MINI PROJECT -1

AMGOTH PRAVALIKA

1.IAM-Hands-On

2.Billing Alarams

3.S3 Buckets

4.EC2-Instances

5.Security Groups

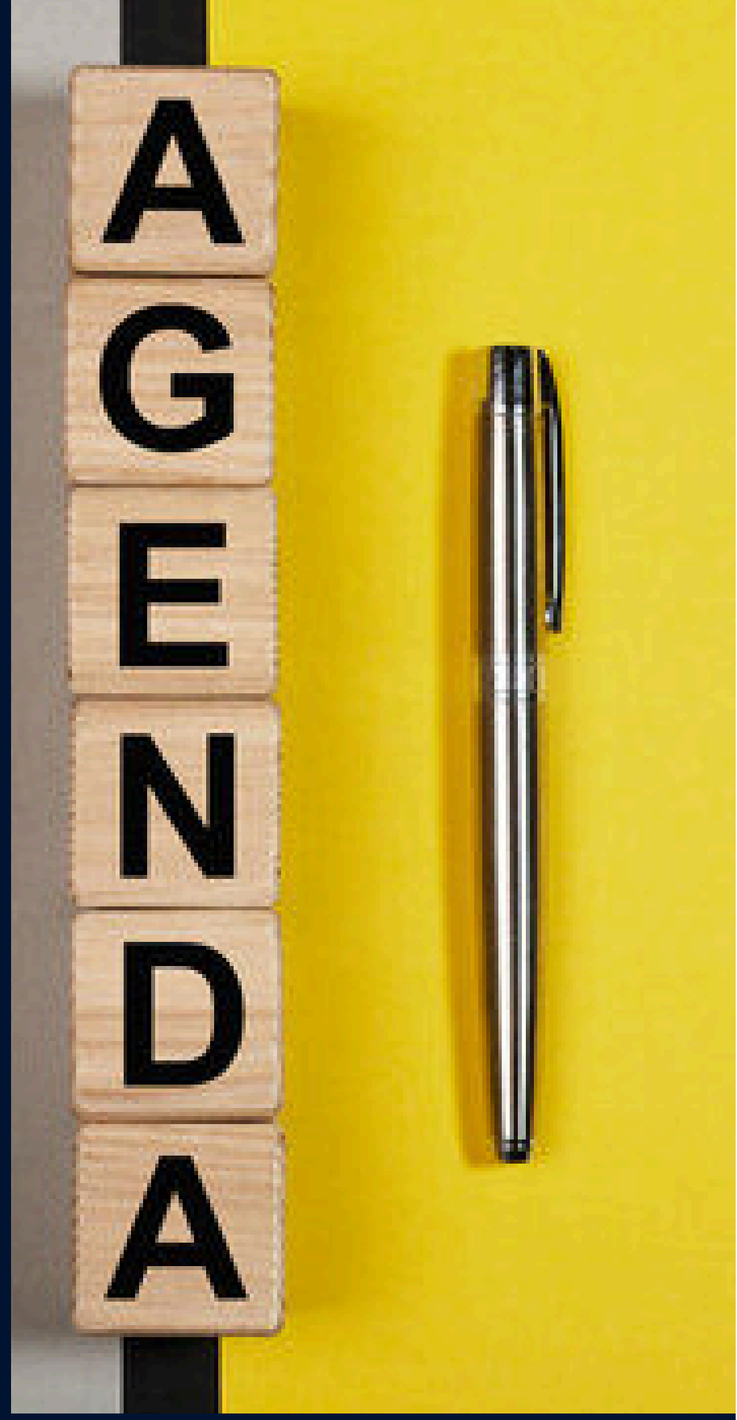
6.Volumes and Snapshots

7.AMI'S

8.Load Balancers

9.ASG and LT

10.RDS



Mini Project 1 Objectives

-Cost Monitoring and Management

-Data Storage and Security with S3

-Provisioning and Managing Compute Resources

-High Availability through Load Balancing and Auto-Scaling

-Disaster Recovery and Backup Management

-Reliable and Scalable Database Infrastructure

IAM HANDS ON

Identity and Access Management (IAM) is a system that controls user access to resources within an organization. It encompasses processes like verifying identities, granting permissions, and managing user roles to ensure that only approved individuals have access to certain systems and information. IAM strengthens security, supports compliance requirements, and improves operational effectiveness.

Multi-factorAuthentication

SETUPING AN MFA TO ROOT USER

--Enable MFA for Root User: Strengthen security by enabling Multi-Factor Authentication (MFA) for the root account.

--Create New User with Console Access: Add a new user with console login and review default permissions.

--Assign EC2-Only Permissions: Restrict the user to EC2 access only, then test to confirm they can’t access other services.

--Grant Admin Privileges: Upgrade user permissions to Admin for full access across AWS services

BILLING ALARM

A billing alarm notifies you when your cloud spending exceeds a set limit.

Set a Budget: Define your spending cap.

Create an Alarm: Monitor usage and costs.

Set Threshold: Trigger alerts when spending reaches a set percentage (e.g., 80%).

Get Notifications: Receive alerts via email or SMS.

S3 Bucket

Amazon S3 is a scalable storage service for managing data in buckets.

1.Create Bucket: Upload a test file and access it via URL (initial error).

2.Check Permissions: Make the file public and access it via the browser.

3.Enable Versioning: Upload a new file version and verify both versions.

4.Delete File: Use versioning to access the deleted file.

EC2-INSTANCE

1.LAUNCH AN EC2 INSTANCE: OF TYPE T2.MICRO WITH UBUNTU OS (FREE TIER).

2.Configure Security Group: Open required ports (e.g., SSH port 22) for external access.

3.Access Instance: Connect via SSH using PuTTY or similar software.

SECURITY GROUPS

1.Create Security Group: Name it "mynewsg" and review default rules.

2.Configure Inbound Rules: Allow ports 80 (HTTP) and 22 (SSH) from your IP range (e.g., /28).

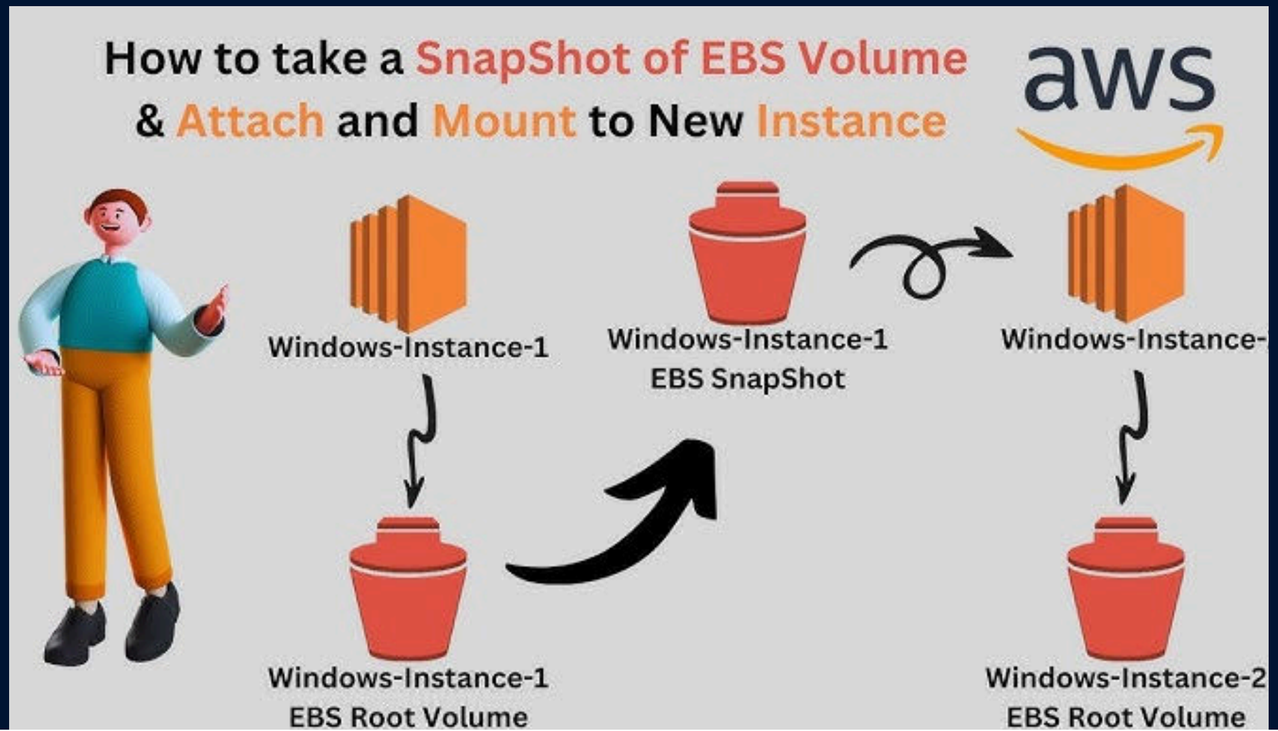
3.Attach to EC2: Link the security group to your existing EC2 instance.

4.Test Access: Verify SSH connectivity to the instance.

Volumes & Snapshots

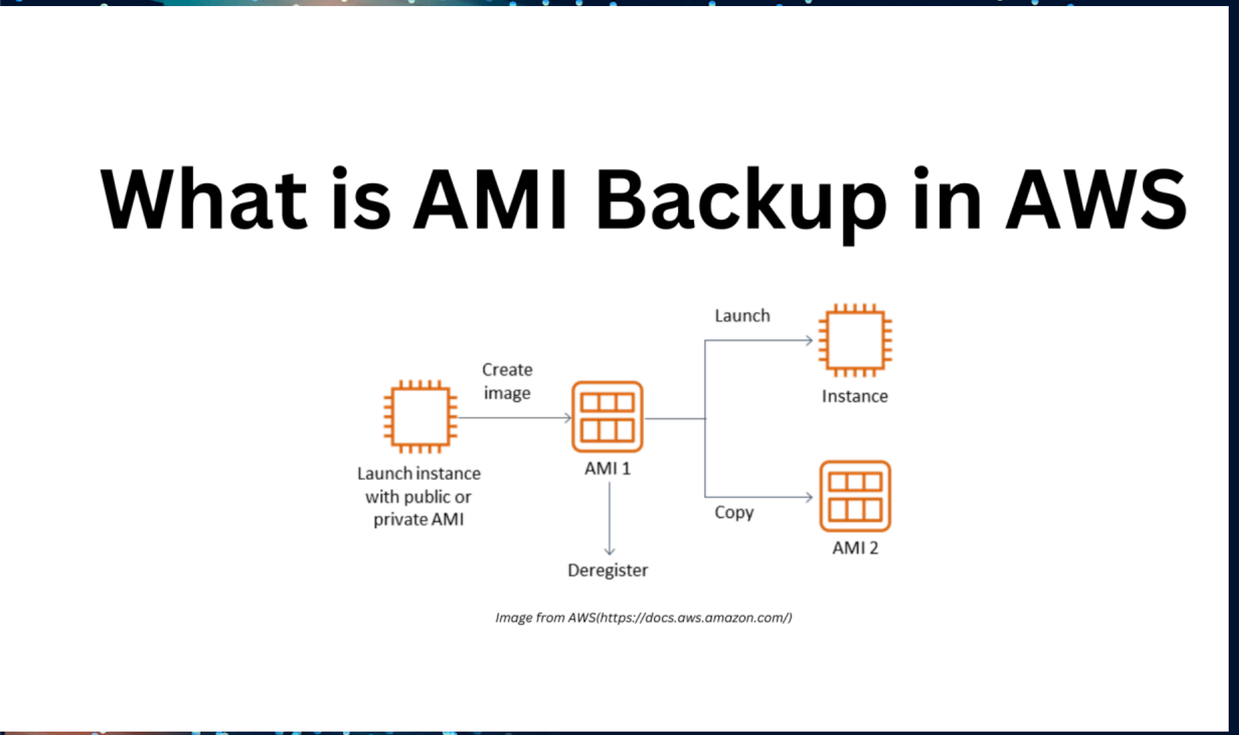
Volumes: Persistent block storage for EC2 instances, allowing data retention even when the instance is stopped.

Snapshots: Backups of EBS volumes stored in S3, used for data recovery or creating new volumes.



AWS AMIs

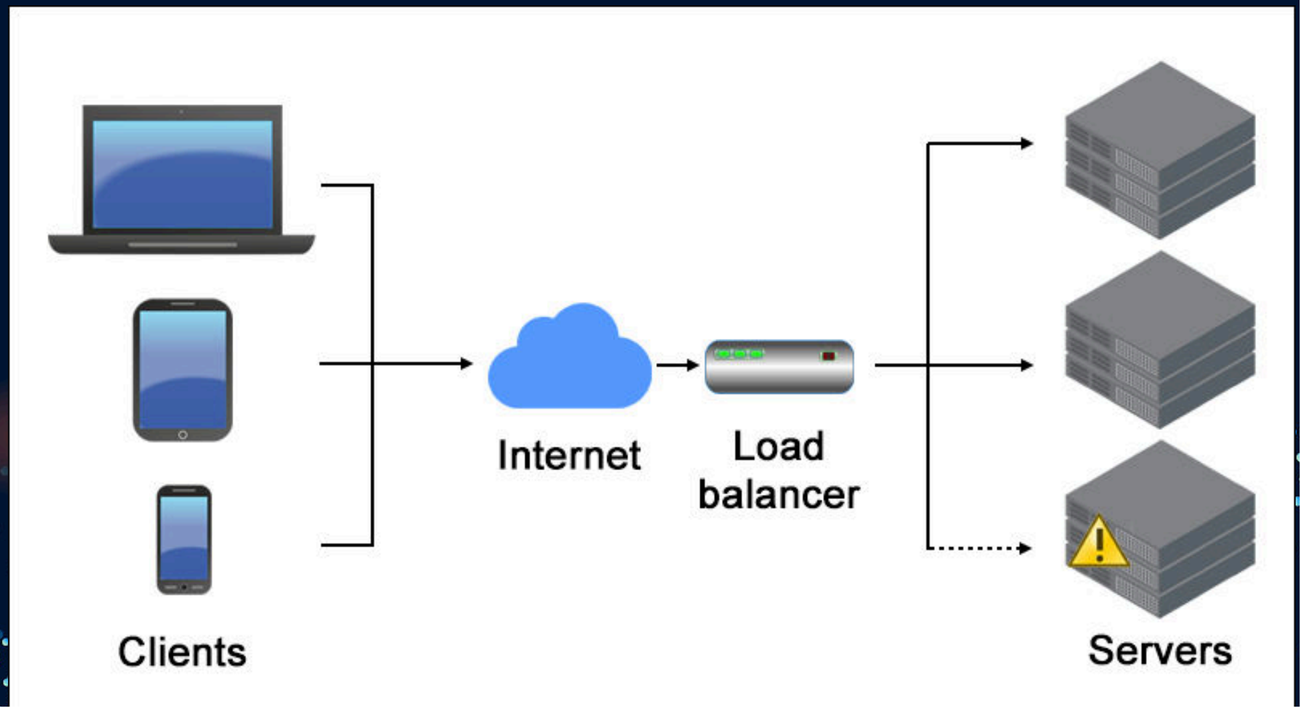
An AWS AMI (Amazon Machine Image)is a ready-to-use template that contains an operating system, software, and settings, making it easy to launch new EC2 instances with the same configuration.



Create an AMI from your running EC2 instance to save its configuration, data, and installed software. This enables you to launch new instances with the identical setup.

LOAD BALANCERS

A Load Balancer in AWS distributes incoming network traffic across multiple EC2 instances to ensure no single instance is overwhelmed. It helps improve application availability, performance, and fault tolerance by balancing traffic efficiently.



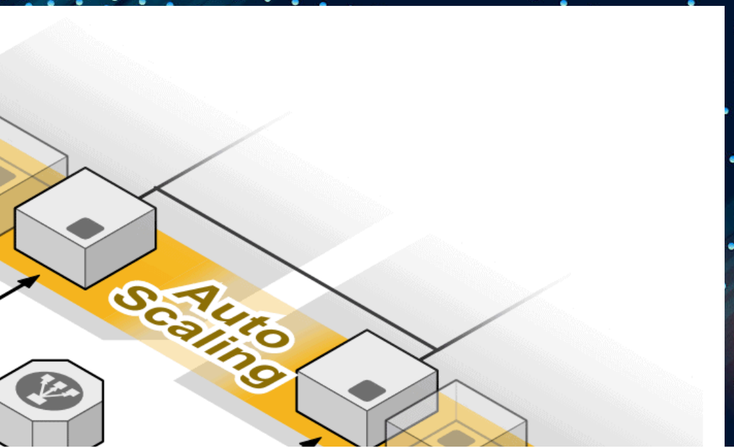
Auto Scaling Group(ASGs)

Auto Scaling in AWS automatically adjusts the number of EC2 instances based on traffic or demand. It helps maintain application performance by adding instances during high demand and reducing them when demand is low, optimizing both cost and availability.

1.Create Launch Template: Configure an Ubuntu server.

2.Set Up Auto Scaling Group: Use the template, set min 1 and max 3 instances.

3.Increase Capacity: Adjust the max limit and verify that new instances are launched.4o mini



RDS

AWS RDS (Relational Database Service)is a managed service that makes it easy to set up, operate, and scale relational databases like MySQL, PostgreSQL, and SQL Server in the cloud. It handles tasks like backups, patching, and scaling, allowing you to focus on your application.

1.Provision RDS: Create a MySQL instance in the AWS Management Console.

2.Configure Security Group: Open port 3306 for access.

3.Connect from EC2: Use a MySQL client to connect to the RDS instance

AWS RDS


Thank you!

GIT MINI PROJECT - 2

VERSION CONTROL SYSTEM

What is Git?

Git is a distributed version control system (VCS) used primarily for tracking changes in source code during software development. It allows multiple developers to work on the same project simultaneously, with the ability to track changes, revert to previous states, and merge code from different sources.

Key Features of Git

1.Distributed Version Control

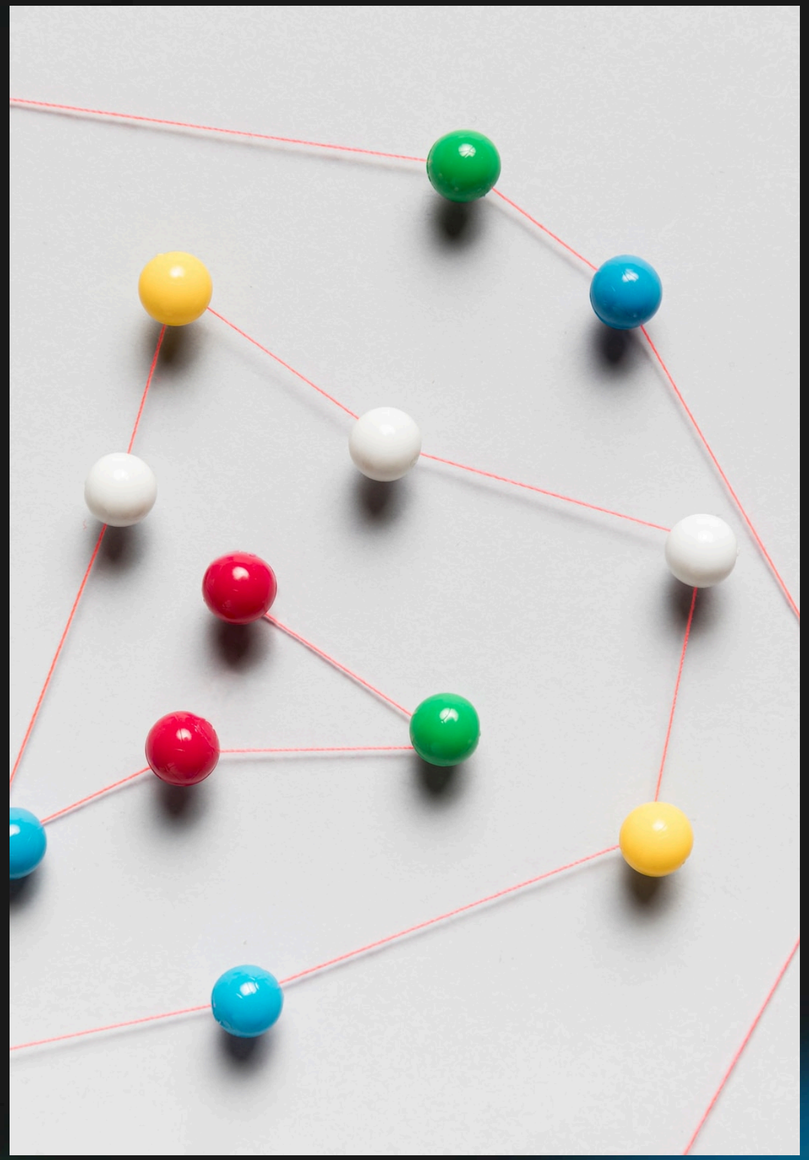
2.Branching & Merging

3.History and change Tracking

4. Open Source and Widely Adopted

5.Branching and Merging

6.Collaboration and Conflict Resolution



LAB-1: CREATING EC2 INSTANCE

1.LAUNCH THE INSTANCE WITH THE AMAZON LINUX 2 AMI.

2.CONNECT THE SERVER WITH THE GITBASH

```
client_loop: send disconnect: Connection reset by peer

ASUSN@king MINGW64 ~/desktop/aws
$ ssh -i pinku.pem ec2-user@3.92.20.15
The authenticity of host '3.92.20.15 (3.92.20.15)' can't be established.
ED25519 key fingerprint is SHA256:VO9IMxnvN1gXNLYUGsjYCgtr3ppANOxxqr/d2Y7bdkg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Could not create directory '/c/Users/ASUS?/.ssh' (No such file or directory).
Failed to add the host to the list of known hosts (/c/Users/ASUS?/.ssh/known_hosts).
Last login: Fri Nov  1 05:46:51 2024 from 49.206.54.95
       #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2025-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

9 package(s) needed for security, out of 12 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-24-229 ~]$ sudo su -
Last login: Fri Nov  1 05:46:56 UTC 2024 on pts/0
[root@ip-172-31-24-229 ~]# mkdir pinku
[root@ip-172-31-24-229 ~]# cd pinku
[root@ip-172-31-24-229 pinku]# git https://github.com/Pravalika-27/miniproject-lab-3.git
git: 'https://github.com/Pravalika-27/miniproject-lab-3.git' is not a git command. See 'git --help'.
[root@ip-172-31-24-229 pinku]# ls -a
.  ..
[root@ip-172-31-24-229 pinku]# ls
[root@ip-172-31-24-229 pinku]# ll
total 0
[root@ip-172-31-24-229 pinku]# git clone  https://github.com/Pravalika-27/miniproject-lab-3.git
Cloning into 'miniproject-lab-3'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
[root@ip-172-31-24-229 pinku]# ls -a
.  ..  miniproject-lab-3
[root@ip-172-31-24-229 pinku]# touch aws{1..5}
[root@ip-172-31-24-229 pinku]# ll
total 0
-rw-r--r-- 1 root root  0 Nov  1 06:05 aws1
-rw-r--r-- 1 root root  0 Nov  1 06:05 aws2
-rw-r--r-- 1 root root  0 Nov  1 06:05 aws3
-rw-r--r-- 1 root root  0 Nov  1 06:05 aws4
```

LAB-2: CREATE REPO IN THE LOCAL MACHINE

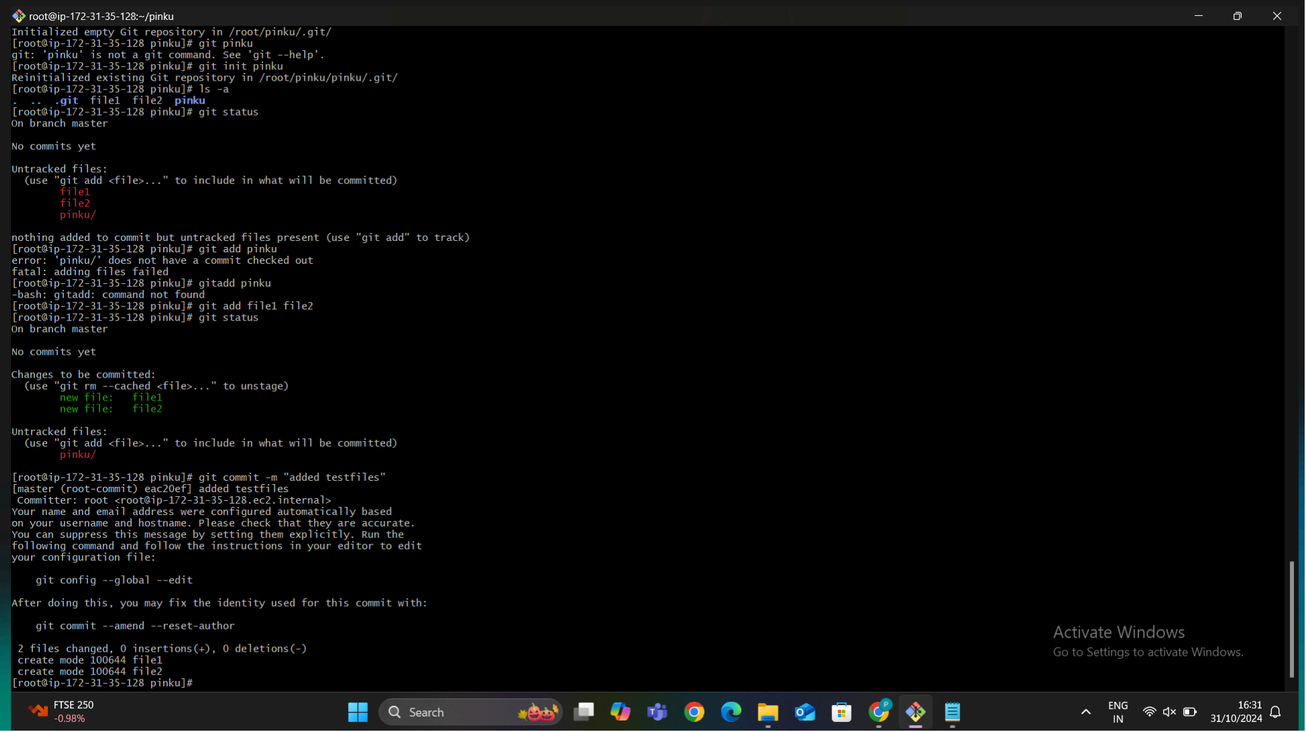
1.Create and Initialize Repository:

2.Check Status

3.Add a File

4.Stage and Commit

5.Final Status Check



LAB-3: CREATING REPO IN REMOTE LOCATION

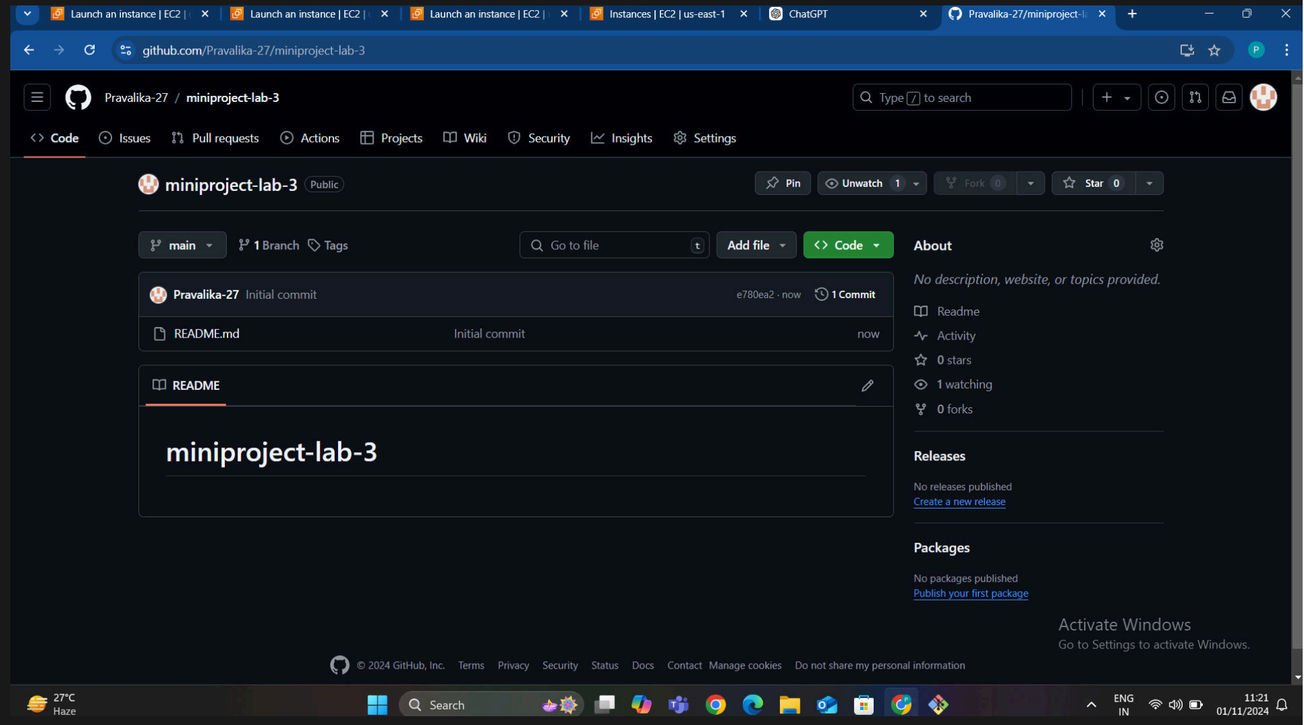
Create a Repository on GitHub

1.Go to your GitHub account.

2.Create a New Repository

3.Add README.md

4.Create Repository



LAB-4: WORKING WITH REMOTE REPO

1.Clone Repo :git clone <repo_url>

2.Make Changes & Commit

3.Create Personal Access Token

4.Clone Again with Token

5.Push Changes

6.verify your github repo

LAB-5: PUSHING A LOCALLY CREATED REPO TO GITHUB

1.Create Repo Locally

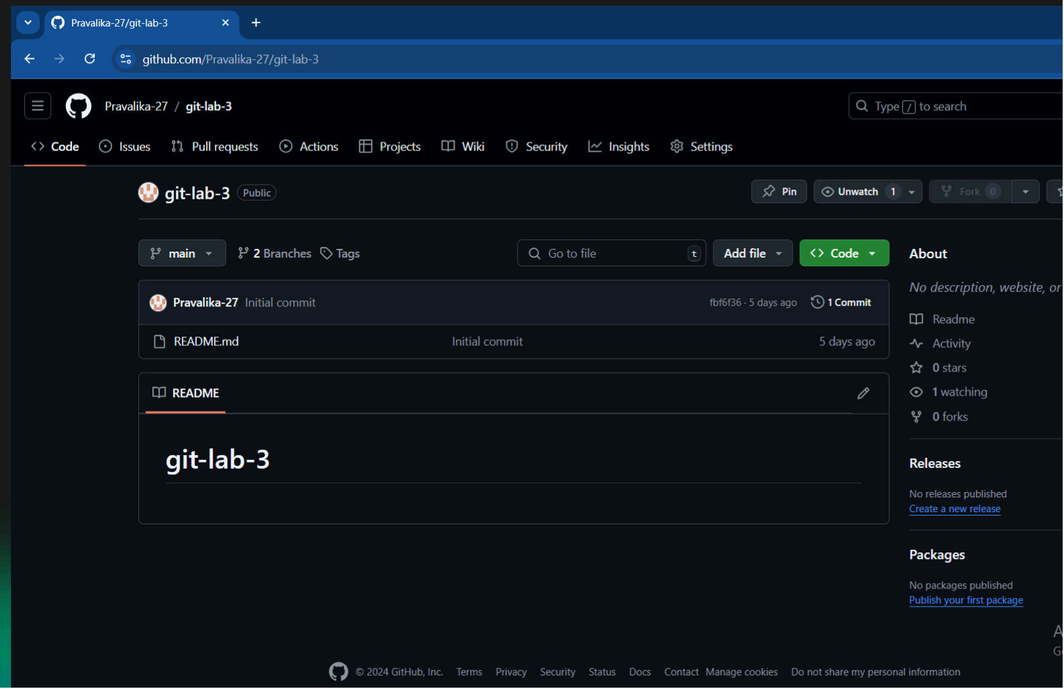
2.Create Same Repo on GitHub

3.Link Local Repo to GitHub

i.(Rename branch to "main")

ii.git remote add origin <repo_URL>

4.Push to GitHub:git push -u origin main



LAB-6: CREATING A NEW BRANCH FROM YOUR MAIN BRANCH

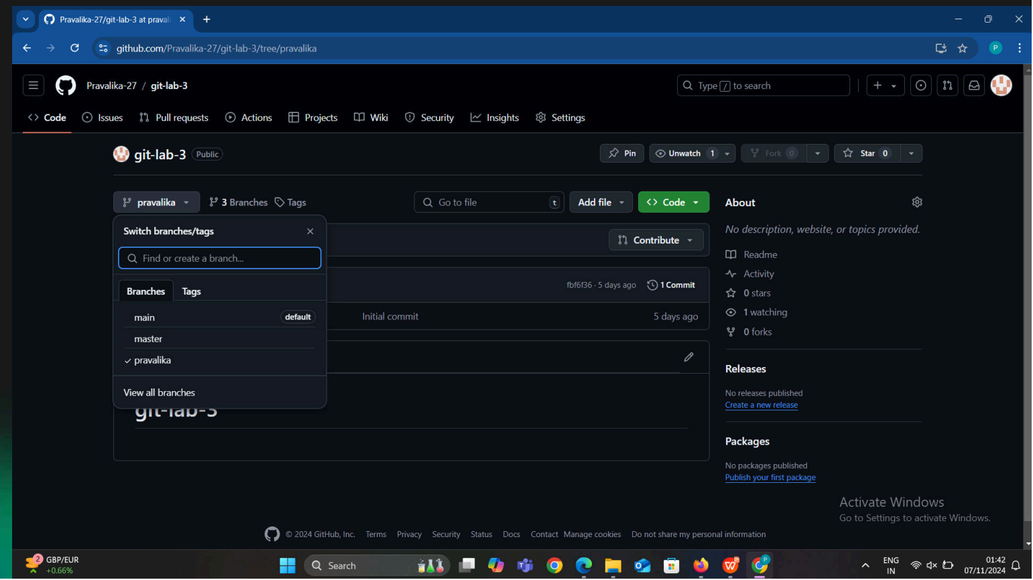
1.Go to GitHub Repository

2.Create a New Branch

3.Make Changes in New Branch

4.Here the new changes are made in new branch

5.Will not affect to the main branch



LAB-7: PULL ALL THE BRANCHES IN YOUR LOCAL MACHINE

1 Go to the local machine

2.Pull the Repo

3.Check Branch & Switch to New Branch

4.Make Changes:create files

5.Push Changes to Remote

6.No Effect on Main Branch

LAB-8: MERGE OUR FEATURE BRANCH WITH MAIN BRANCH

1.Go to the GitHub repo and verify the new branch.

2.Create Pull Request

3.Put the main branch and feature branch in the request blocks.

4.Go to the pull request tab click on the pull request are available.

5.Review and Merge

6.Verify in Main Branch

LAB-9: GO TO THE LOCAL MACHINE

1.Go to your local machine where the copy of remote repo

2.Checkout to the main branch:Git checkout<branch name>

3.Now run the command “git pull” to all the new changes such as branches from remote location

4.Here the new changes are only available in main branch.


Thank You