**Terraform Task -1**

**Q.1** **Make a note on:**

* 1. What is Terraform?
  2. Why Terraform?
  3. Benefits of Terraform

***Ans:-***

**Terraform** is an open-source infrastructure as code (IaC) software tool developed by HashiCorp. It allows users to define, provision, and manage infrastructure in a consistent, repeatable, and automated manner. Here’s a closer look at what Terraform is and how it functions:

* **Infrastructure as Code (IaC):** Terraform uses code to manage and provision all types of infrastructure, including cloud services (AWS, Azure, Google Cloud), on-premises environments, and SaaS/PaaS offerings.
* **Configuration Language:** It utilizes the HashiCorp Configuration Language (HCL) or JSON to describe the desired state of infrastructure components.
* **Resource Management:** Terraform can manage a wide range of external resources such as servers, databases, networking, and more through its configuration files.
* **State Tracking:** It keeps a state file to track the current state of the infrastructure, ensuring it only applies the necessary changes to achieve the desired configuration.

**Why Use Terraform?**

Terraform offers several compelling advantages for managing and provisioning infrastructure:

1. **Consistency and Reliability:**
   * **Code-Based Management:** Infrastructure as code ensures that infrastructure setup and configuration are consistent and reliable, reducing the risk of human error.
   * **Repeatability:** Code can be reused across different environments, ensuring that deployments are identical every time.
2. **Automation and Efficiency:**
   * **Automated Provisioning:** Terraform automates the deployment and management of infrastructure, saving time and reducing the need for manual intervention.
   * **Scalable Management:** It can manage complex infrastructures and scale up or down as needed, making it suitable for both small applications and large data centers.
3. **Multi-Cloud and Hybrid Support:**
   * **Unified Management:** Terraform supports multiple cloud providers and on-premises environments, allowing users to manage a hybrid or multi-cloud infrastructure from a single tool.
   * **Provider Ecosystem:** With a wide range of providers (AWS, Azure, Google Cloud, VMware, etc.), it can handle diverse infrastructure needs across various platforms.
4. **Collaboration and Version Control:**
   * **Code Sharing:** Teams can share and collaborate on infrastructure configurations through version-controlled code repositories.
   * **Auditing and Rollback:** Changes to infrastructure can be tracked, audited, and rolled back if necessary, enhancing transparency and control.
5. **Modularity and Reusability:**
   * **Modules:** Terraform’s module system allows for the creation of reusable configurations, simplifying the management of complex setups and promoting code reuse.
   * **Flexible Configuration:** Users can define reusable components that can be customized and deployed across multiple projects or environments.
6. **Planning and Visibility:**
   * **Execution Plans:** Terraform provides an execution plan (terraform plan) that previews the changes it will make before applying them, allowing users to review and approve changes.
   * **Visibility:** This plan enhances visibility and control over what changes are made to the infrastructure.

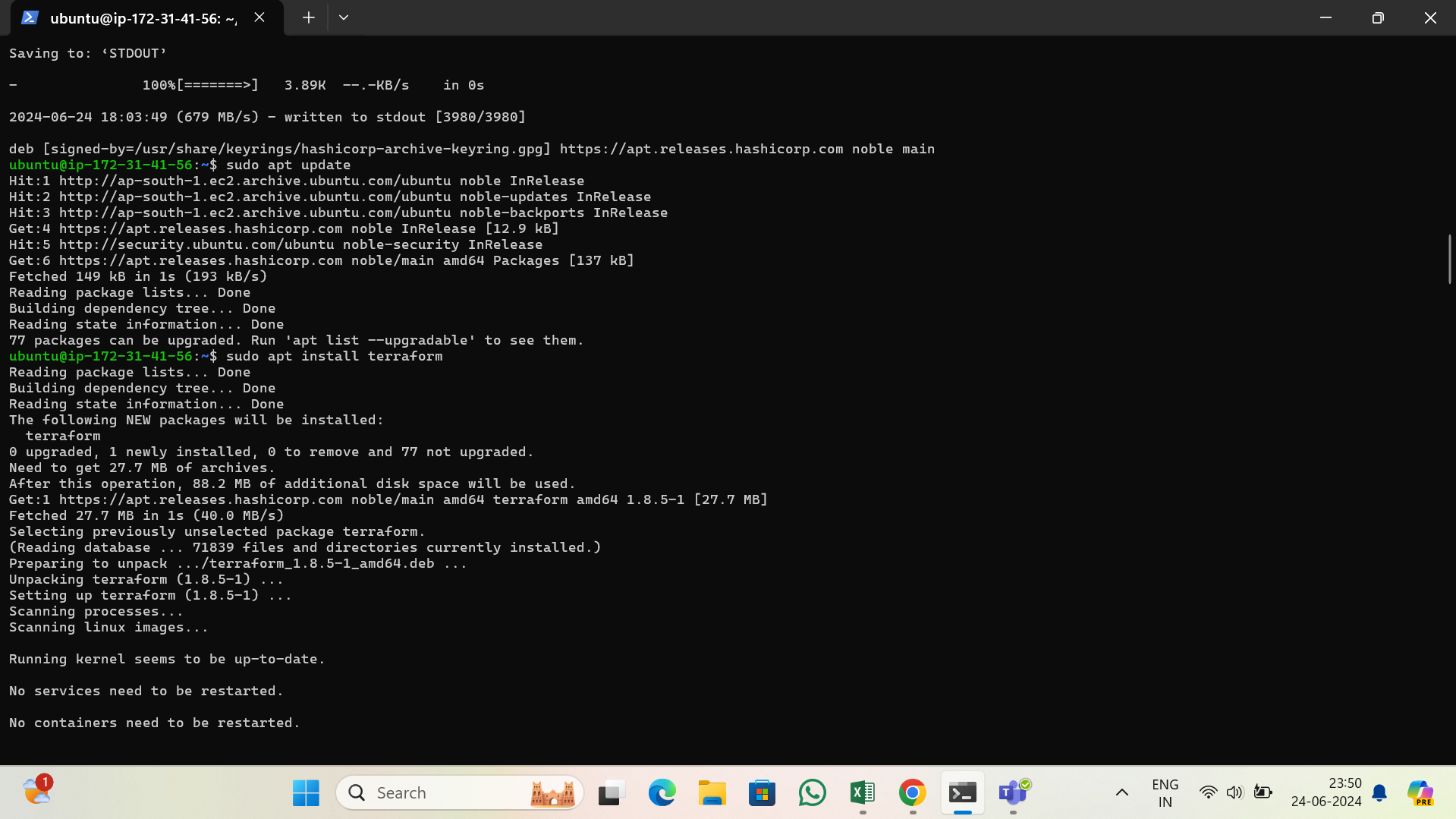
**Practical Applications:**

* **Cloud Infrastructure Deployment:** Automate the provisioning of cloud resources such as virtual machines, storage, and networks.
* **Environment Management:** Ensure consistent configuration across development, staging, and production environments.
* **CI/CD Integration:** Integrate with continuous integration and deployment pipelines to automate infrastructure changes.
* **Hybrid and Multi-Cloud Orchestration:** Manage resources seamlessly across multiple cloud providers and on-premises environments.

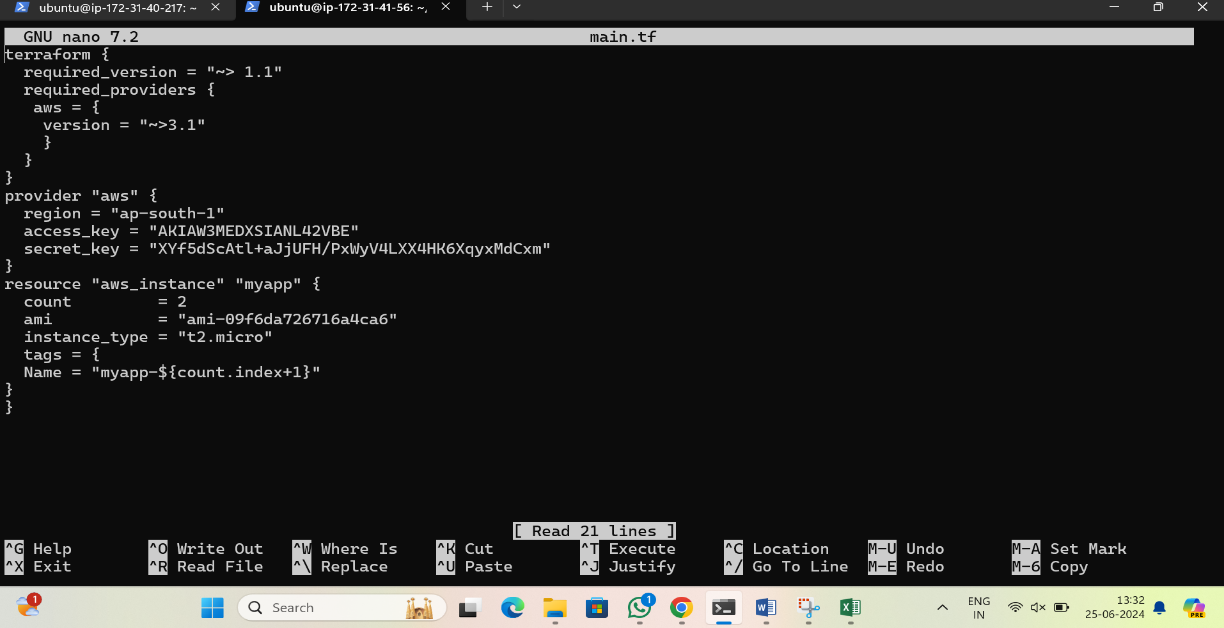
**Q.2 Launch two EC2 instances with names as “myapp-1” and “myapp-2” using Amazon-Linux OS in ‘ap-south-1’ region.**

To launch two EC2 instances named "myapp-1" and "myapp-2" using Amazon **Linux OS** in the **ap-south-1** region with Terraform, you need to create a **main.tf** configuration file with the necessary code. Here's a sample main.tf file to achieve this:

* **Terraform Installation**



* **nano main.tf**

****

**Provider Block:**

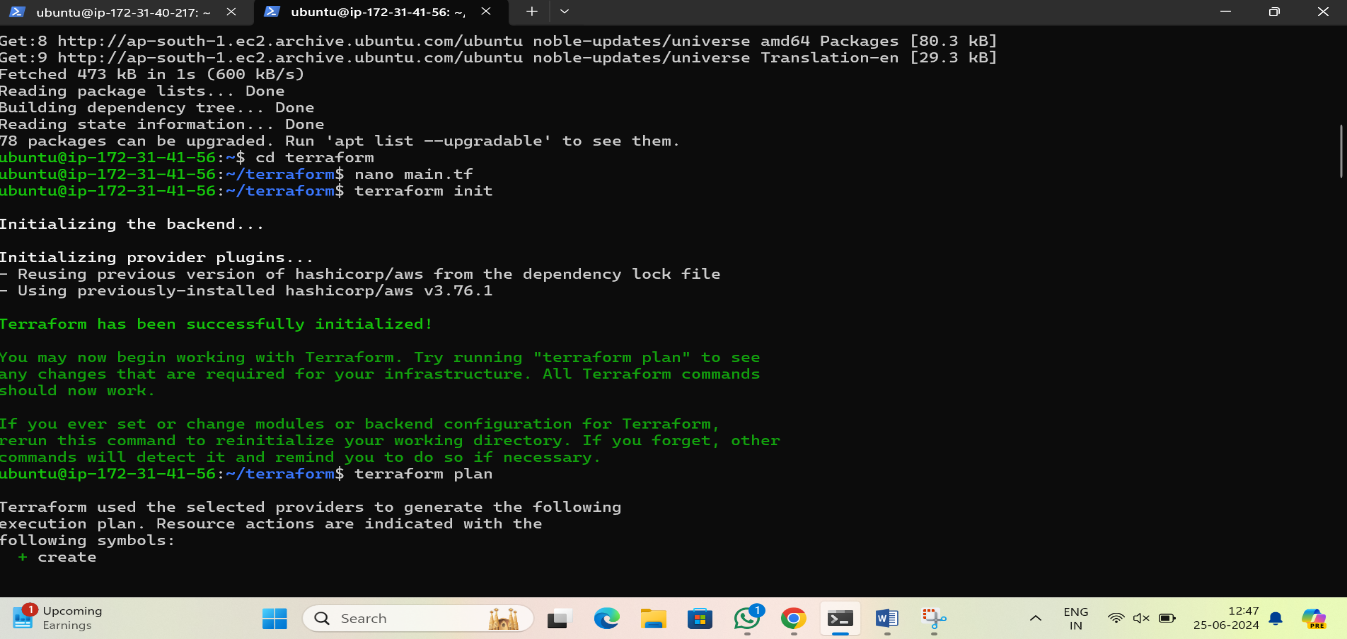
Specifies the AWS provider and the region where the resources will be created.

Also provide a access key and secret key of AWS account.

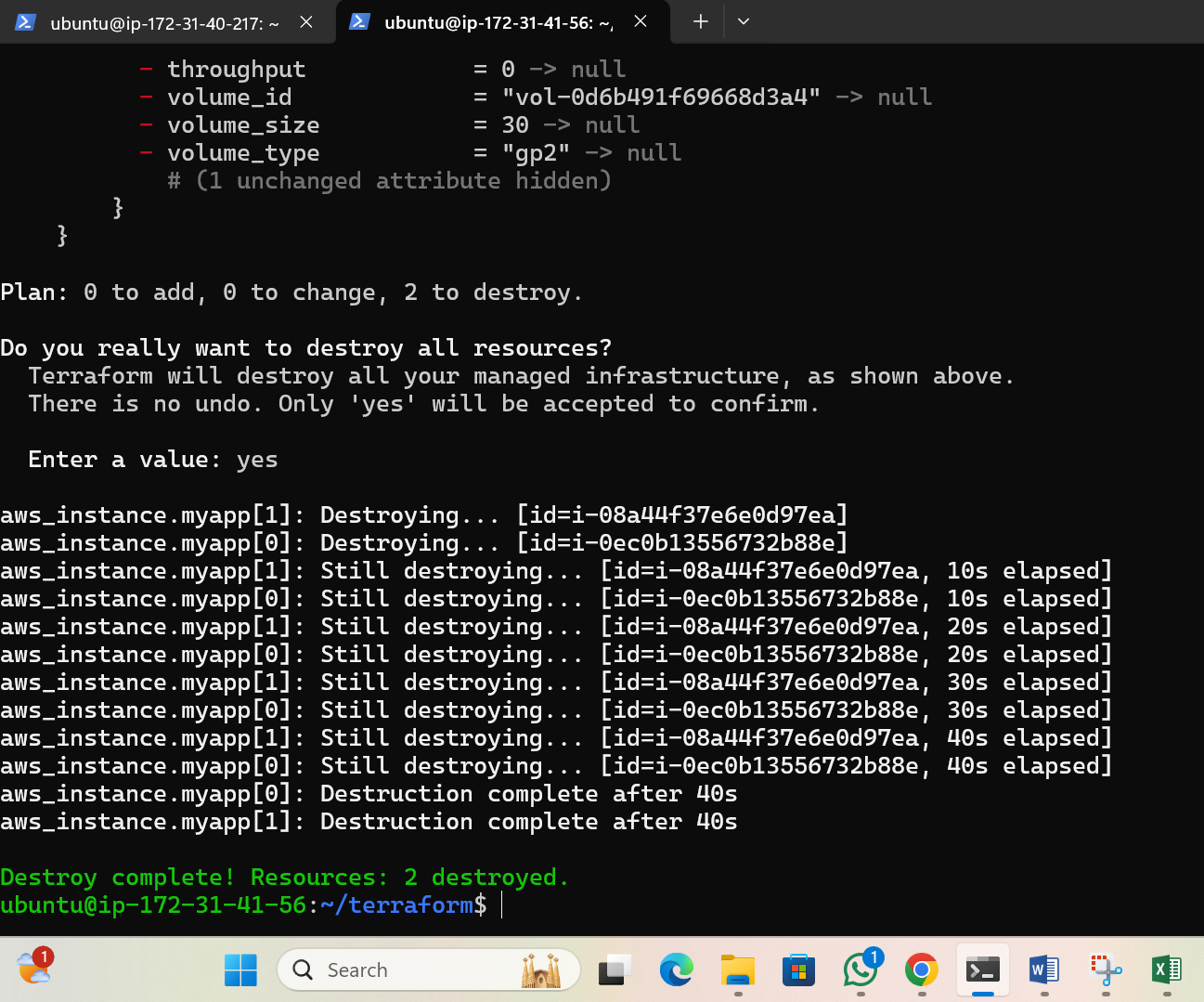
**Resource Block:**

Defines the aws\_instance resource to launch two EC2 instances.

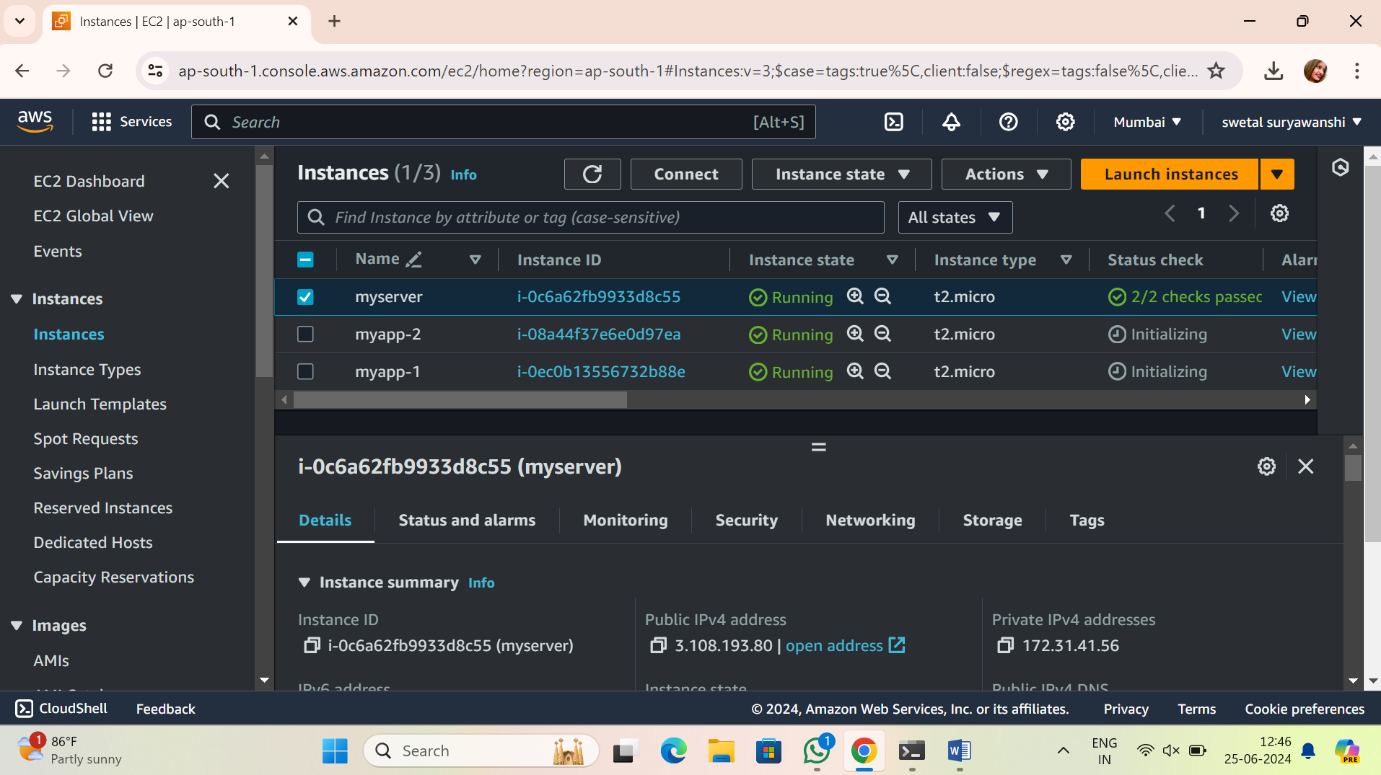
* **count:** Used to create multiple instances. In this case, we set count = 2 to create two instances.
* **ami:** The Amazon Machine Image (AMI) ID for Amazon Linux 2 in the ap-south-1 region. We need to mentioned " ami-0e1d06225679bc1c5" with the latest Amazon Linux 2 AMI ID for the ap-south-1 region.
* **instance\_type:** Specifies the type of instance to be created. Here, t2.micro is used.
* **tags:** Assigns a name tag to each instance, using the count index to differentiate between "myapp-1" and "myapp-2".
* **Initialize Terraform**: Run **“terraform init”** to initialize the configuration and download the necessary provider plugins.

****

* **Plan the Changes**: Run **“terraform plan”** to preview the changes that will be applied.
* **Confirm the Apply Step**: Type **“yes”** when prompted to confirm that we want to create the resources.



* This will create two EC2 instances in the **ap-south-1** region, with names **"myapp-1"** and **"myapp-2"** using the specified Amazon Linux AMI.



**Q.3 Install Terraform on local machine (Laptop), integrate aws and terraform with VS code. Using VS code launch an EC2 instances with name ‘myserver’ using Windows OS in ‘ap-south-1’ region.**

**Step 1: Install Terraform**

**On Windows:**

1. **Download Terraform:**

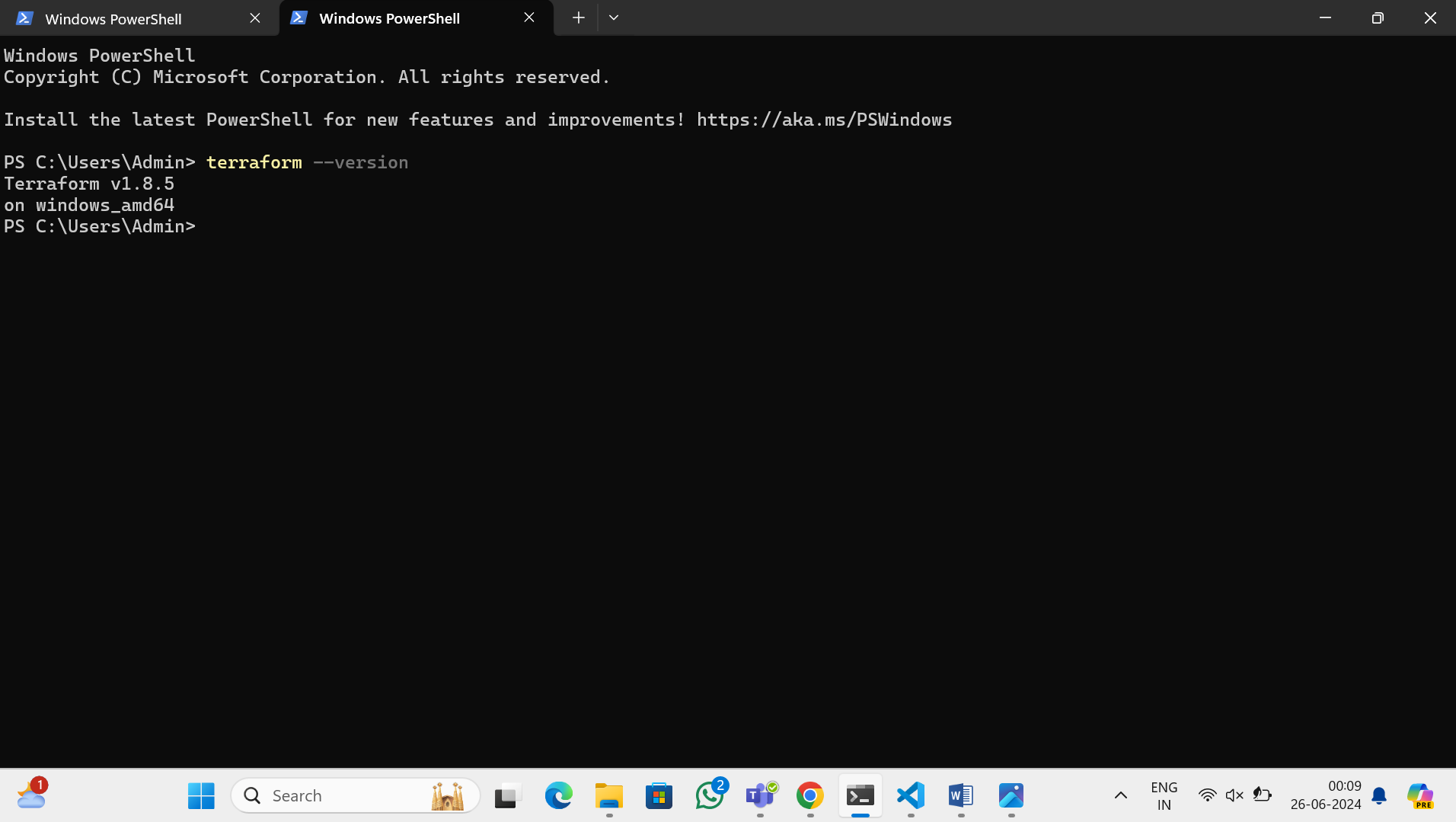
* Go to the Terraform download page.
* Download the appropriate package for your operating system.

1. **Install Terraform:**

* Unzip the downloaded file.
* Move the terraform.exe file to a directory included in our system's PATH. Typically, we can place it in **C:\Program Files\Terraform**
* Search system variables in laptop & click on edit syetem variables.
* Open a window which is shiown below.
* a. Click on Envirnonment Variables
* b. In system variable drop down list select path and click on Edit
* c. Open a new window click on new button
* d. Add a path which is shown as **C:\Program Files\Terraform**

1. **Verify Installation:**

* Open a command prompt and type:

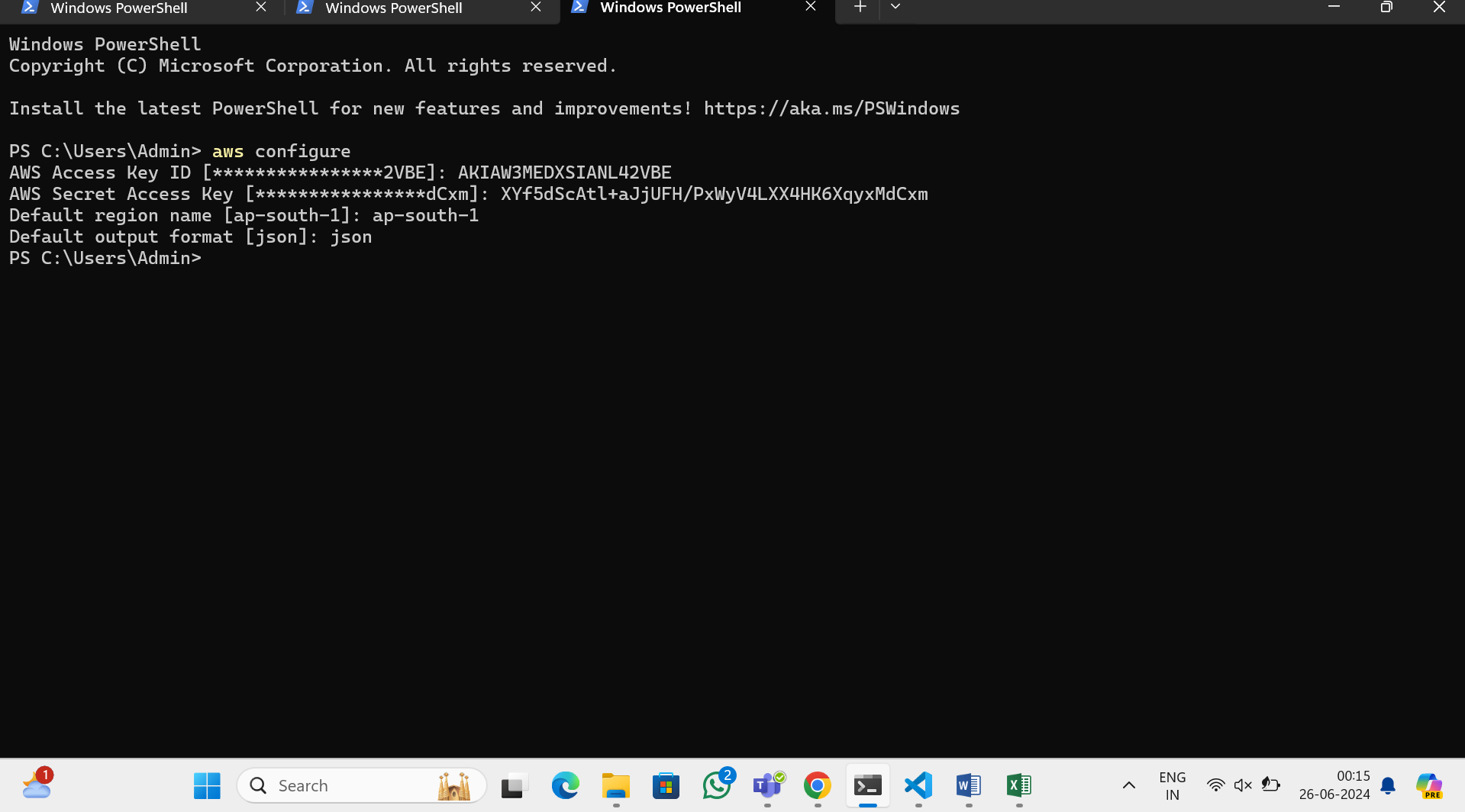
****

**Step 2: Set Up AWS CLI**

* Install AWS CLI: <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>
* Follow the installation instructions for our OS from the above link

Configure AWS CLI:

* Open a command prompt and configure AWS CLI with our credentials:
* Enter AWS Access Key ID, Secret Access Key, default region name (ap-south-1), and default output format (e.g., json).

****

**Step 3: Integrate AWS and Terraform with VS Code**

* Install Visual Studio Code:
* Download and install VS Code. <https://code.visualstudio.com/docs?dv=win>
* Install Terraform Extension in VS Code:
* Open VS Code.
* Go to the Extensions view by clicking on the square icon in the sidebar or pressing Ctrl+Shift+X.
* Search for “Terraform” and install the extension by HashiCorp.
* Install AWS Toolkit Extension in VS Code:
* In the Extensions view, search for “AWS Toolkit” and install the extension by AWS.

**Step 4: Create Terraform Configuration in VS Code**

**Open a New Folder:**

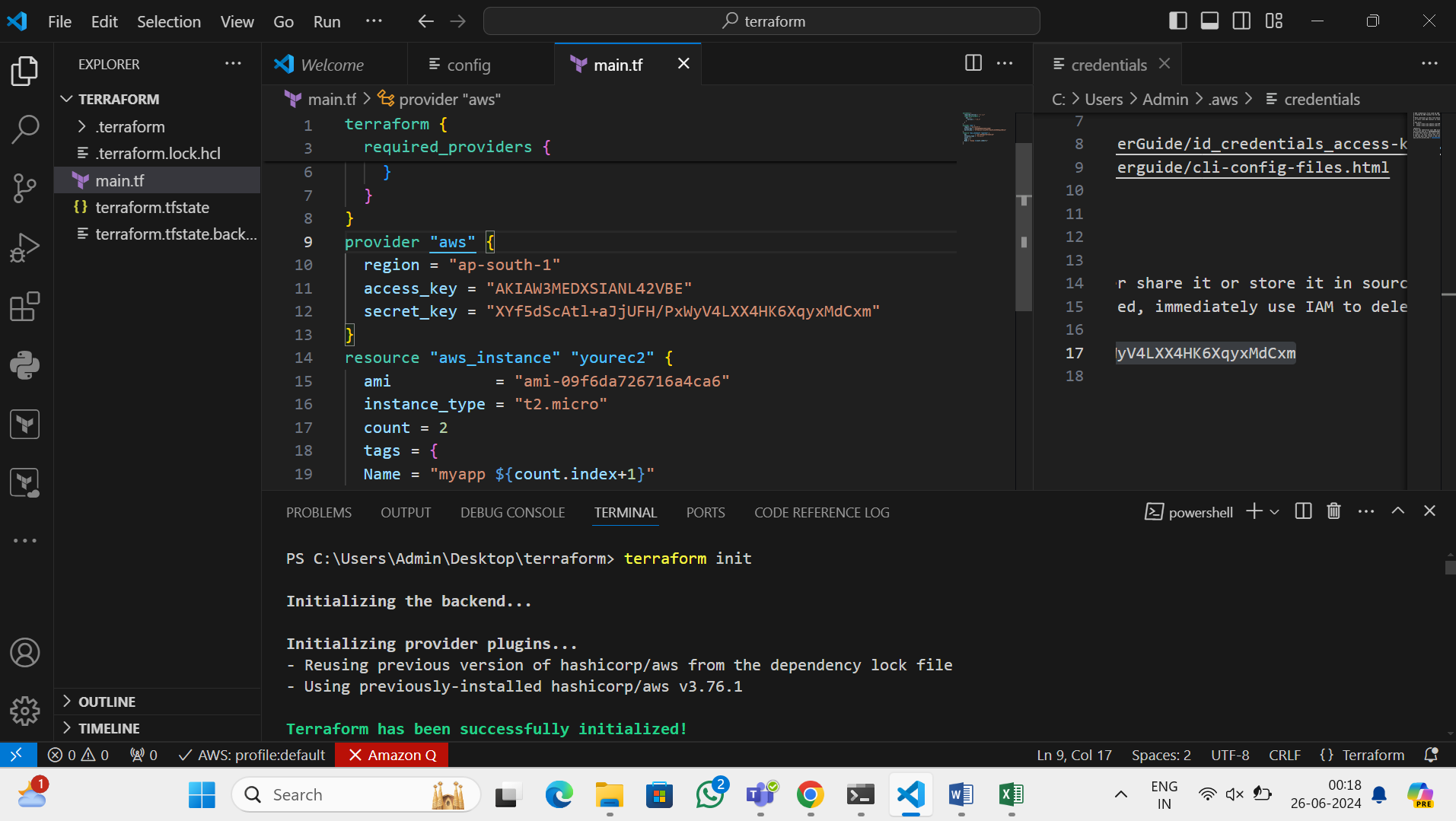
* In VS Code, open a new folder where we will store our Terraform configuration files.

**Create “ main.tf ” File:**

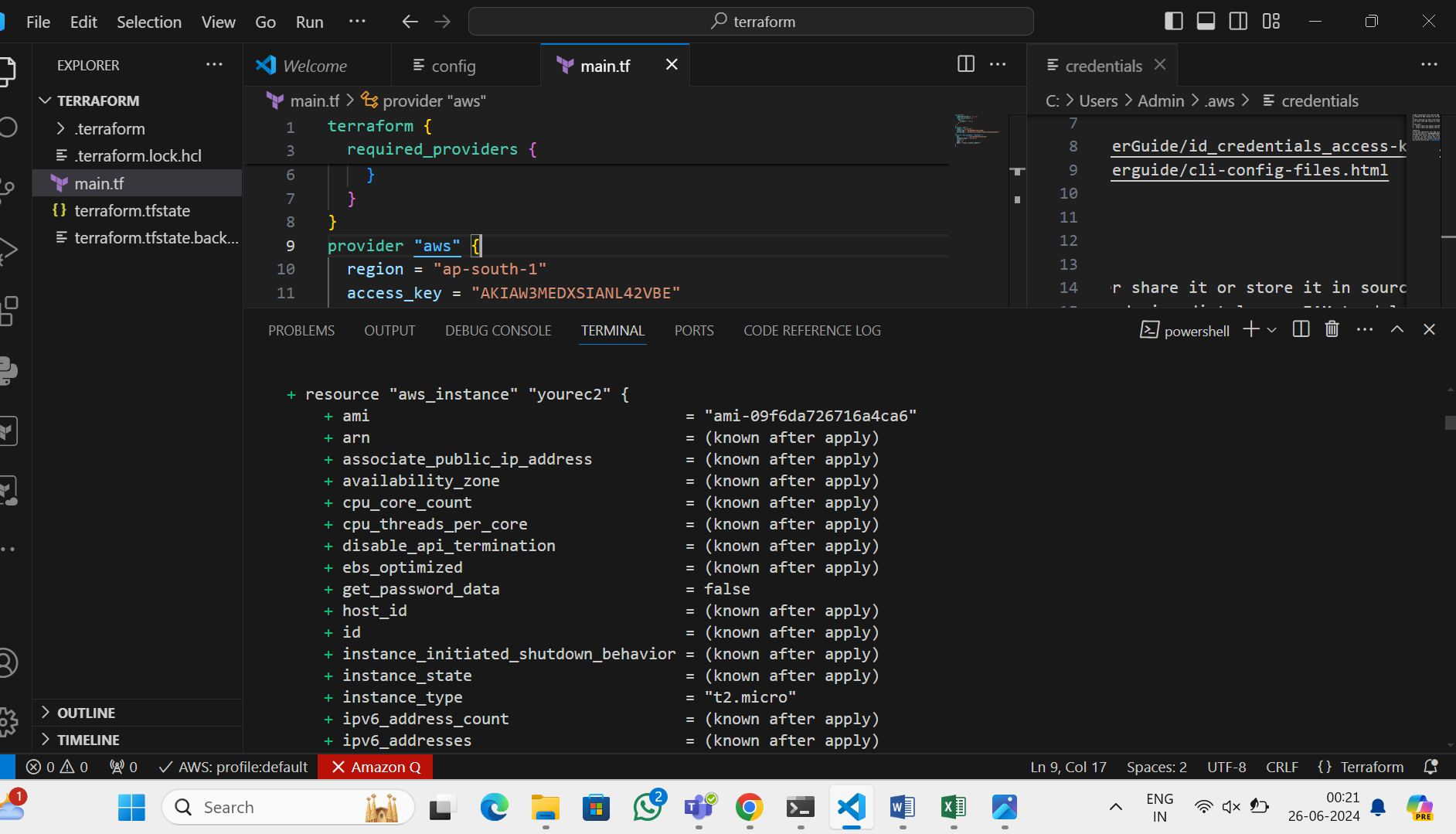
* Create a new file named main.tf in the folder and add the following configuration:
* Make sure to replace the AMI ID with the latest Windows Server AMI for the ap-south-1 region.

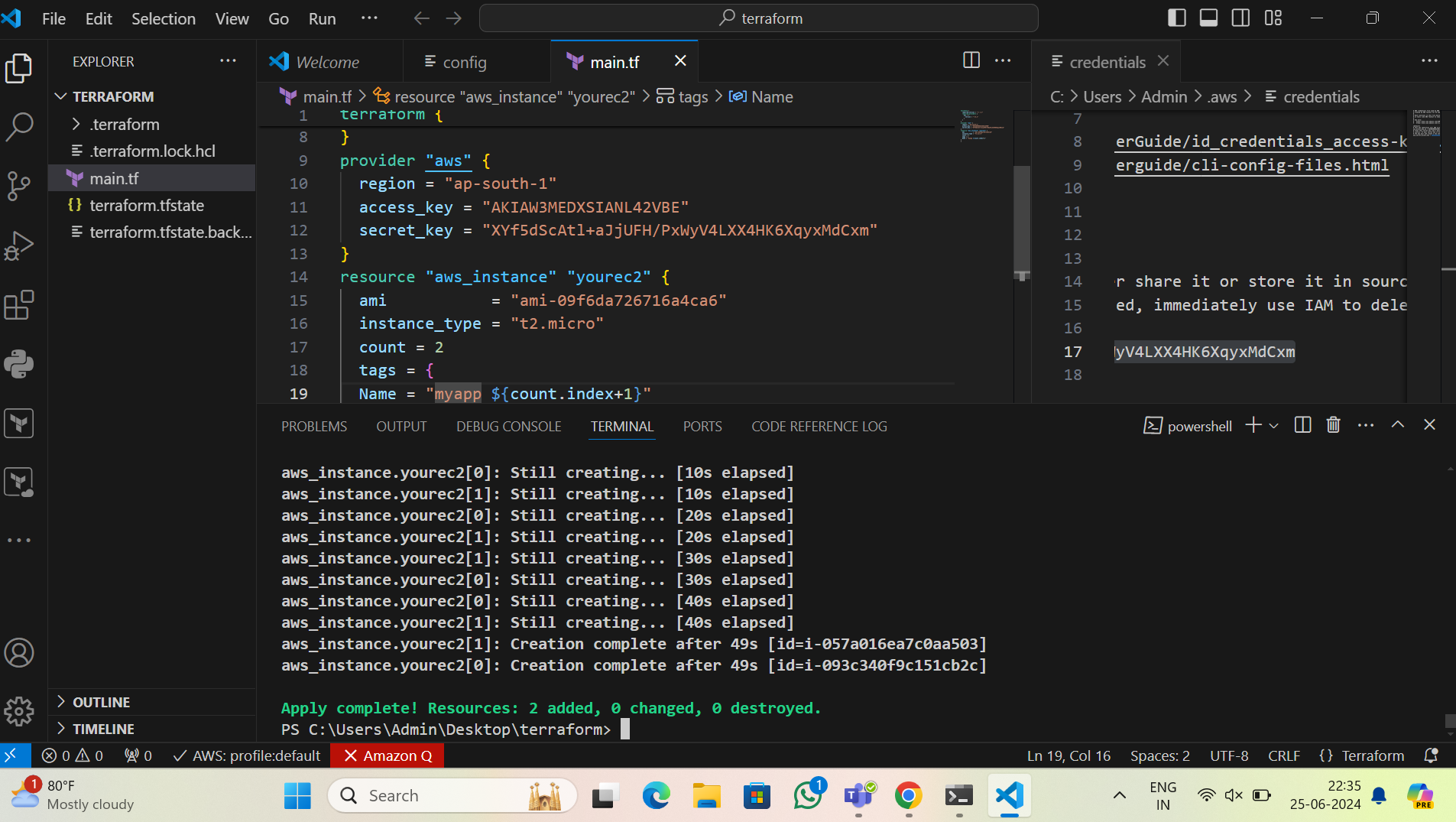
**Open the integrated terminal in VS Code & Run the following command**

* terraform init

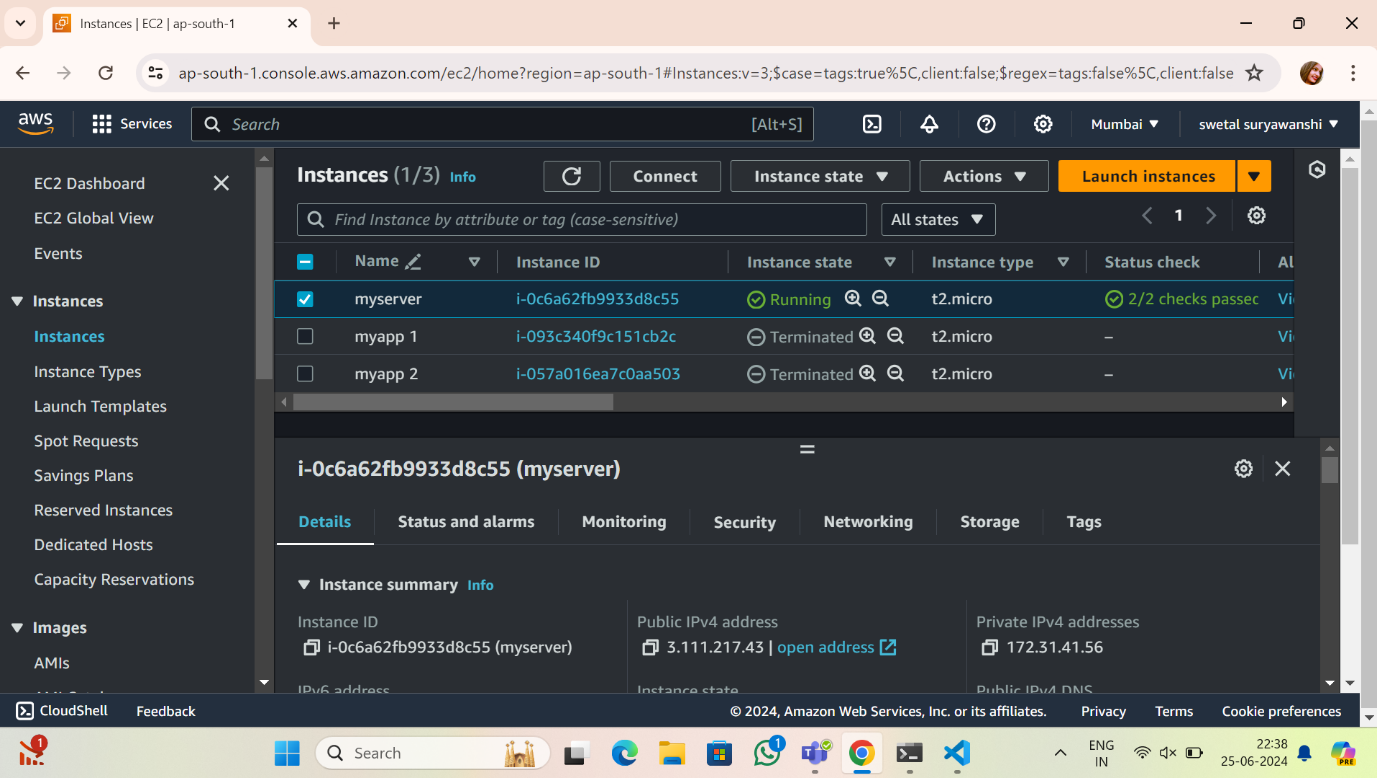
****

* terraform plan

****

****

* terraform apply

****

Created an EC2 instance named myserver with a Windows OS in the ap-south-1 region.