

**SYSTEM PROVISIONING AND CONFIGURATION MANAGEMENT**

**LAB**

YEAR: 2024

NAME: Aastha Mishra

SAP ID: 500094912

ROLL NO: R2142210013

COURSE: BTech CSE (DevOps)

SEMESTER: Sixth (B3)

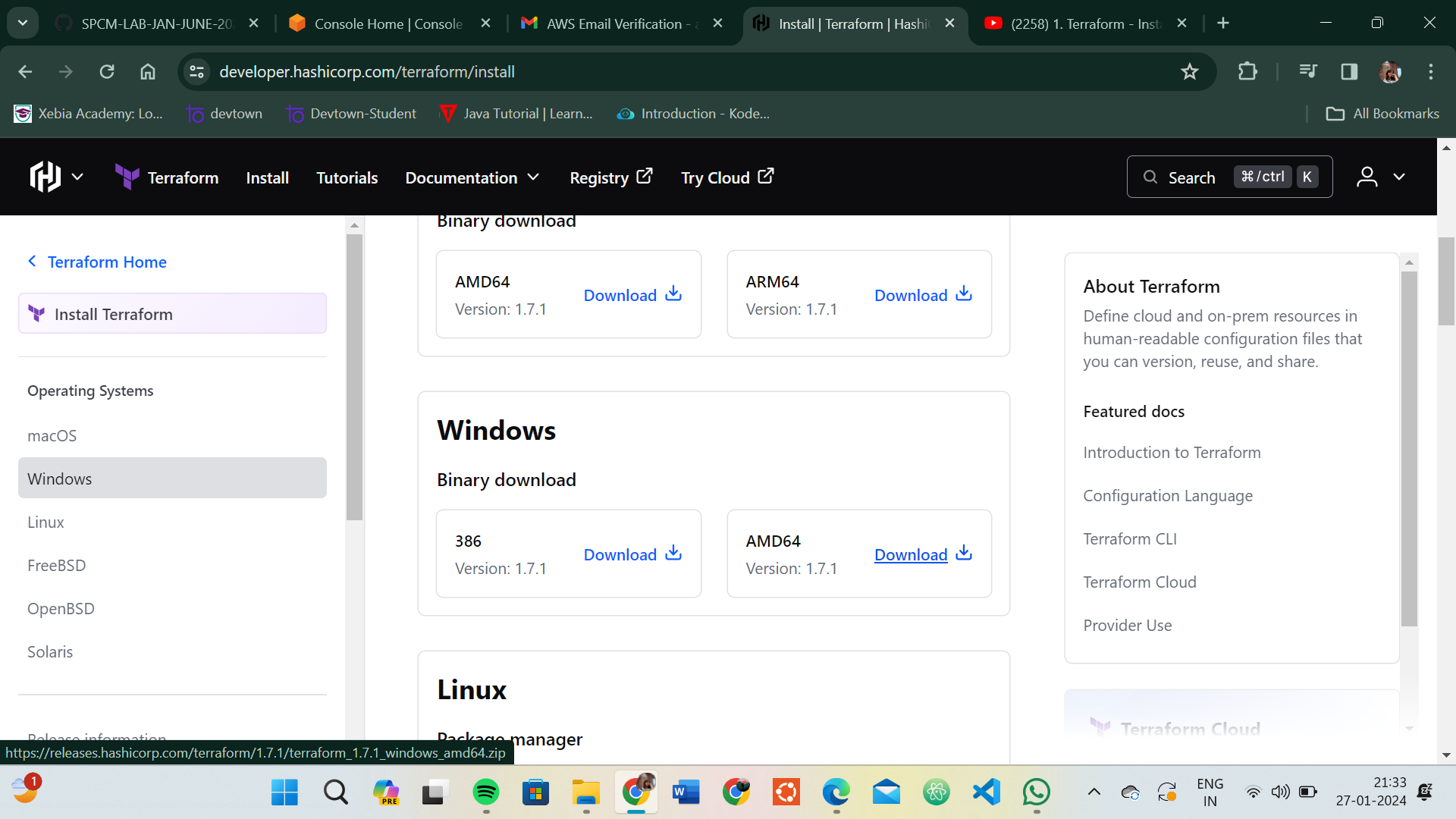
FACULTY NAME: Dr.Hitesh Kumar Sharma

Lab Exercise 1– Install Terraform on Windows

# Download Terraform File for Windows

To find the latest version of Terraform for Windows:

1. Browse to the [Download Terraform](https://developer.hashicorp.com/terraform/downloads) page.
2. Select the Windows tab under the **Operating System** heading. The latest version is preselected.

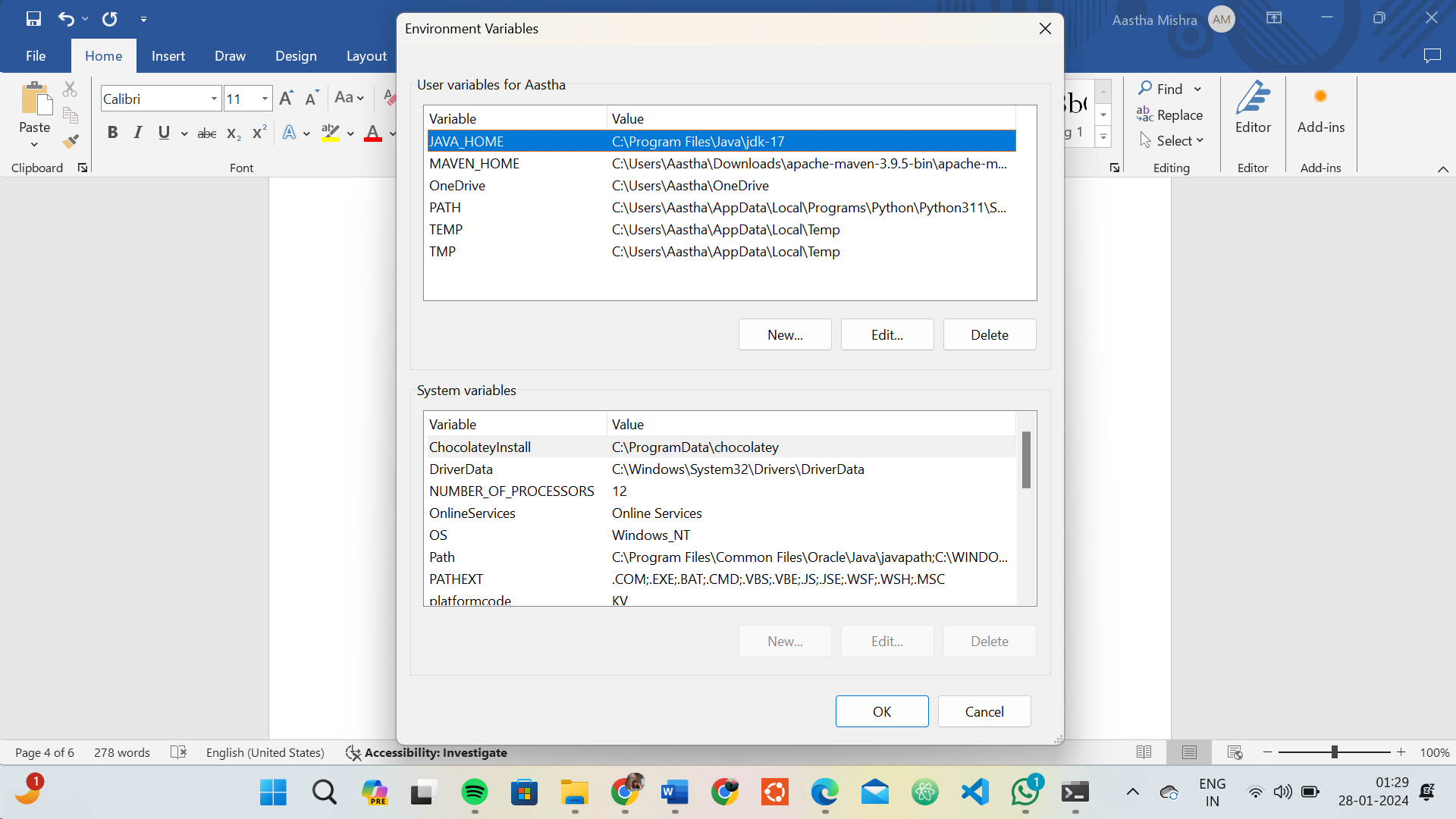


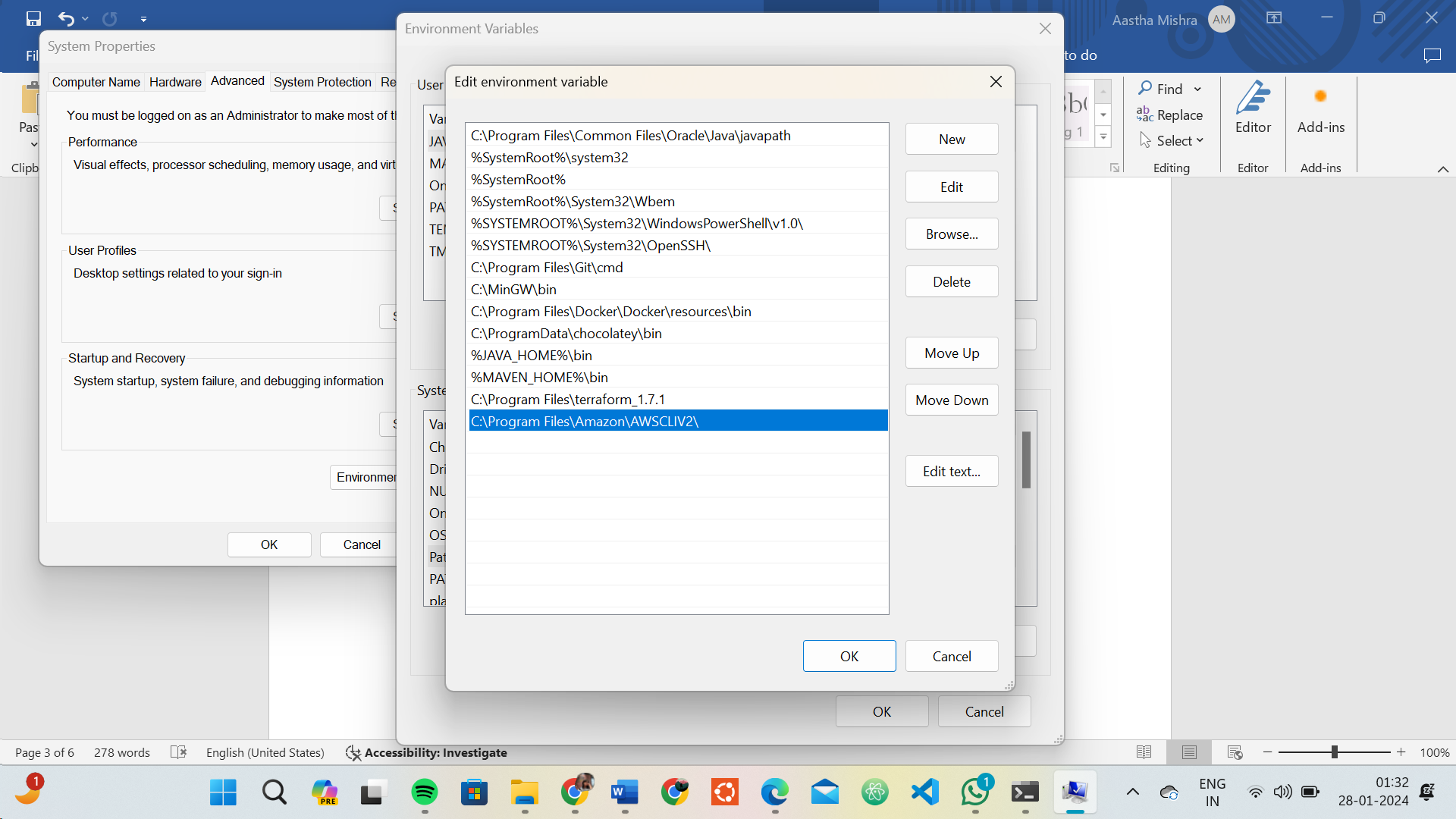
1. Choose the binary to download. Select 386 for 32-bit systems or [AMD64](https://phoenixnap.com/glossary/amd64) for 64-bit systems. Choose the download location for the zip file if the download does not start automatically.
2. Unzip the downloaded file. For example, use the *C:\terraform* path. Remember this location so you can add the path to the environment variables.

# Add Terraform Path to System Environment Variables

To add the Terraform executable to the system's global path:

1. Open the start menu, start typing *environment* and click **Edit system environment variables**. The System Properties window opens.
2. Click the **Environment Variables...** button.

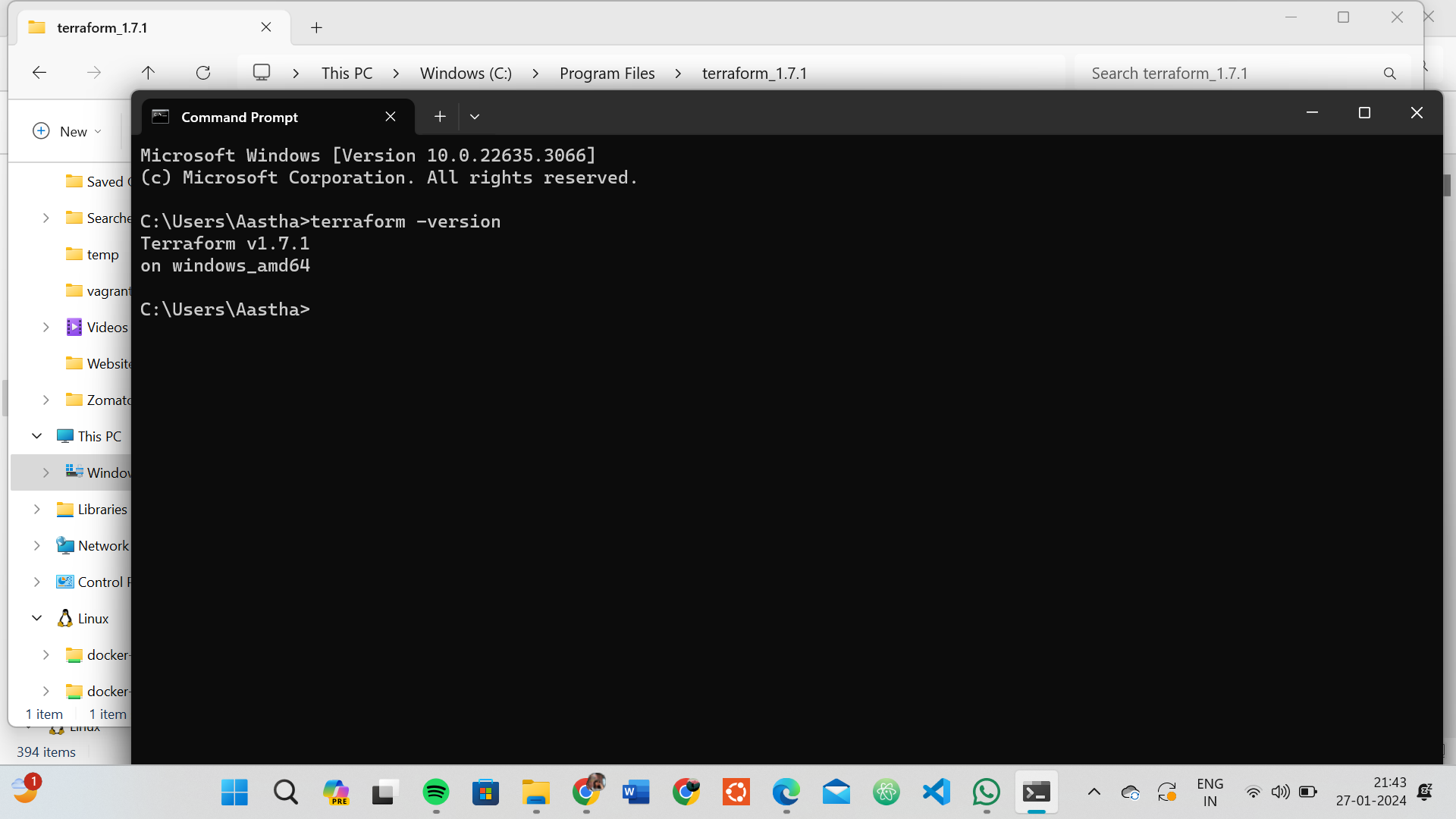


1. Select the **Path variable** in the System variables section to add terraform for all accounts. Alternatively, select **Path** in the **User variables** section to add terraform for the currently logged-in user only. Click **Edit** once you select a Path**.**
2. Click **New** in the edit window and enter the location of the Terraform folder
3. Click **OK** on all windows to apply the changes.

# Verify Windows Terraform Installation

To check the Terraform global path configuration:

1. Open a new command-prompt window.
2. Enter the command to check the Terraform version: terraform -version

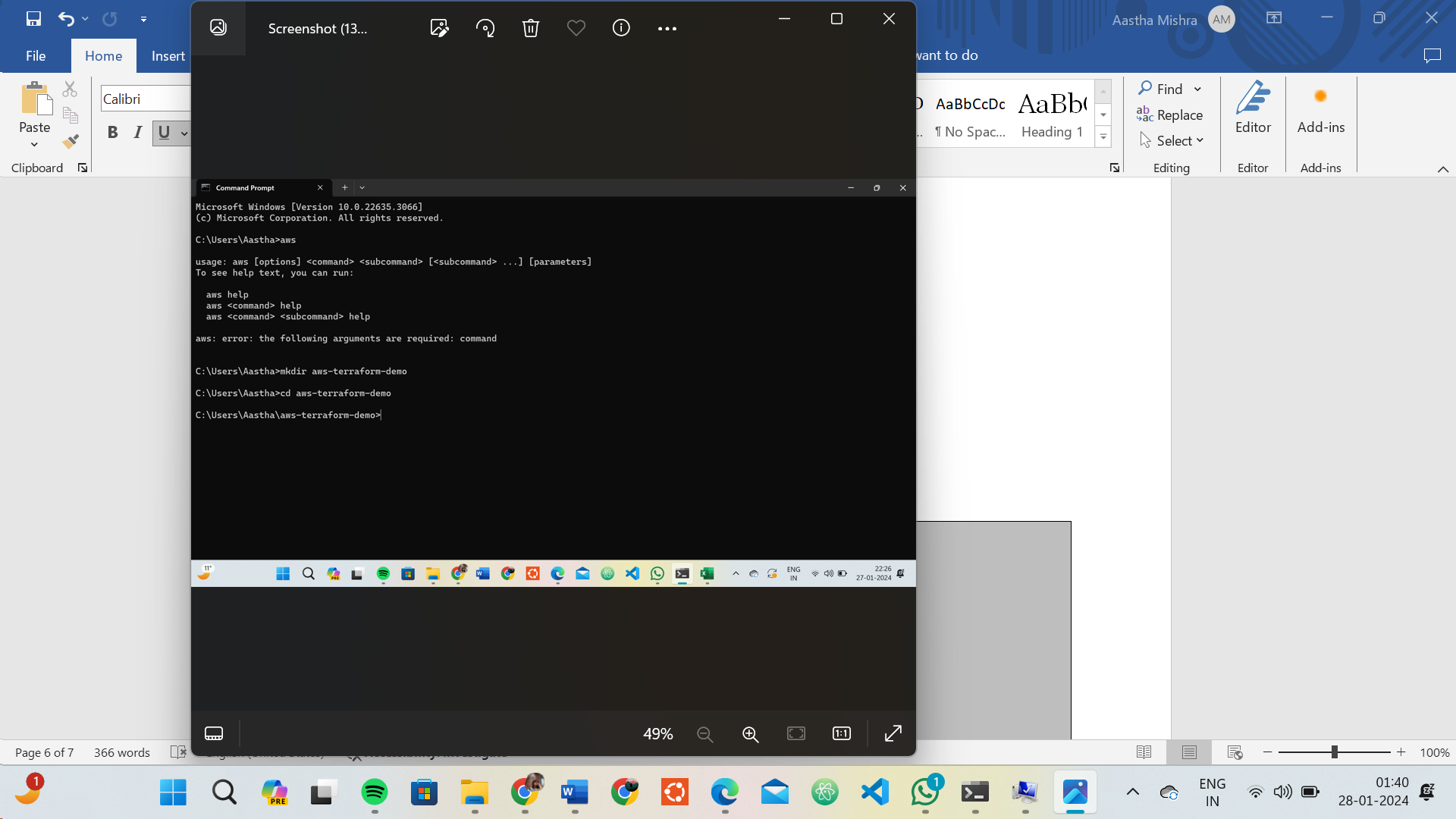


Lab Exercise 2– Terraform AWS Provider and IAM User Setting

Exercise Steps:

Step 1: Create a New Directory:

Create a new directory for your Terraform configuration:



Step 2: Create Terraform Configuration File (main.tf):

Create a file named main.tf with the following content:

terraform { required\_providers { aws = {

source = "hashicorp/aws" version = "5.31.0"

**}**

**}**

**}**

provider "aws" {

region = "ap-south-1"

access\_key = "your IAM access key" secret\_key = "your secret access key"

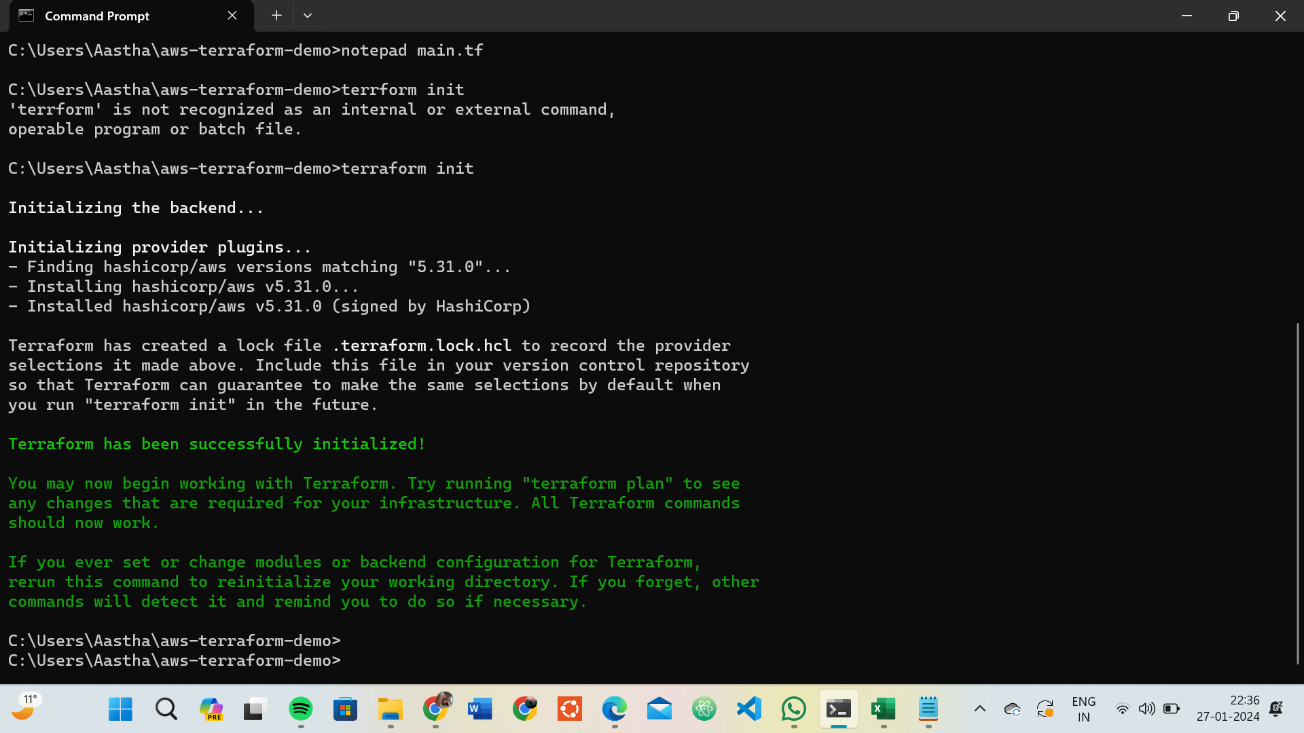
**}**

This script defines an AWS provider and provisions an EC2 instance.

Step 3: Initialize Terraform:

Run the following command to initialize your Terraform working directory:

terraform init



Lab Exercise 3–Provisioning an EC2 Instance on AWS

# **Exercise Steps:**

**Step 1: Create a New Directory:**

Create a new directory for your Terraform configuration:

**mkdir aws-terraform-demo**

**cd aws-terraform-demo**

# Step 2: Create Terraform Configuration File (main.tf):

Create a file named main.tf with the following content:

**terraform { required\_providers { aws = {**

**source = "hashicorp/aws" version = "5.31.0"**

**}**

**}**

**}**

**provider "aws" {**

**region = "ap-south-1"**

**access\_key = "your IAM access key" secret\_key = "your secret access key"**

**}**

This script defines an AWS provider and provisions an EC2 instance.

# Step 3: Initialize Terraform:

Run the following command to initialize your Terraform working directory:

**terraform init**

# Step 4: Create Terraform Configuration File for EC2 instance (instance.tf):

Create a file named instnace.tf with the following content:

**resource "aws\_instance" "My-instance" { instance\_type = "t2.micro"**

**ami = "ami-03f4878755434977f"**

**count = 1 tags = {**

**Name = "UPES-EC2-Instnace"**

**}**

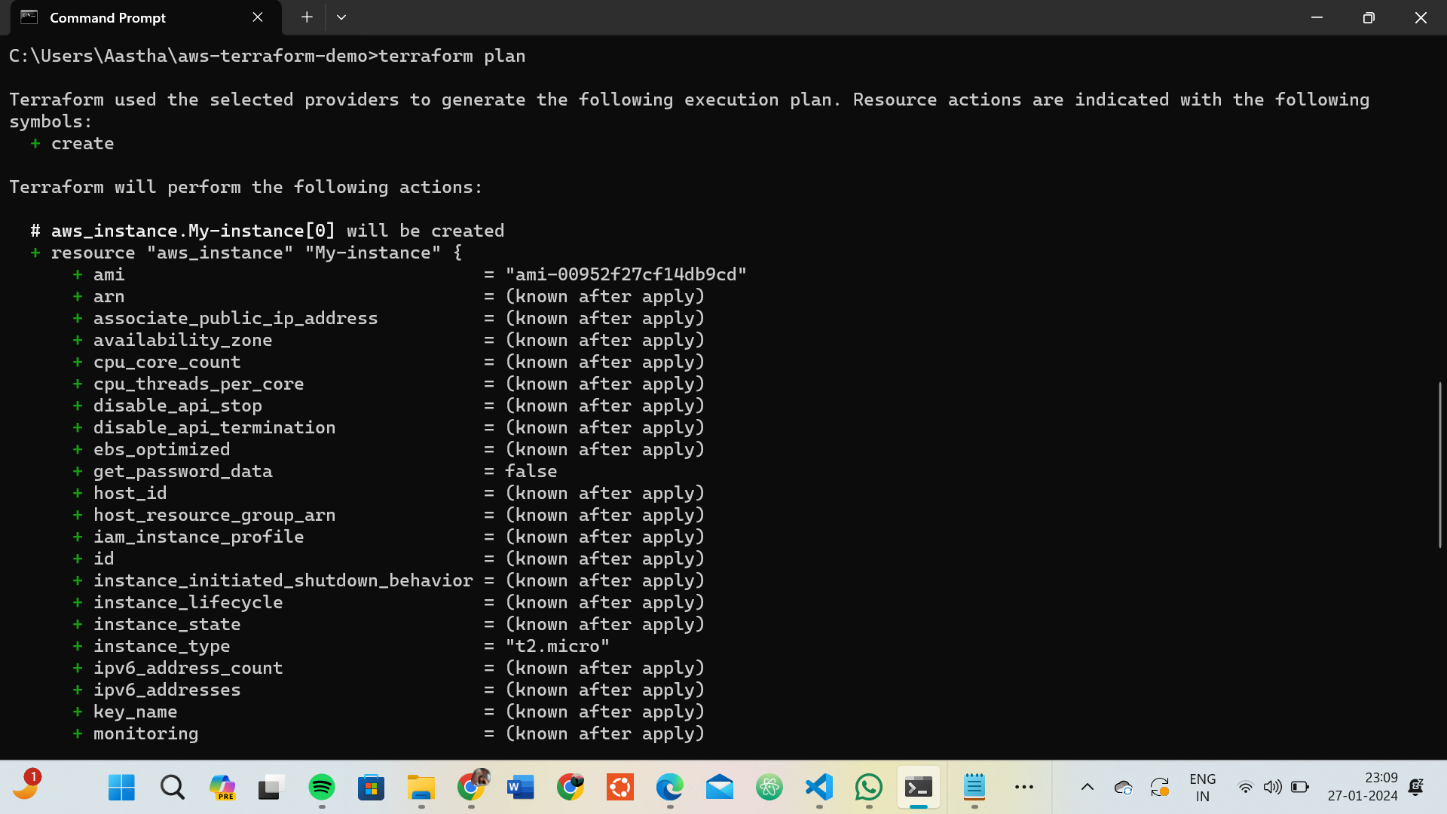
**}**

# Step 5: Review Plan:

Run the following command to see what Terraform will do:

**terraform plan**

Review the plan to ensure it aligns with your expectations.

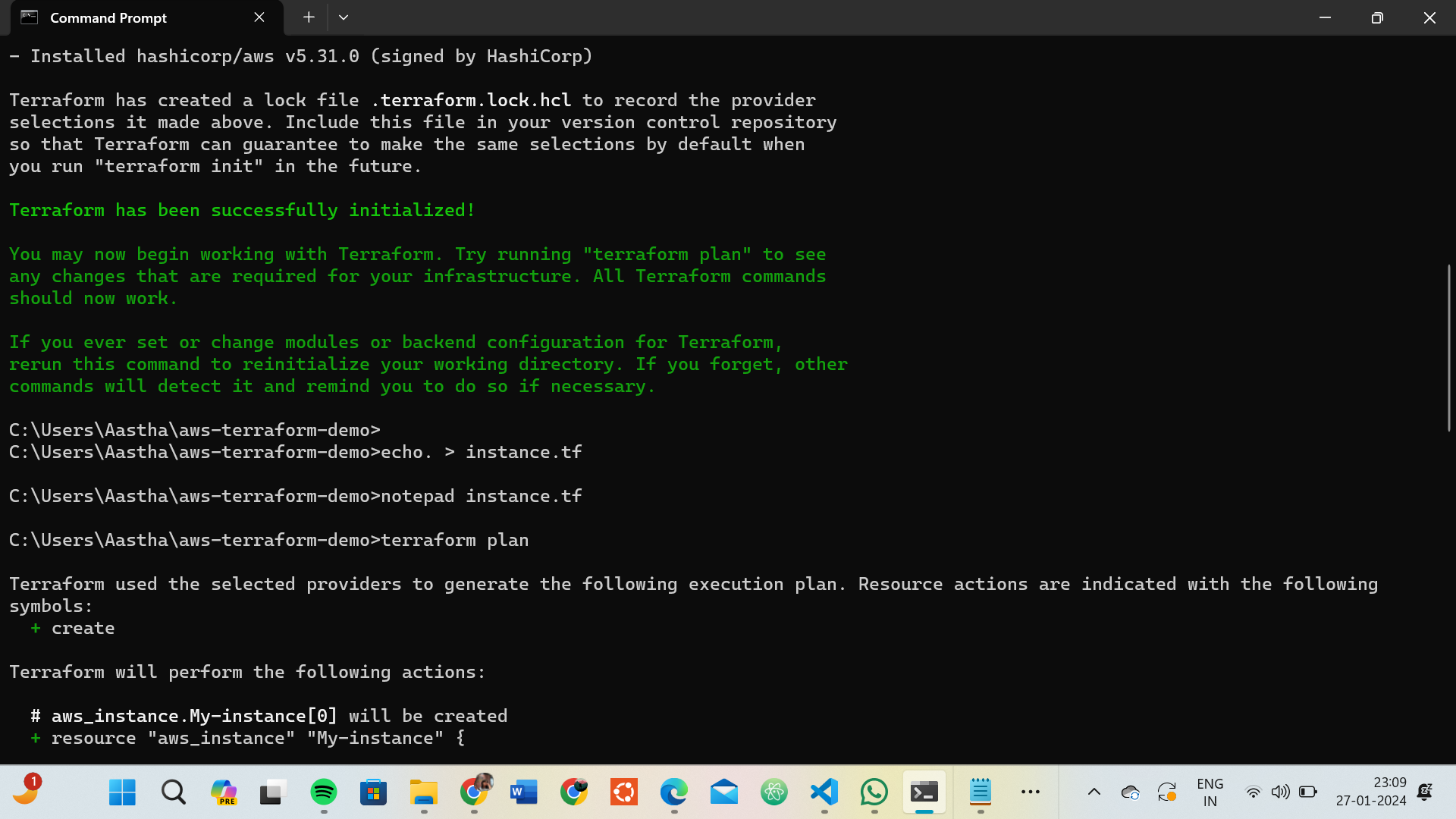


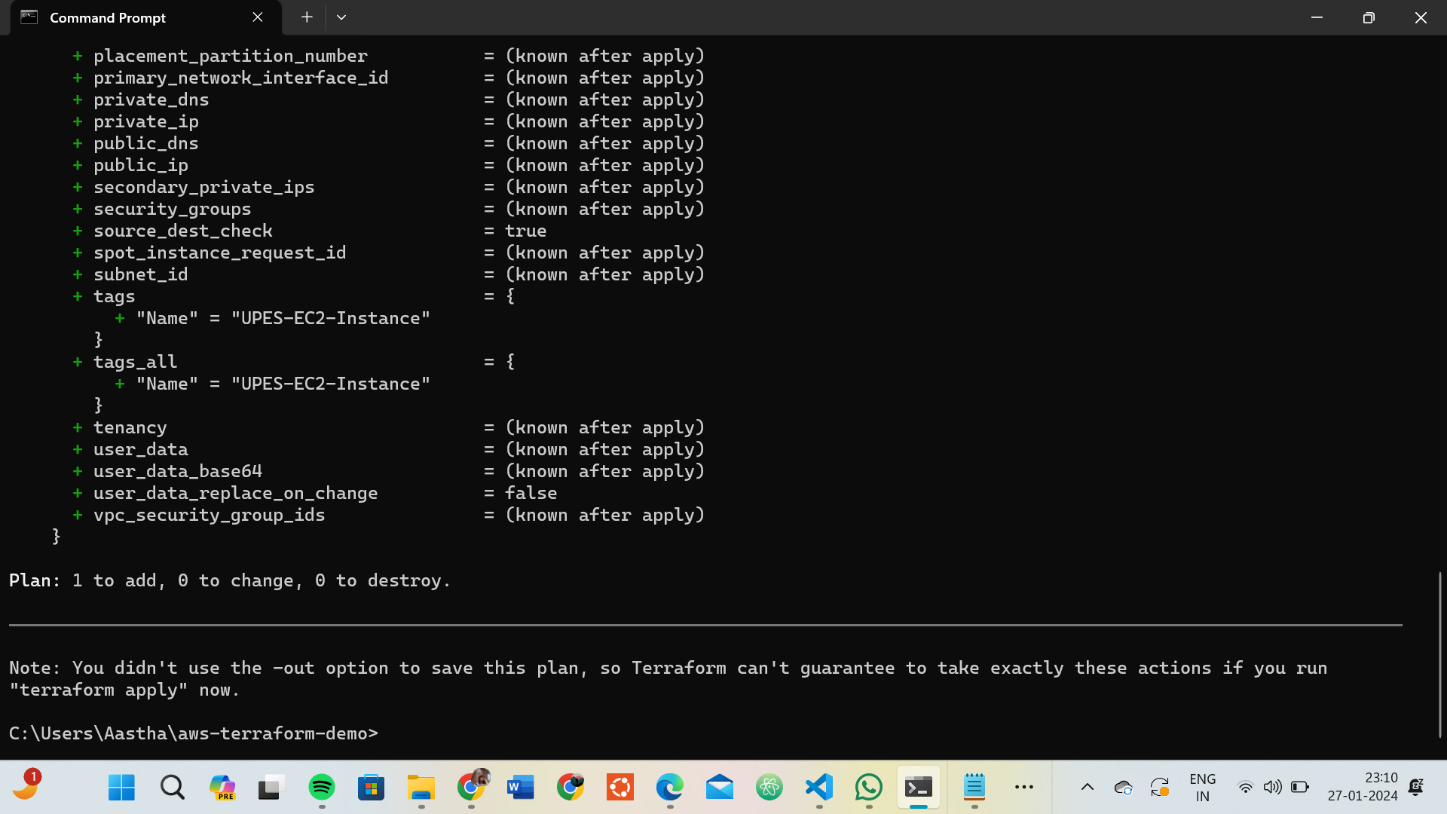
# Step 6: Apply Changes:

Apply the changes to create the AWS resources:

**terraform apply**

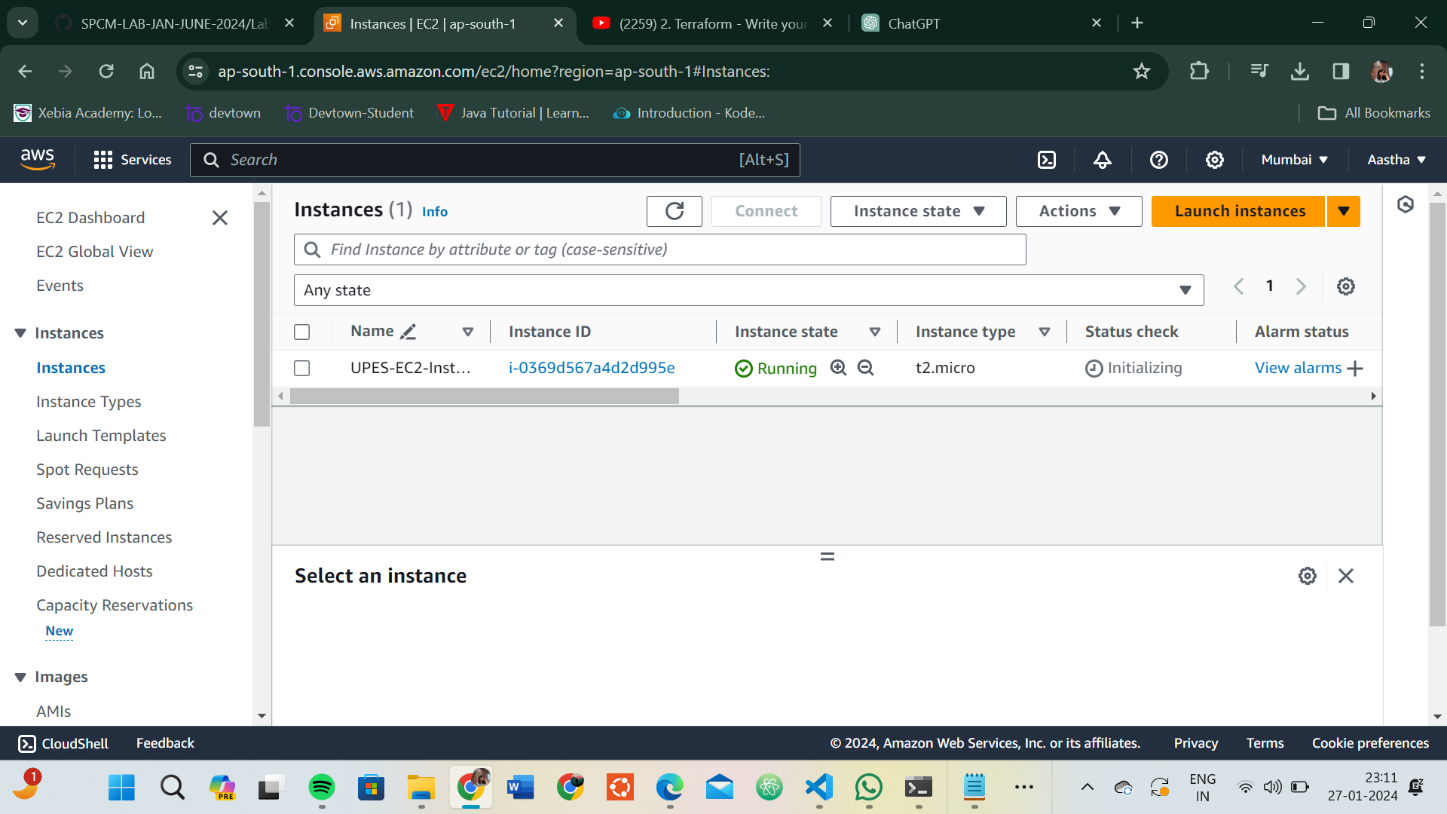
Type yes when prompted.





# Step 7: Verify Resources:

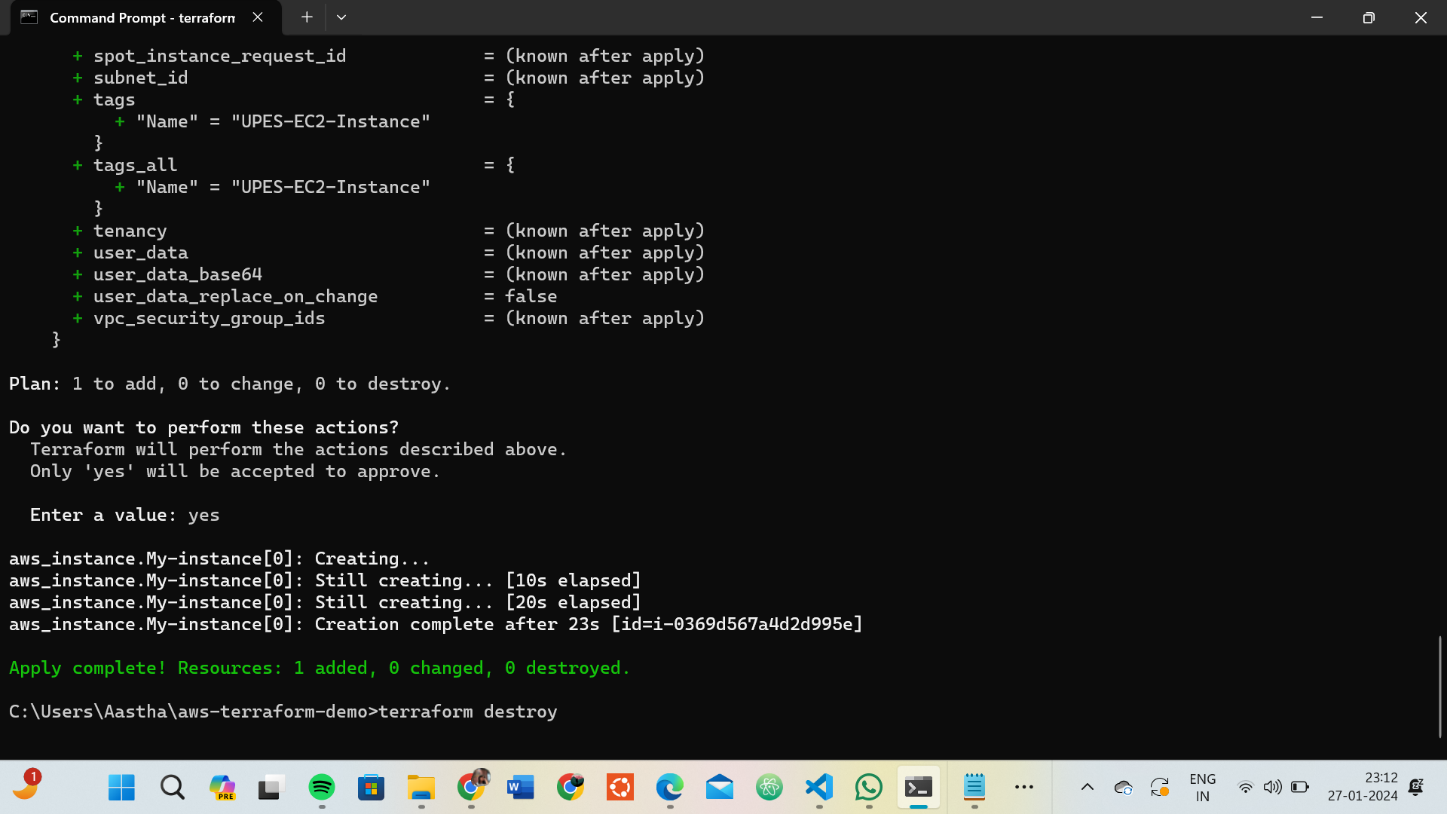
After the terraform apply command completes, log in to your AWS Management Console and navigate to the EC2 dashboard. Verify that the EC2 instance has been created.



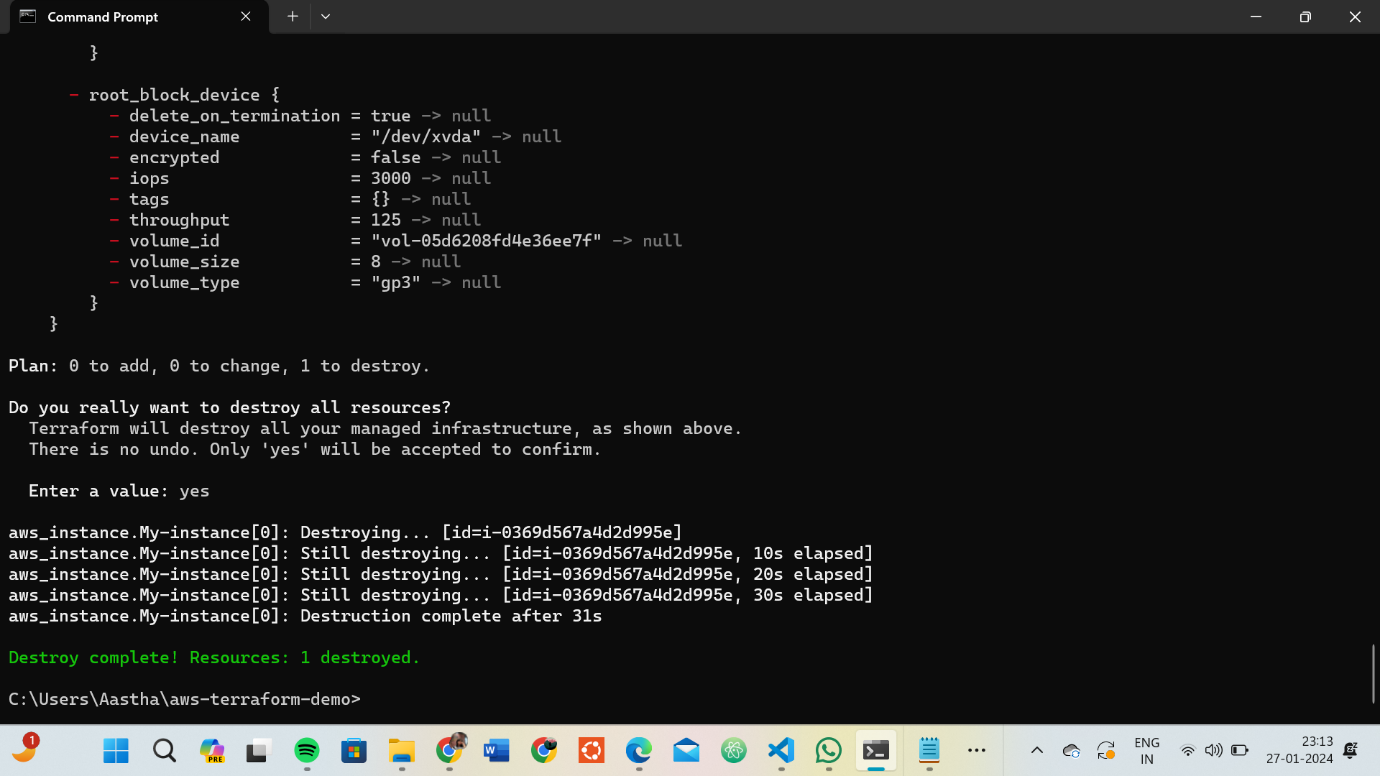
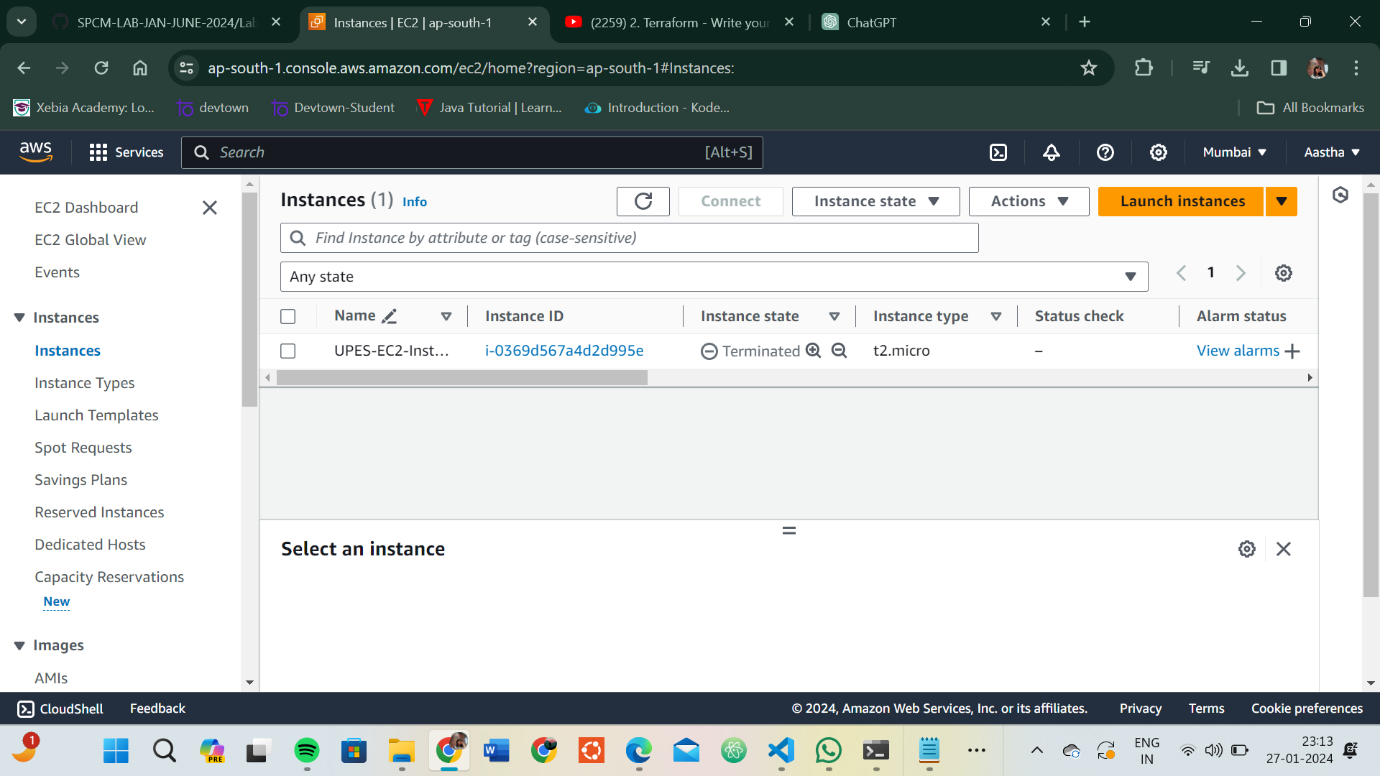
# Step 8: Cleanup Resources:

When you are done experimenting, run the following command to destroy the created resources:

**terraform destroy**

Type yes when prompted.





Lab Exercise 4– Terraform Variables

# Steps:

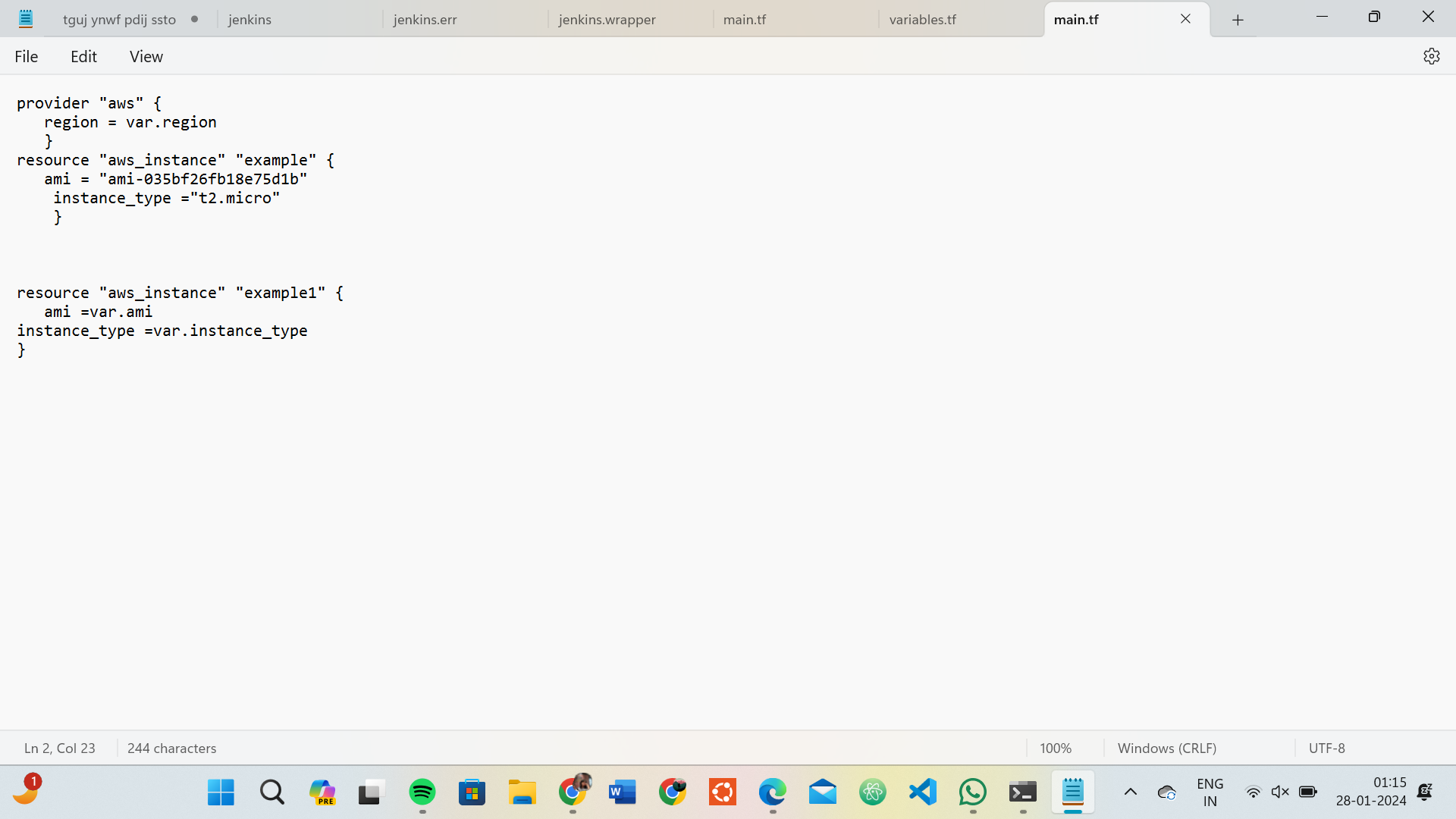
1. **Create a Terraform Directory:**
   * Create a new directory for your Terraform project.

**mkdir terraform-variables cd terraform-variables**

# Create a Terraform Configuration File:

* + Create a file named main.tf within your project directory.

**# main.tf**



# Define Variables:

* + Open a new file named variables.tf. Define variables for region, ami, and instance\_type.

**# variables.tf**

****

# Use Variables in main.tf:

* + Modify main.tf to use the variables.

**# main.tf**

**provider "aws" { region = var.region**

**}**

**resource "aws\_instance" "example" { ami = var.ami**

**instance\_type = var.instance\_type**

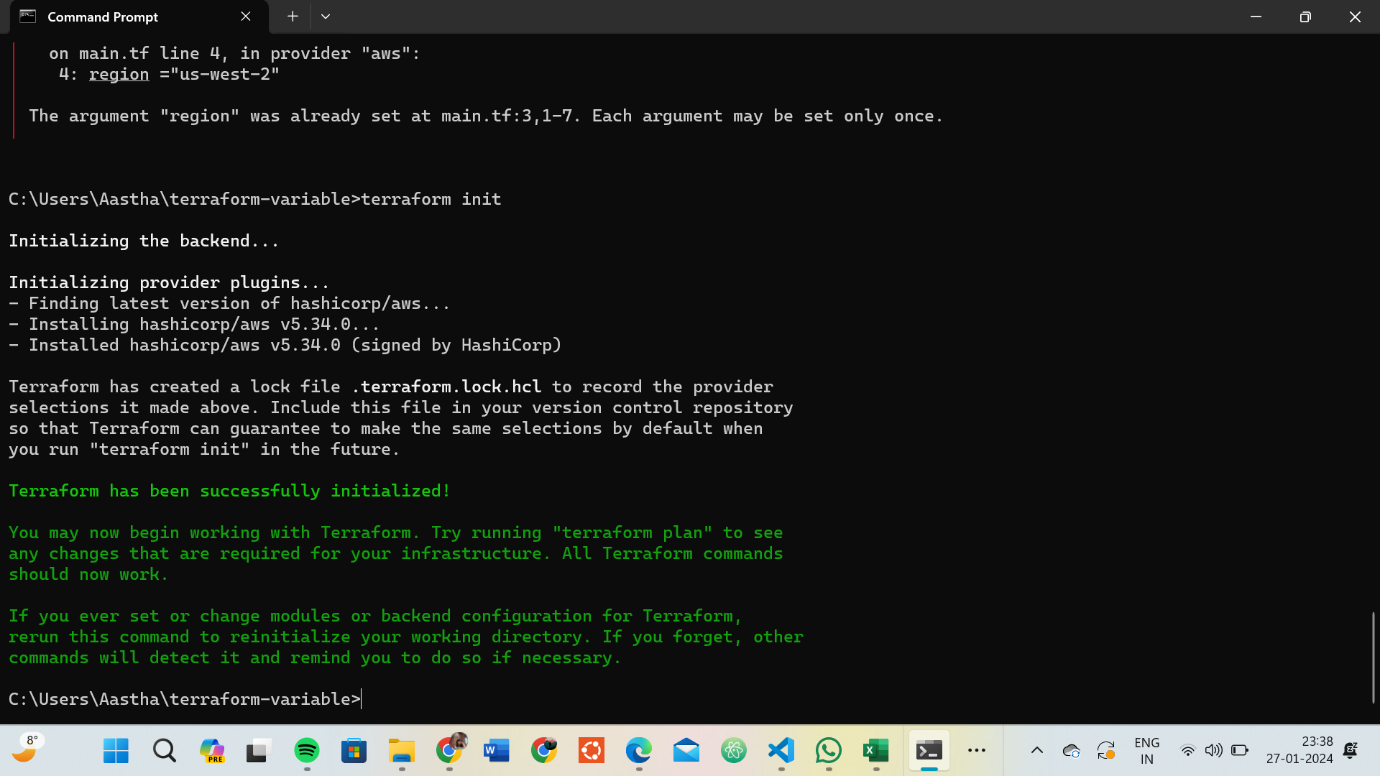
**}**

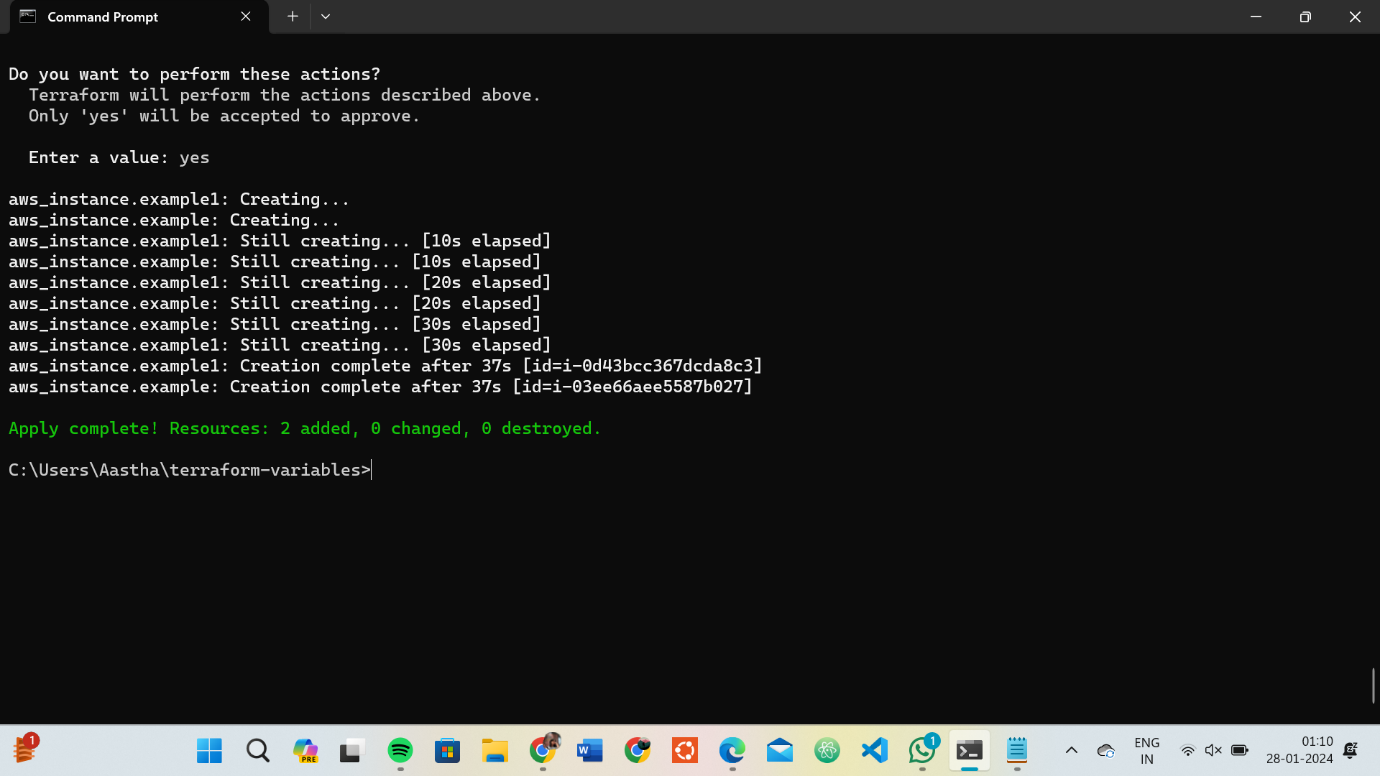
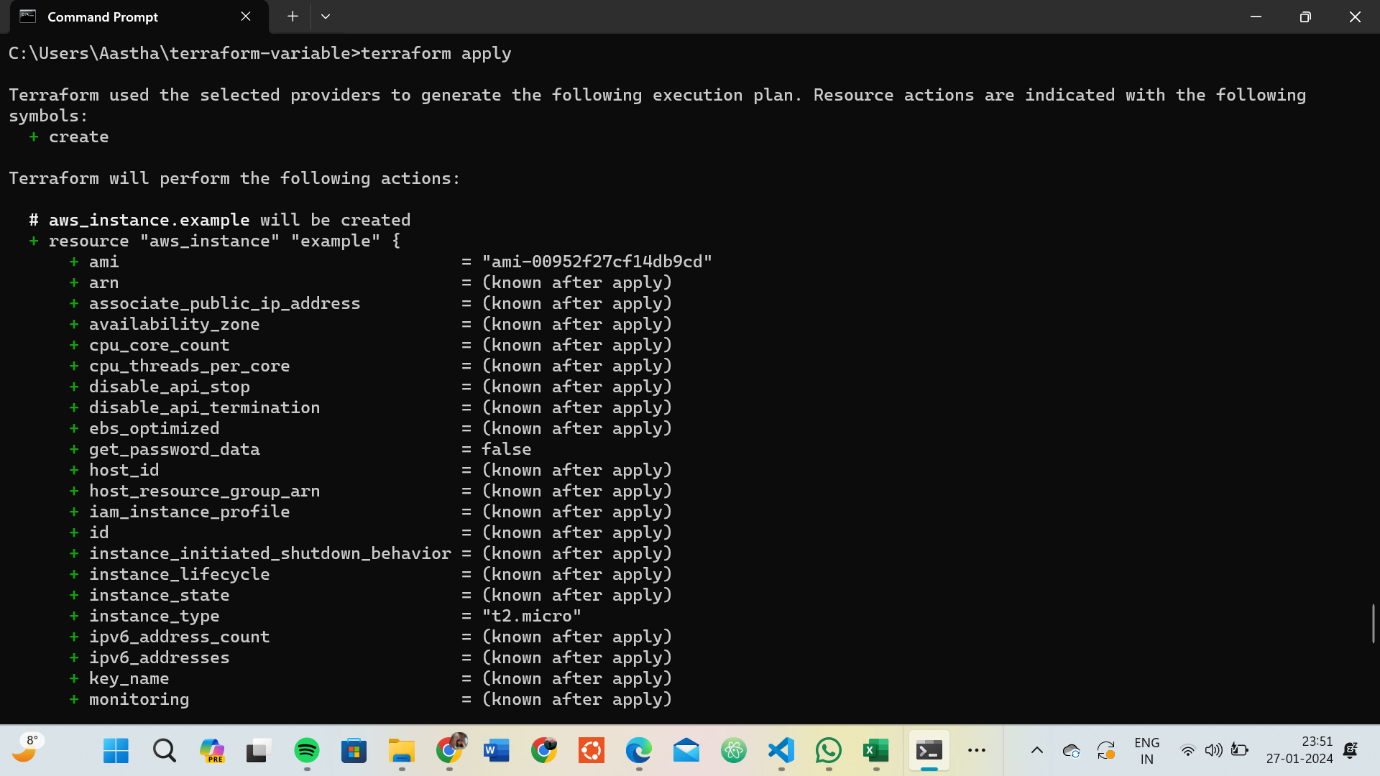
# Initialize and Apply:

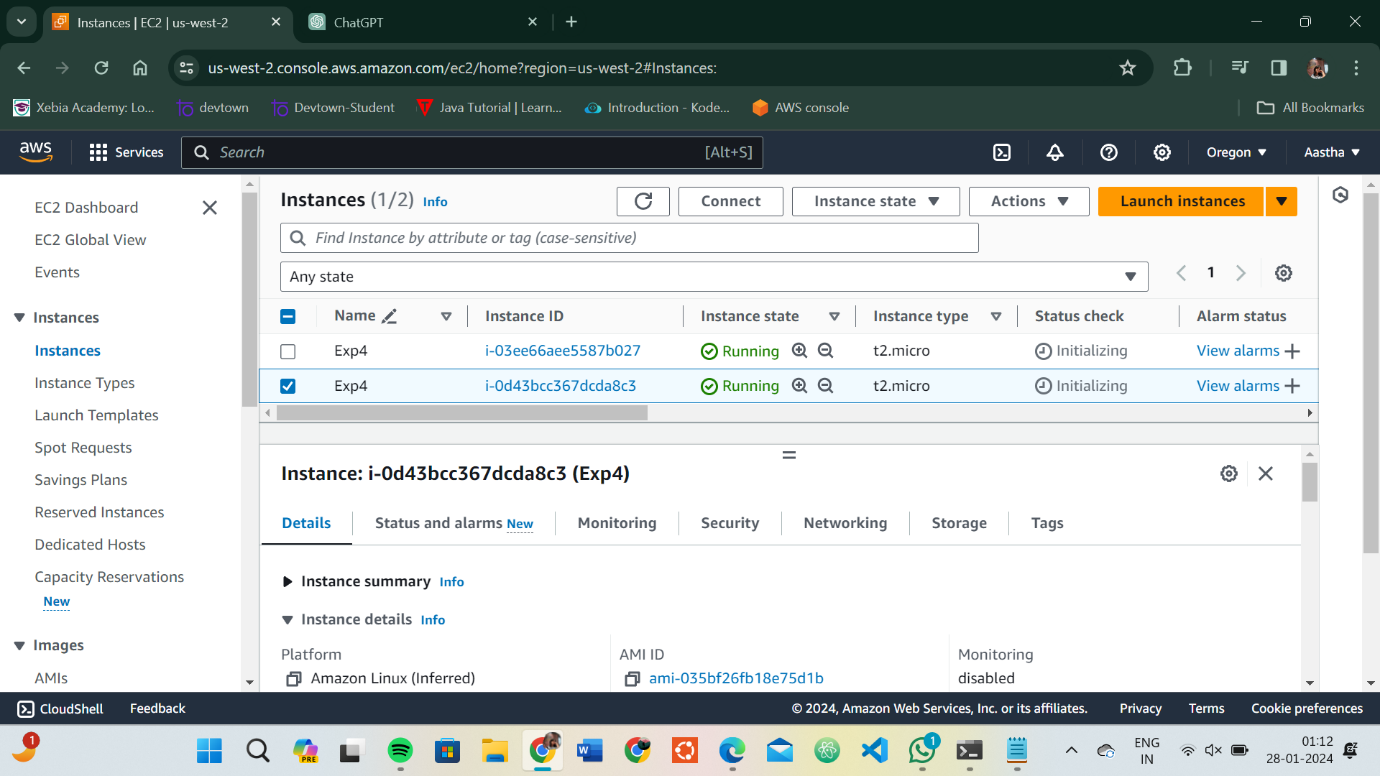
Run the following Terraform commands to initialize and apply the configuration.

**terraform init terraform apply**

Observe how the region changes based on the variable override.







# Clean Up:

After testing, you can clean up resources.

**terraform destroy**

Confirm the destruction by typing yes.

