Management Console.
   * Navigate to the **S3 Service**.
   * Create a new bucket with a **unique name**.
   * Enable **Versioning** in the bucket properties to maintain multiple versions of files.

# Web Interface Design (HTML)

* + Create an HTML page with the following elements:
    - File input for selecting files.
    - Upload button to trigger the file upload process.
    - Display area for the generated pre-signed URL.

# AWS SDK Integration (JavaScript)

* + Include **AWS SDK for JavaScript** in the HTML page.
  + Configure AWS credentials securely using **IAM users** or **Cognito**.
  + Initialize the S3 client in JavaScript with the bucket name and region.

# File Upload Process

* + Capture the file selected by the user through the HTML input.
  + Use the **S3 upload API** to send the file to the S3 bucket.
  + The uploaded file will be stored in the bucket, and a new version will be created if versioning is enabled.

# Pre-Signed URL Generation

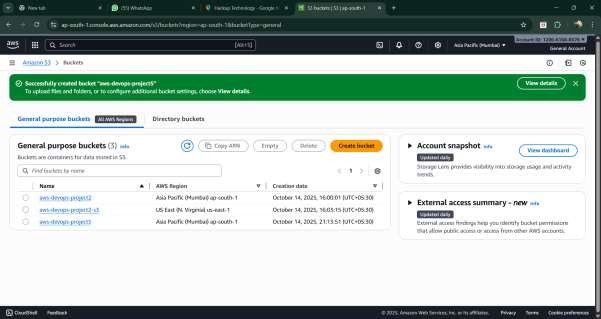
* + After upload, generate a **pre-signed URL** for the file using the S3 client.
  + The URL allows temporary access to the file without exposing AWS credentials.
  + Set an expiration time for secure, time-limited access.

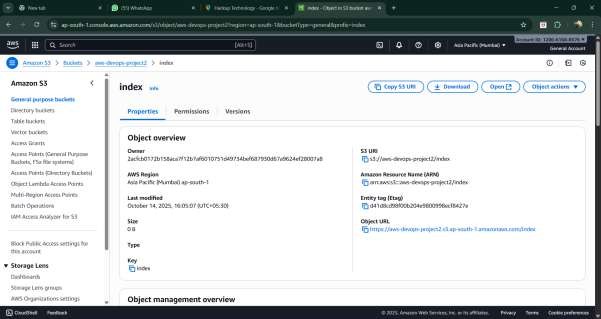
# Display URL to User

* + Show the pre-signed URL in the HTML interface so the user can copy or access it.

1. **End**

**Output Screenshots**

****



**RESULT:**

Files are successfully uploaded to the S3 bucket with versioning enabled, allowing multiple versions to be maintained. Pre-signed URLs are generated to provide secure, temporary access to the files. The system ensures safe and controlled file storage and sharing through the web interface.