LOYALIST COLLEGE
IN TORONTO

WINP2000 - Cloud Management

Week 13 – In class Lab 6

Use Terraform Infrastructure as Code (IaC) to deploy a VM to AWS and manage it by adding network security.

Assignment Submitted by:

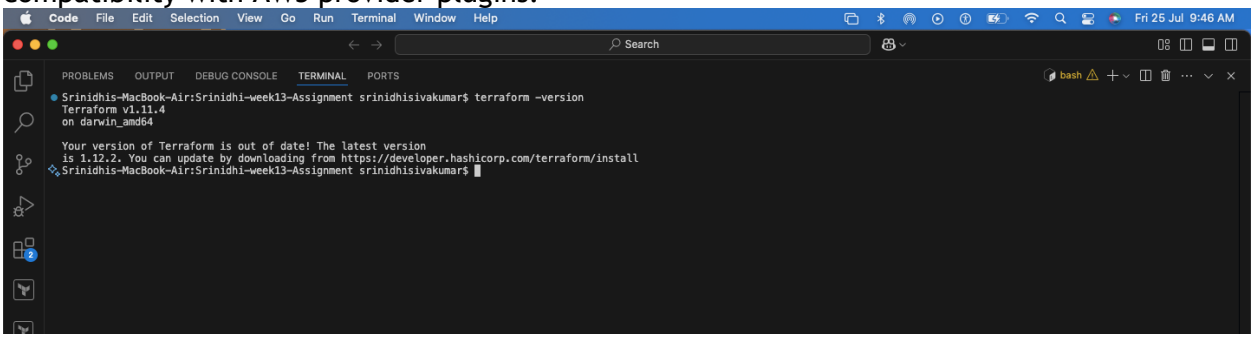
Student Name: Srinidhi Sivakumar
Student ID: 500237144
Course Code: WINP2000
Instructor Name: Sergio Loza

1. Verifying Terraform Installation

Terraform is an Infrastructure as Code tool that allows provisioning and managing cloud resources using declarative configuration files. Before starting, Terraform was installed and its version was confirmed using:

terraform -version

The output confirmed that terraform **v1.11.4** was installed successfully, ensuring compatibility with AWS provider plugins.



2. Configuring AWS CLI and Verifying Login

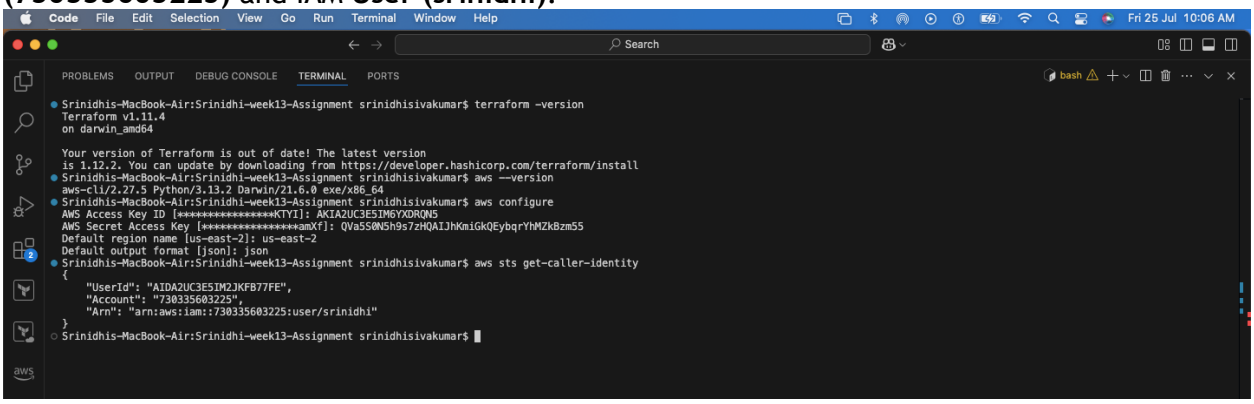
To interact with AWS, the AWS CLI was configured using programmatic access keys created in the IAM console. The configuration process included setting up the Access Key ID, Secret Access Key, region (us-east-2), and output format (json).
Configuration command:

aws configure

Login verification was done with:

aws sts get-caller-identity

This command confirmed successful authentication, showing the AWS **Account ID (730335603225)** and IAM **User (srinidhi)**.



3. Preparing Terraform Files and Initializing Terraform

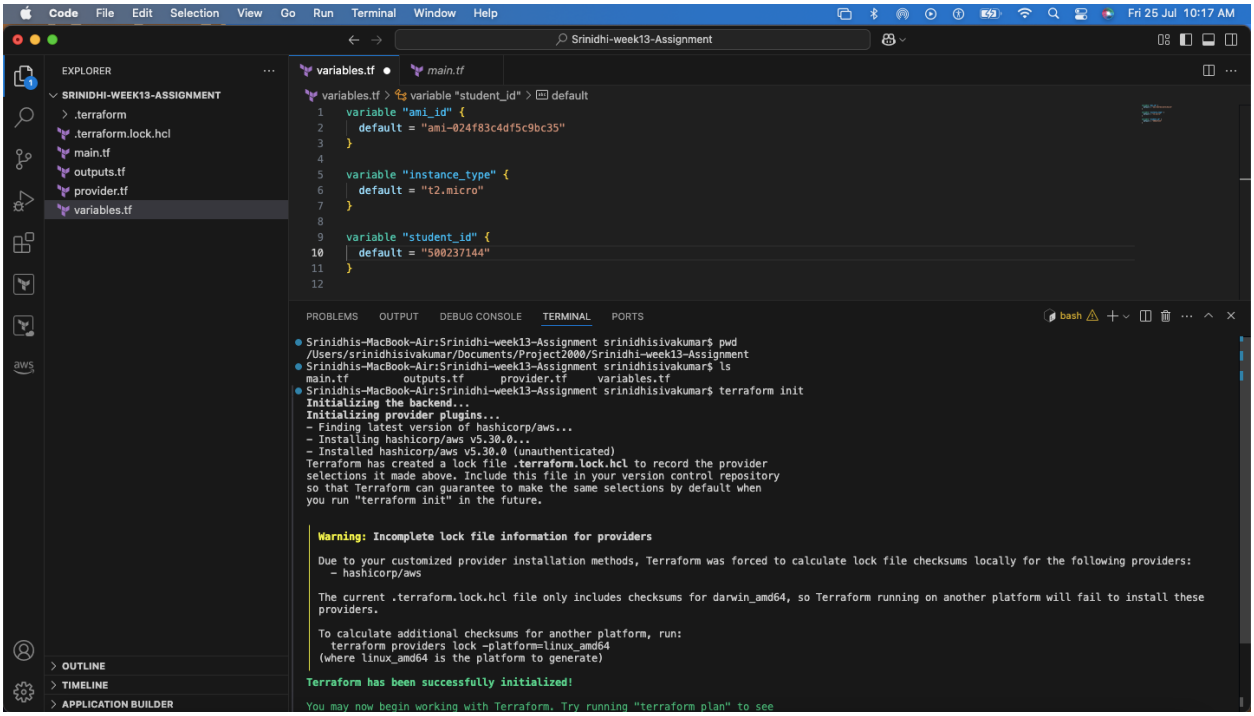
The project included the following Terraform configuration files:

- **provider.tf** – Configures AWS as the cloud provider.
- **variables.tf** – Defines variables such as AMI ID, instance type, and student ID.
- **main.tf** – Declares the EC2 instance resource and tags.
- **outputs.tf** – Outputs instance ID and public IP for easy reference.

Terraform was initialized using:

terraform init

This command downloaded the necessary AWS provider plugins and prepared the working directory.
The Terraform files were uploaded to GitHub for reference:
🔗 GitHub Repository – AWS_terraform



4. Running Terraform Plan (Initial Deployment)

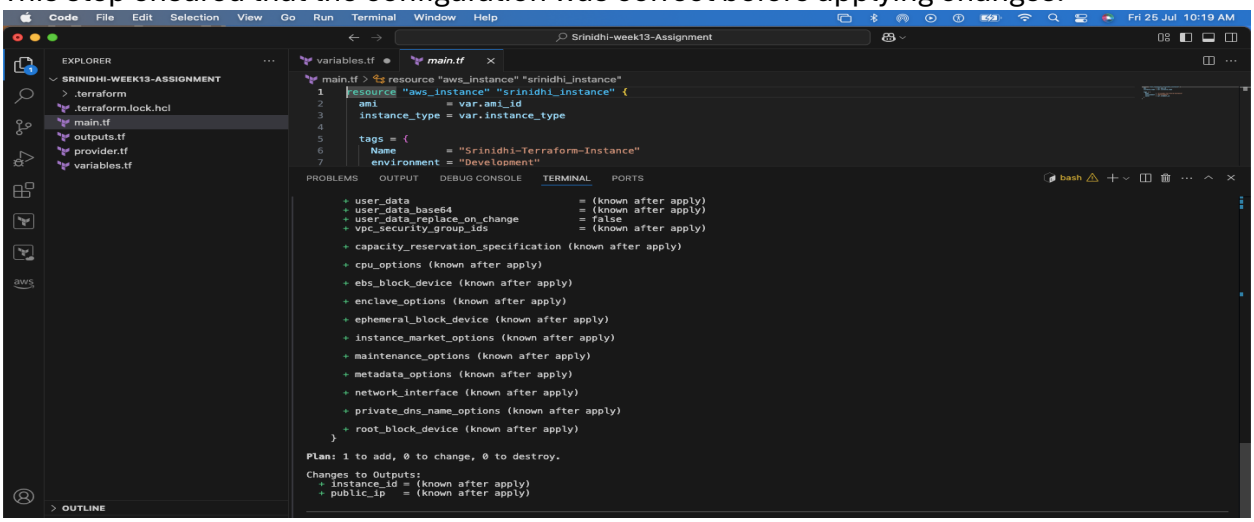
The plan command shows the actions Terraform will perform without applying them. It was executed as:

terraform plan

Terraform displayed an execution plan to **create a new EC2 instance** in the us-east-2 region with the required tags:

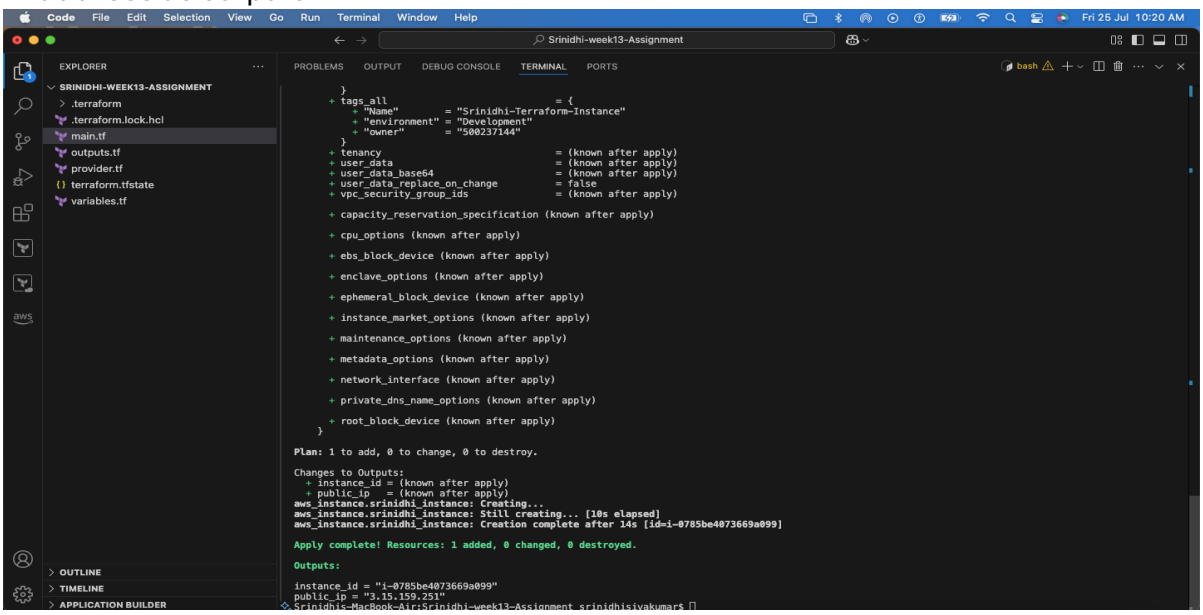
- environment = Development
- owner = 500237144

This step ensured that the configuration was correct before applying changes.



5. Applying Terraform to Create the EC2 Instance

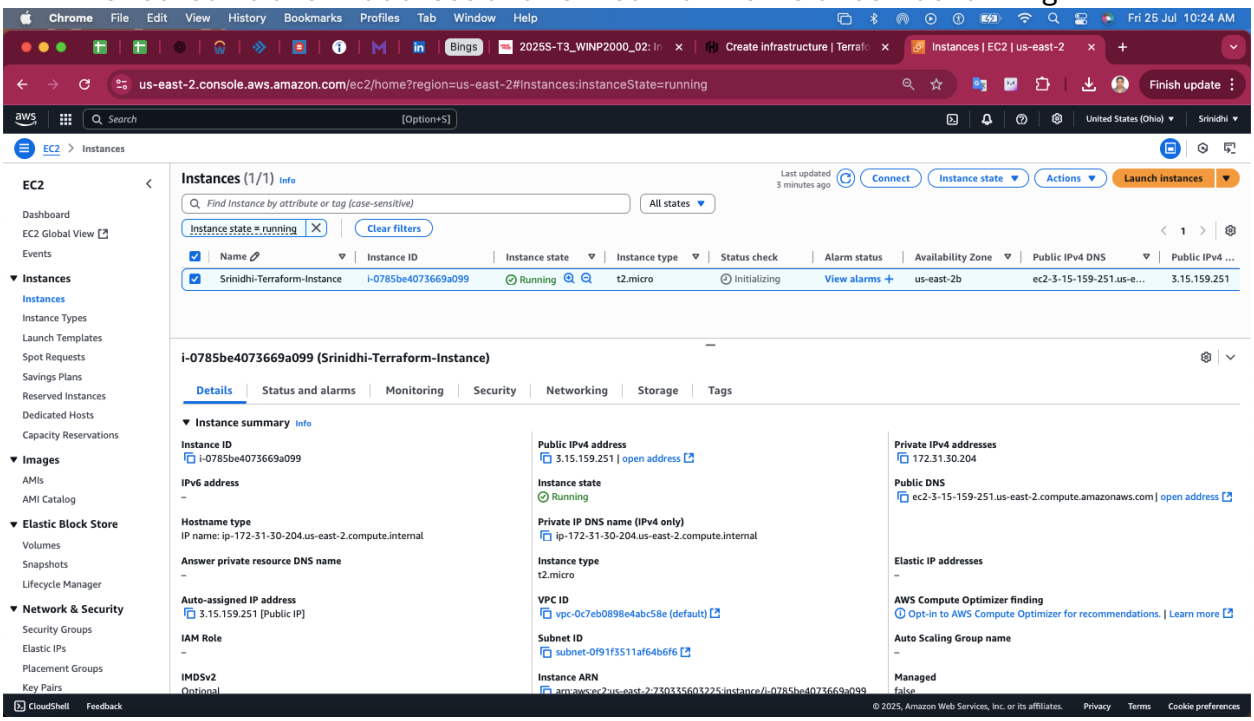
The following command was executed to provision the infrastructure:
terraform apply -auto-approve
Terraform successfully created an **EC2 instance** and displayed the **Instance ID** and **Public IP address** as outputs.



6. Validating the Instance in AWS Console

The EC2 instance was verified in the AWS Management Console:

- Matched **Instance ID** from Terraform outputs.
- Checked **Public IP address** and verified that the instance was running.

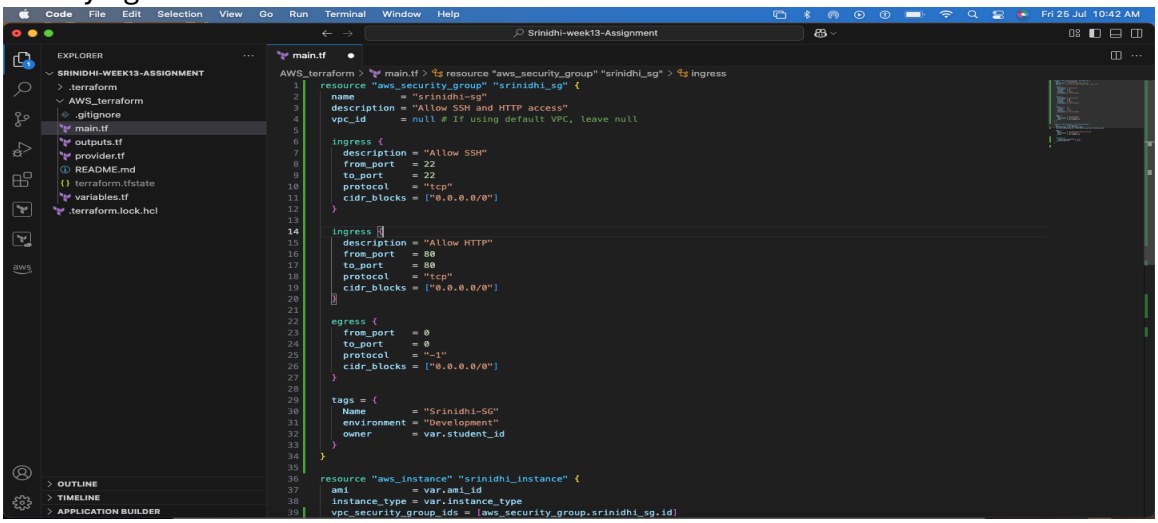


7. Updating Terraform to Add Network Security

To enhance security, a **Security Group resource** was added to main.tf to allow SSH (port 22) and HTTP (port 80) access. This security group was then attached to the existing EC2 instance using:

vpc_security_group_ids = [aws_security_group.srinidhi_sg.id]

This update demonstrates **managing infrastructure using Terraform** without manually modifying resources in the AWS Console.

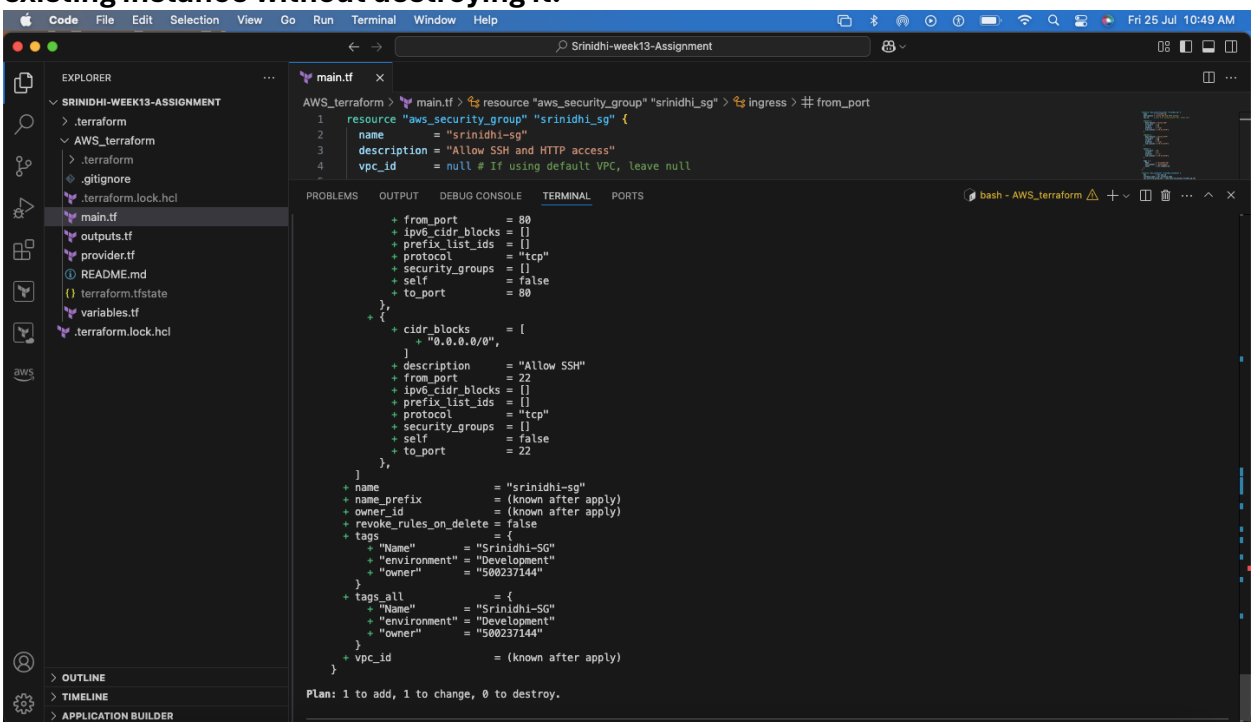


8. Running Terraform Plan (After Modification)

To preview the changes, the plan command was executed again:

terraform plan

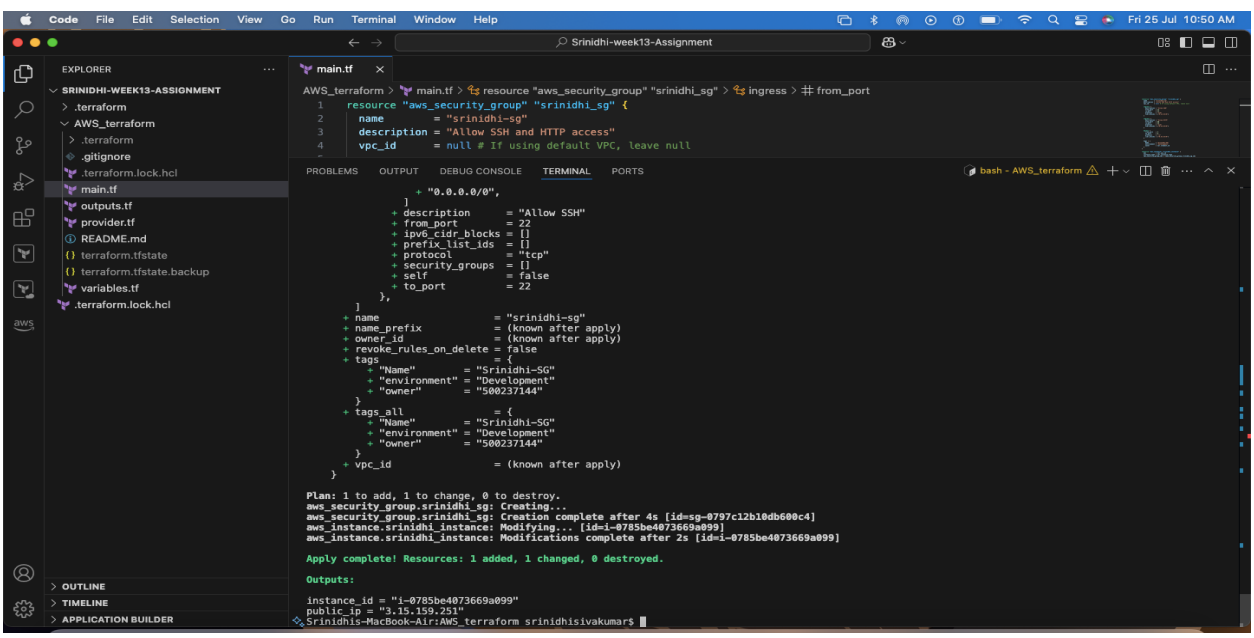
Terraform showed ~ update in-place, meaning the **security group would be added to the existing instance without destroying it.**



9. Applying Changes to Attach the Security Group

Terraform created the new security group and successfully attached it to the existing EC2 instance.

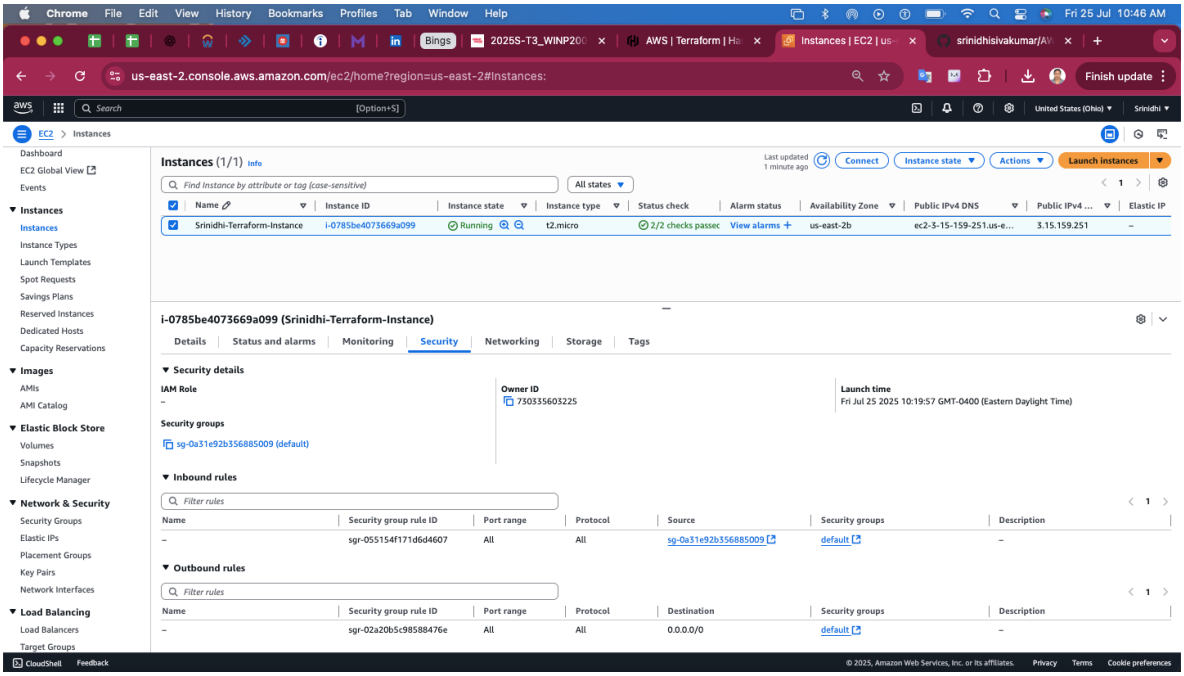
terraform apply -auto-approve



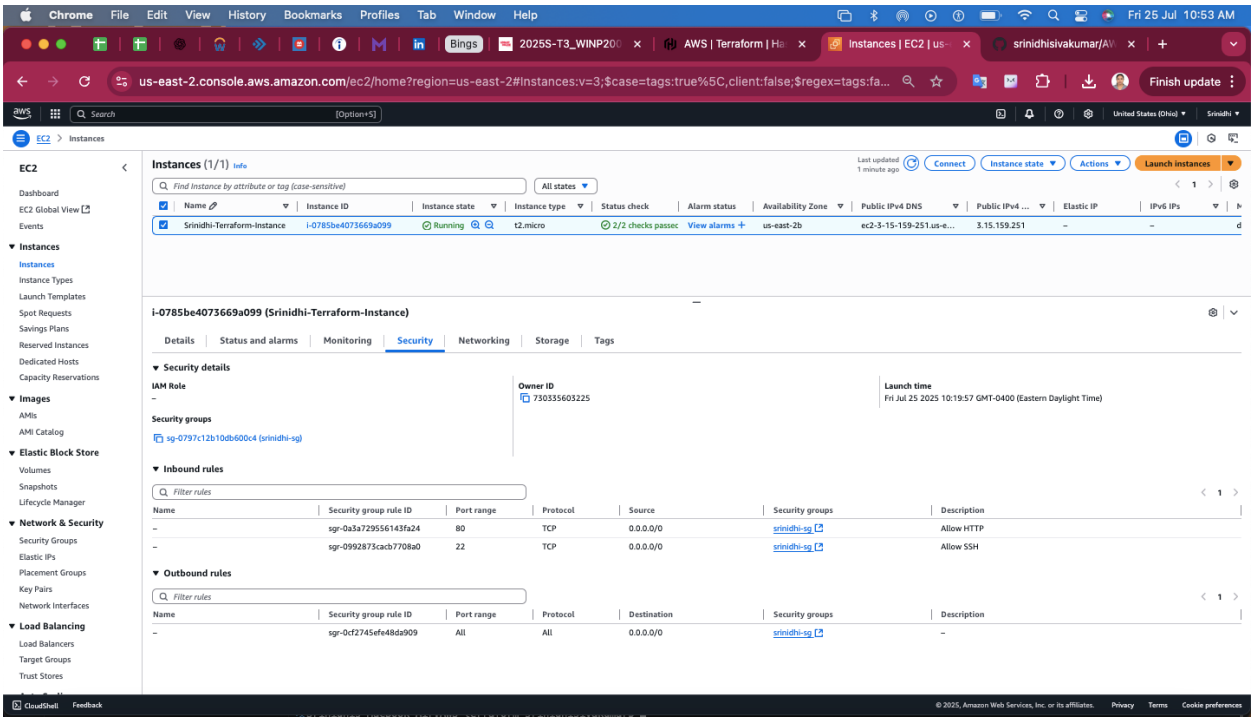
10. Verifying Changes in AWS Console

- Navigated to **EC2 → Instances → Networking → Security Groups**.
- Confirmed that the **new security group (srinidhi-sg)** was attached.
- Verified inbound rules for SSH and HTTP access.

Before:



After:

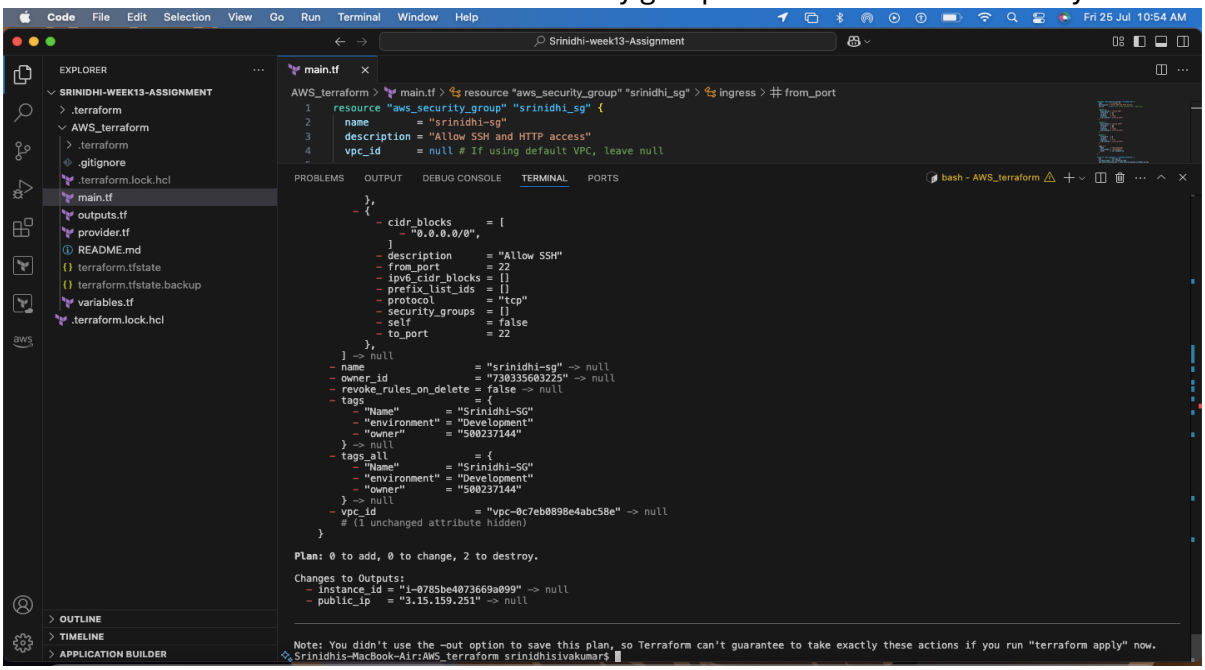


11. Planning to Destroy the Infrastructure

To remove all resources, the following command was executed to preview the resources to be deleted:

terraform plan -destroy

Terraform listed the EC2 instance and security group as resources to be destroyed.

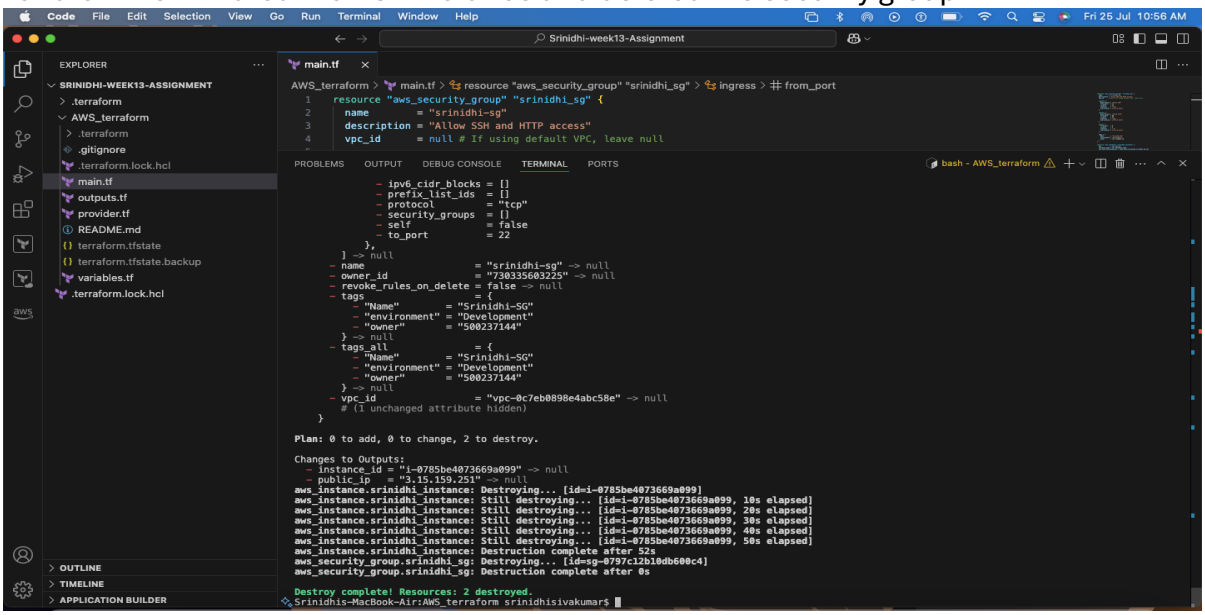


12. Destroying the Infrastructure

To terminate the resources, the following command was run:

terraform destroy -auto-approve

Terraform terminated the EC2 instance and deleted the security group.



13. Final Validation in AWS Console

In the AWS Management Console, the **instance status showed as Terminated**, confirming that Terraform successfully cleaned up all resources.