



# **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

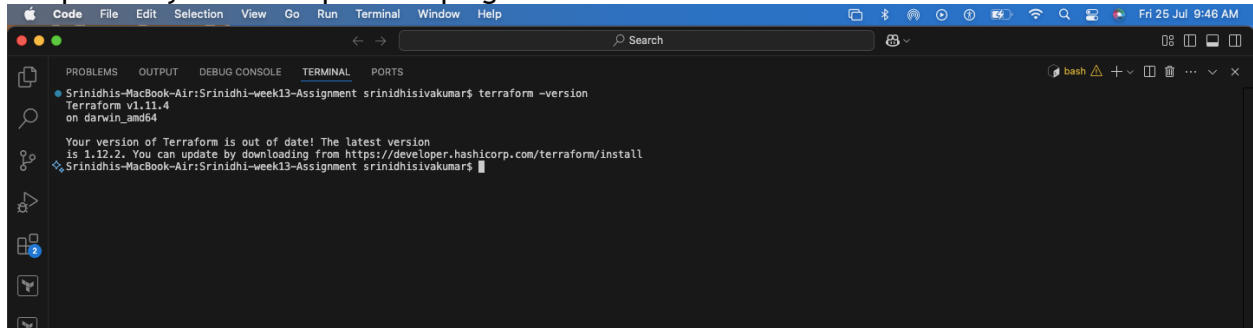
Instructor Name: Sergio Loza

# 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.



```
Code File Edit Selection View Go Run Terminal Window Help
Srinidhis-MacBook-Air:Srinidhi-week13-Assignment srinidhisivakumar$ terraform -version
Terraform v1.11.4
on darwin_amd64

Your version of Terraform is out of date! The latest version
is 1.12.2. You can update by downloading from https://developer.hashicorp.com/terraform/install
Srinidhis-MacBook-Air:Srinidhi-week13-Assignment srinidhisivakumar$
```

## 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).

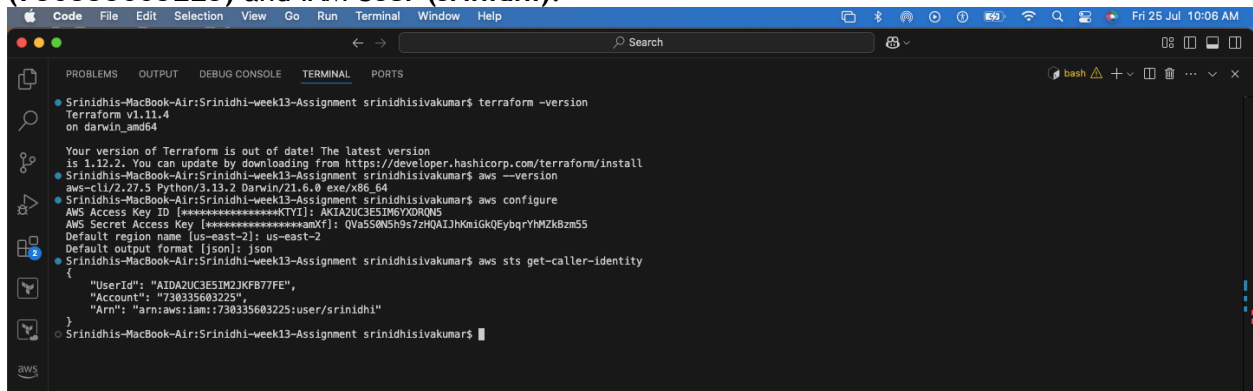
Configuration command:

```
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)**.



```
Code File Edit Selection View Go Run Terminal Window Help
Srinidhis-MacBook-Air:Srinidhi-week13-Assignment srinidhisivakumar$ terraform -version
Terraform v1.11.4
on darwin_amd64

Your version of Terraform is out of date! The latest version
is 1.12.2. You can update by downloading from https://developer.hashicorp.com/terraform/install
Srinidhis-MacBook-Air:Srinidhi-week13-Assignment srinidhisivakumar$ aws --version
aws-cli/2.7.5 Python/3.13.2 Darwin/21.6.0 exe/x86_64
Srinidhis-MacBook-Air:Srinidhi-week13-Assignment srinidhisivakumar$ aws configure
AWS Access Key ID [*****]: AKIA2UC3E5IN6YXORQNS
AWS Secret Access Key [*****]: QVa5S0NSh9s7zHQAIJHm1GkQEyBqRYM2k8zm55
Default region name [us-east-2]: us-east-2
Default output format [json]: json
Srinidhis-MacBook-Air:Srinidhi-week13-Assignment srinidhisivakumar$ aws sts get-caller-identity
{
  "UserId": "AIDA2UC3E5IM2JKFB77FE",
  "Account": "730335603225",
  "Arn": "arn:aws:iam:730335603225:user/srinidhi"
}
Srinidhis-MacBook-Air:Srinidhi-week13-Assignment srinidhisivakumar$
```

### 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.

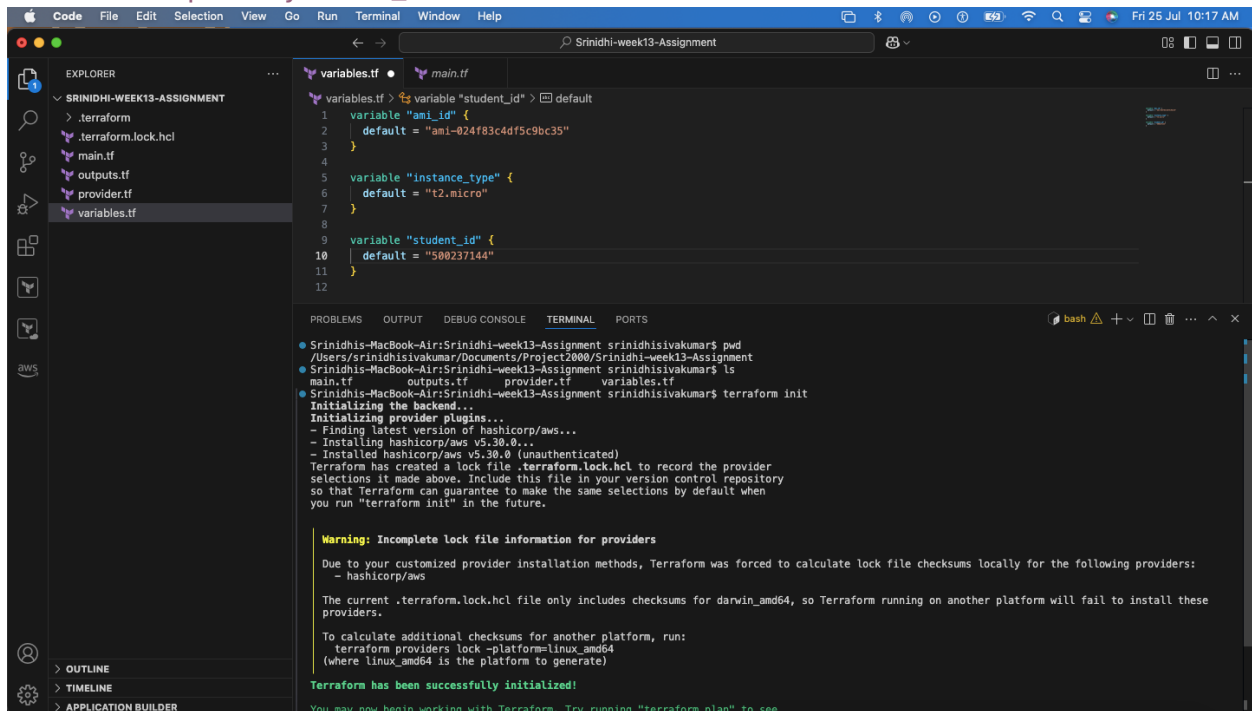
Terraform was initialized using:

```
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](#)



The screenshot shows a code editor with the following files in the Explorer:

- .terraform
- .terraform.lock.hcl
- main.tf
- outputs.tf
- provider.tf
- variables.tf

The **variables.tf** file is open, showing the following content:

```
1 variable "student_id" {
2   default = "500237144"
3 }
4
5 variable "instance_type" {
6   default = "t2.micro"
7 }
8
9 variable "ami_id" {
10  default = "ami-024f83c4df5c9bc35"
11 }
12
```

The terminal window shows the output of the `terraform init` command:

```
Srinidhis-MacBook-Air:Srinidhi-week13-Assignment srinidhisivakumar$ pwd
/Users/srinidhisivakumar/Documents/Project2000/Srinidhi-week13-Assignment
Srinidhis-MacBook-Air:Srinidhi-week13-Assignment srinidhisivakumar$ ls
main.tf  outputs.tf  provider.tf  variables.tf
Srinidhis-MacBook-Air:Srinidhi-week13-Assignment srinidhisivakumar$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.38.0...
- Installed hashicorp/aws v5.38.0 (unauthenticated)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Warning: Incomplete lock file information for providers
Due to your customized provider installation methods, Terraform was forced to calculate lock file checksums locally for the following providers:
- hashicorp/aws
The current .terraform.lock.hcl file only includes checksums for darwin_amd64, so Terraform running on another platform will fail to install these
providers.
To calculate additional checksums for another platform, run:
  terraform providers lock -platform=linux_amd64
(where linux_amd64 is the platform to generate)

Terraform has been successfully initialized!
You may now begin working with Terraform. Try running "terraform plan" to see
```

## 4. Running Terraform Plan (Initial Deployment)

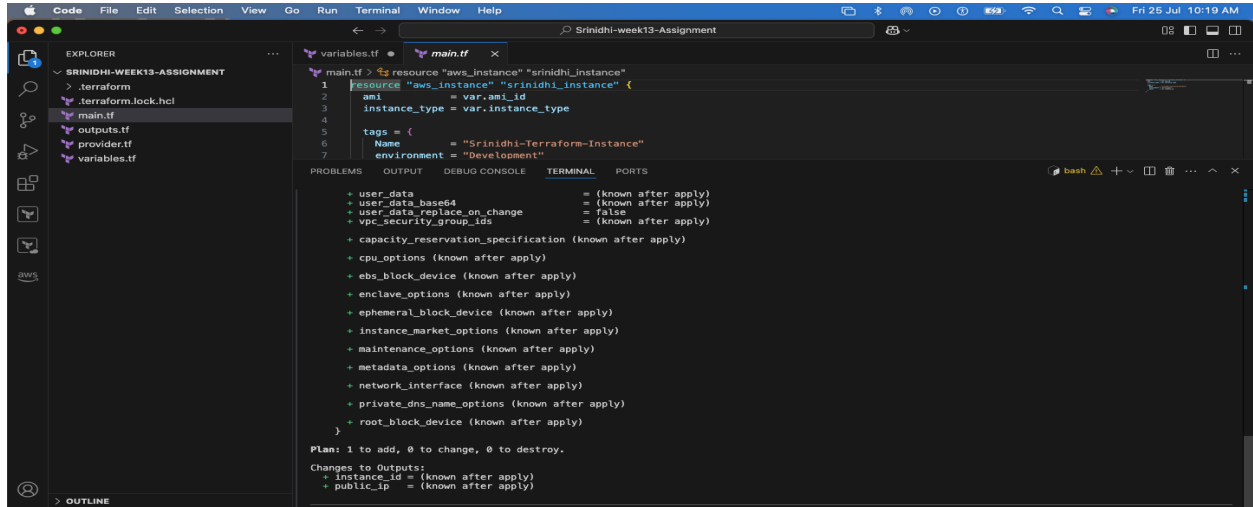
The plan command shows the actions Terraform will perform without applying them. It was executed as:

```
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.



The screenshot shows the VS Code interface with the Terraform plan output displayed in the terminal. The plan shows that a new EC2 instance will be created with the following configuration:

```
resource "aws_instance" "srinidhi_instance" {
  ami           = var.ami_id
  instance_type = var.instance_type
  tags = {
    Name        = "Srinidhi-Terraform-Instance"
    environment = "Development"
  }
  user_data_base64 = (known after apply)
  user_data_replace_on_change = false
  vpc_security_group_ids = (known after apply)
  capacity_reservation_specification (known after apply)
  cpu_options (known after apply)
  ebs_block_device (known after apply)
  enclave_options (known after apply)
  ephemeral_block_device (known after apply)
  instance_market_options (known after apply)
  maintenance_options (known after apply)
  metadata_options (known after apply)
  network_interface (known after apply)
  private_dns_name_options (known after apply)
  root_block_device (known after apply)
}
```

Plans: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:

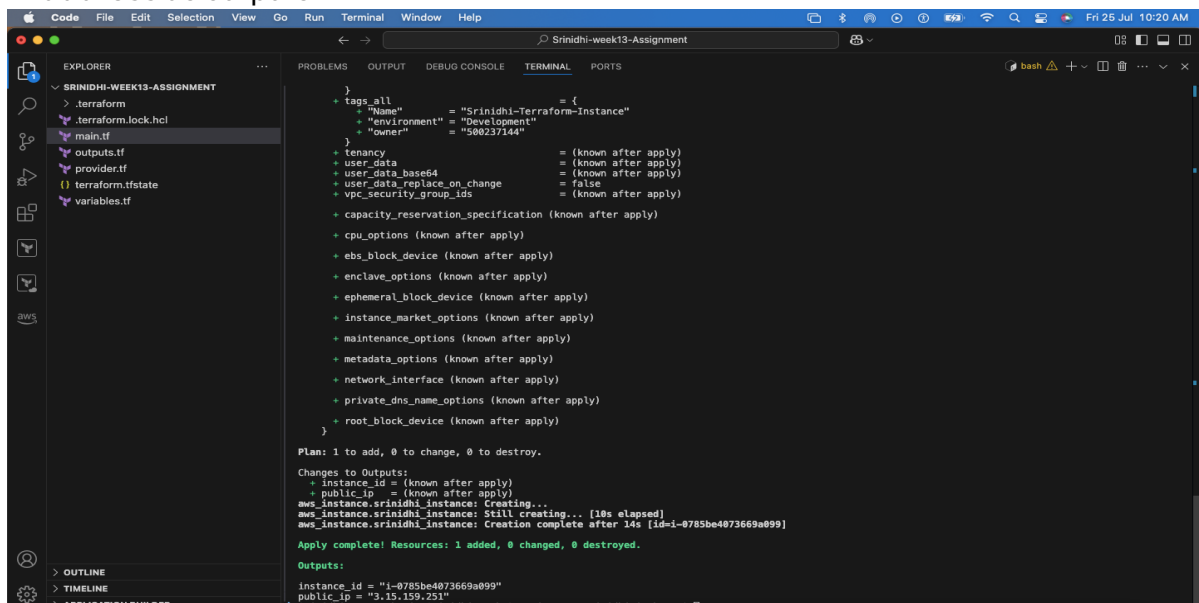
- + instance\_id = (known after apply)
- + public\_ip = (known after apply)

## 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.



The screenshot shows the VS Code interface with the Terraform apply output displayed in the terminal. The output shows that the EC2 instance was successfully created with the following configuration:

```
tags = {
  Name        = "Srinidhi-Terraform-Instance"
  environment = "Development"
  owner       = "500237144"
}
+ tenancy = (known after apply)
+ user_data = (known after apply)
+ user_data_base64 = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids = (known after apply)
+ capacity_reservation_specification (known after apply)
+ cpu_options (known after apply)
+ ebs_block_device (known after apply)
+ enclave_options (known after apply)
+ ephemeral_block_device (known after apply)
+ instance_market_options (known after apply)
+ maintenance_options (known after apply)
+ metadata_options (known after apply)
+ network_interface (known after apply)
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}
```

Plans: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:

- + instance\_id = (known after apply)
- + public\_ip = (known after apply)

aws\_instance.srinidhi\_instance: Creating...

aws\_instance.srinidhi\_instance: Still creating... [18s elapsed]

aws\_instance.srinidhi\_instance: Creation complete after 14s [id=i-0785be4073669a099]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

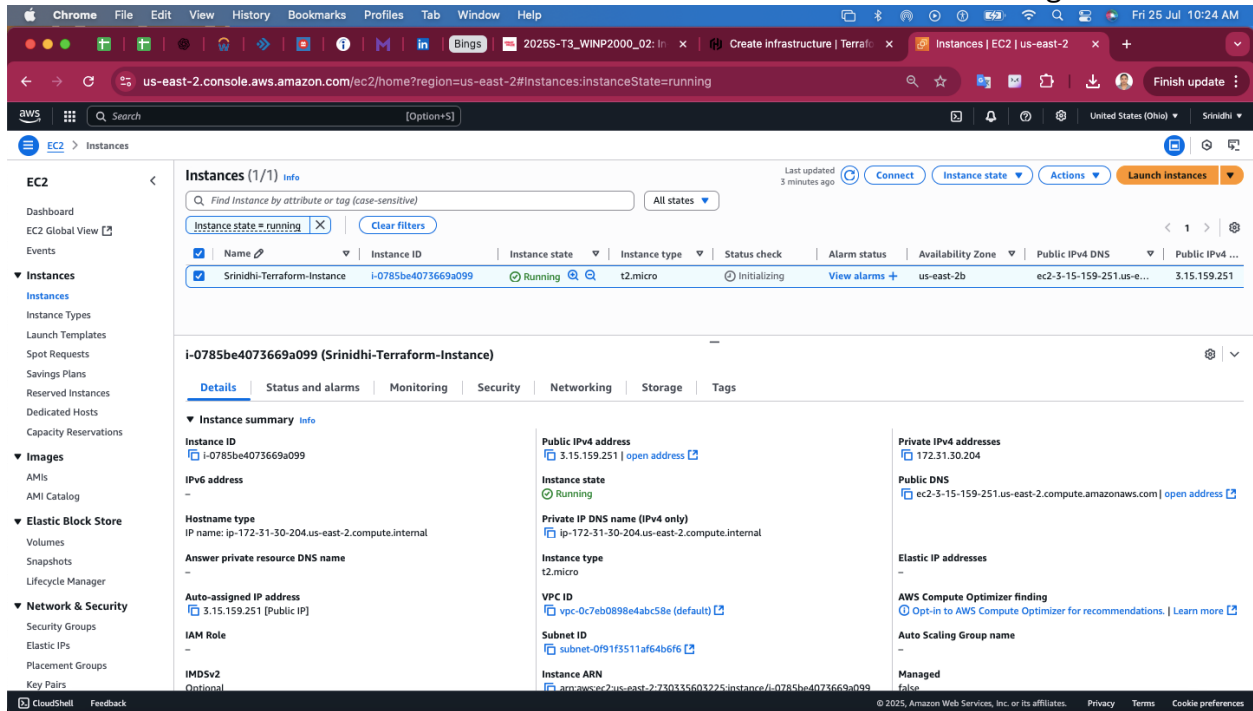
Outputs:

- instance\_id = "i-0785be4073669a099"
- public\_ip = "3.15.159.251"

## 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.

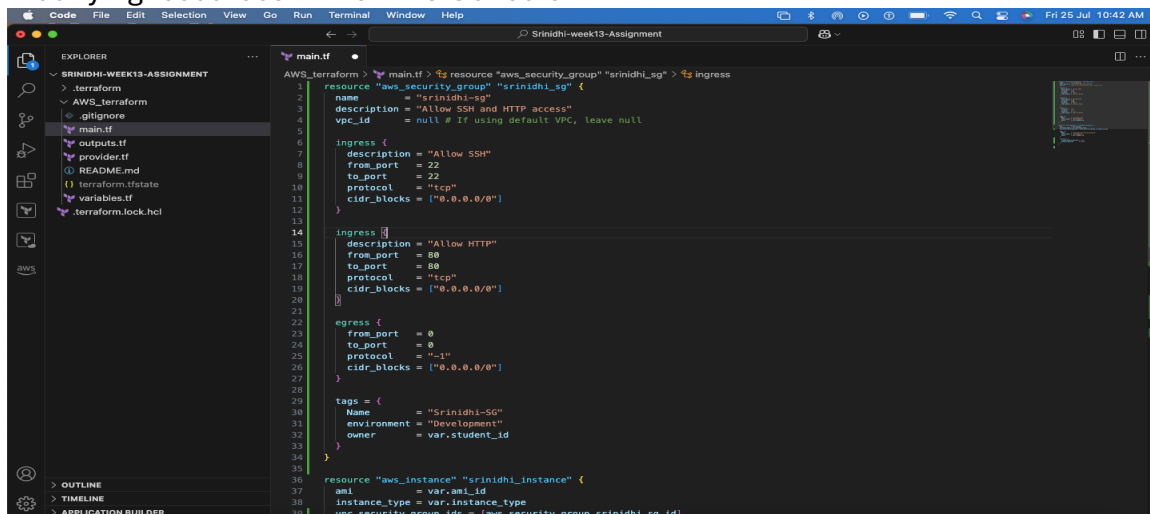


## 7. Updating Terraform to Add Network Security

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.

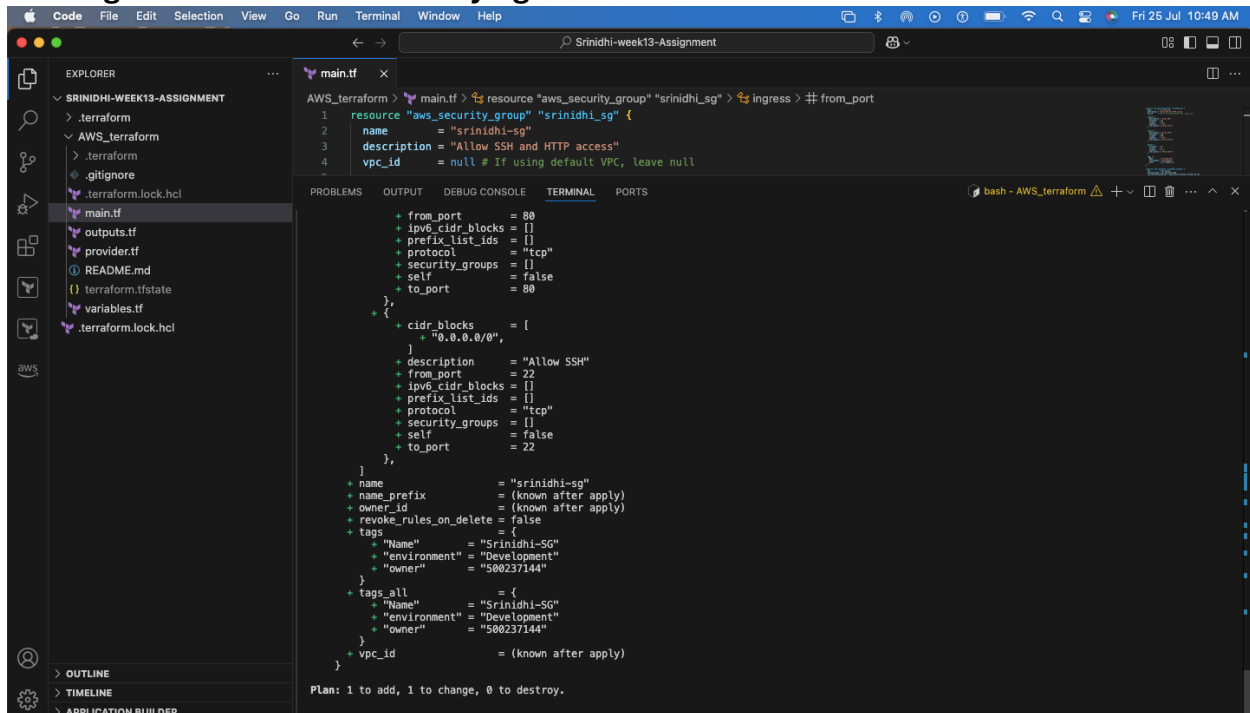


## 8. Running Terraform Plan (After Modification)

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.**



The screenshot shows the VS Code interface with the Explorer pane on the left displaying the project structure for 'Srinidhi-week13-Assignment'. The main editor shows the 'main.tf' file with a Terraform resource definition for 'aws\_security\_group' named 'sridinhi\_sg'. The resource configuration includes a name, description, vpc\_id, and ingress rules for port 80. The terminal pane at the bottom shows the output of the 'terraform plan' command, indicating that the plan is to add the security group and update the existing instance configuration.

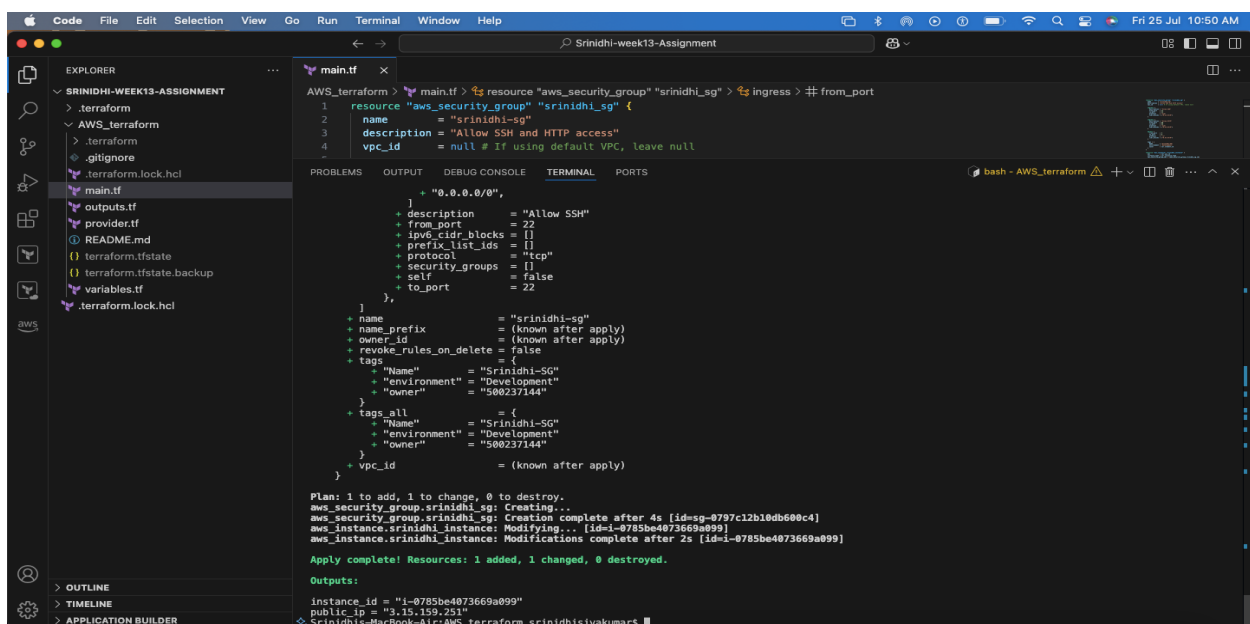
```
1 resource "aws_security_group" "sridinhi_sg" {
2   name       = "sridinhi-sg"
3   description = "Allow SSH and HTTP access"
4   vpc_id     = null # If using default VPC, leave null
5 }
6
7 ingress {
8   from_port = 80
9   to_port   = 80
10  protocol  = "tcp"
11  security_groups = []
12  self       = false
13 }
14
15 cidr_blocks = [
16   "0.0.0.0/0",
17 ]
18
19 description = "Allow SSH"
20 from_port   = 22
21 ipv6_cidr_blocks = []
22 prefix_list_ids = []
23 protocol      = "tcp"
24 security_groups = []
25 self          = false
26 to_port       = 22
27 }
28
29 name       = "sridinhi-sg"
30 name_prefix = (known after apply)
31 owner_id   = (known after apply)
32 revoke_rules_on_delete = false
33 tags       = {
34   "Name"      = "Srinidhi-SG"
35   "environment" = "Development"
36   "owner"     = "S00237144"
37 }
38 tags_all   = {
39   "Name"      = "Srinidhi-SG"
40   "environment" = "Development"
41   "owner"     = "S00237144"
42 }
43 vpc_id     = (known after apply)
44 }
```

Plan: 1 to add, 1 to change, 0 to destroy.

## 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*



The screenshot shows the VS Code interface with the 'main.tf' file open. The terminal pane at the bottom shows the output of the 'terraform apply -auto-approve' command. The output indicates that the security group was successfully created and attached to the existing EC2 instance. The plan shows 1 to add, 1 to change, and 0 to destroy.

```
Plan: 1 to add, 1 to change, 0 to destroy.
aws_security_group.sridinhi_sg: Creating...
aws_security_group.sridinhi_sg: Creation complete after 4s [id=sg-0797c12b10db600c4]
aws_instance.sridinhi_instance: Modifying... [id=i-0785be4073669a899]
aws_instance.sridinhi_instance: Modifications complete after 2s [id=i-0785be4073669a899]

Apply complete! Resources: 1 added, 1 changed, 0 destroyed.

Outputs:
instance_id = "i-0785be4073669a899"
public_ip  = "3.15.159.251"
Srinidhis-MacBook-Air:AWS_terraform sridinhisivakumar$
```

## 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:

The screenshot shows the AWS Management Console for the instance **i-0785be4073669a099** (Srinidhi-Terraform-Instance). The **Security** tab is selected, displaying the following details:

- Security details:** IAM Role is **None**, Owner ID is **730355603225**, and Launch time is **Fri Jul 25 2025 10:19:57 GMT-0400 (Eastern Daylight Time)**.
- Security groups:** **sg-0a31e92b356885009 (default)** is attached.
- Inbound rules:** A single rule is listed with Name **sg-05154f1771064607**, Security group rule ID **sg-05154f1771064607**, Port range **All**, Protocol **All**, Source **sg-0a31e92b356885009**, and Description **default**.
- Outbound rules:** A single rule is listed with Name **sg-02a20b5c98588476e**, Security group rule ID **sg-02a20b5c98588476e**, Port range **All**, Protocol **All**, Destination **0.0.0.0/0**, and Description **default**.

After:

The screenshot shows the AWS Management Console for the instance **i-0785be4073669a099** (Srinidhi-Terraform-Instance). The **Security** tab is selected, displaying the following details:

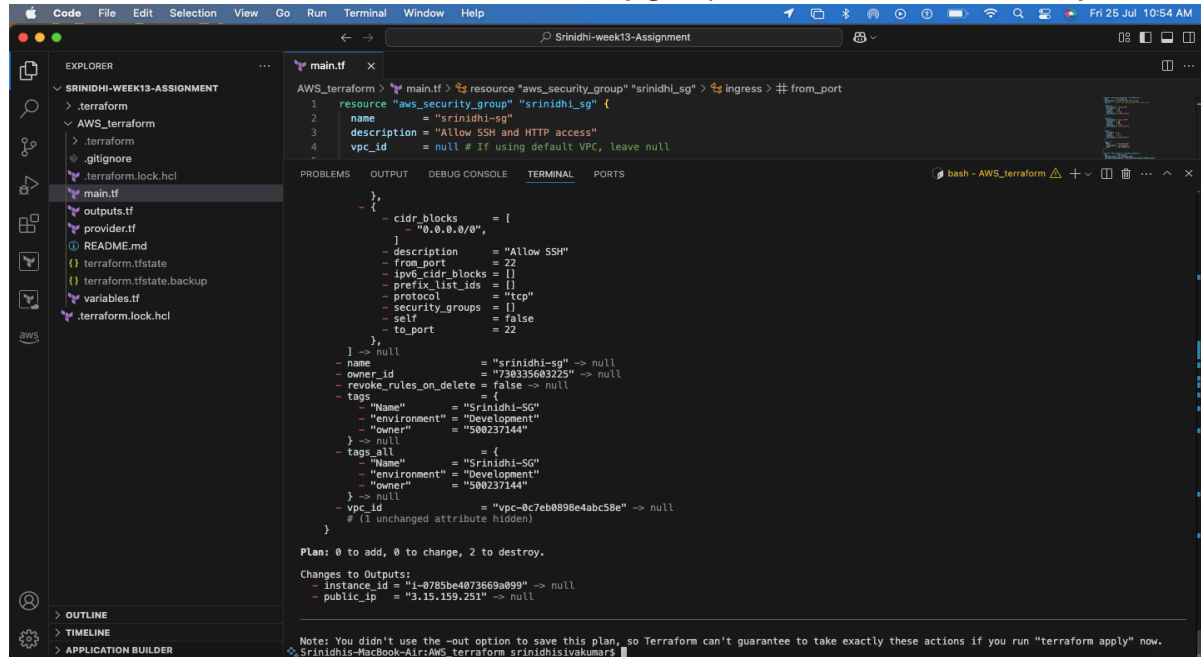
- Security details:** IAM Role is **None**, Owner ID is **730355603225**, and Launch time is **Fri Jul 25 2025 10:19:57 GMT-0400 (Eastern Daylight Time)**.
- Security groups:** **sg-0797c12b10db600c4 (srinidhi-sg)** is attached.
- Inbound rules:** Two rules are listed:
  - Rule 1: Name **sg-0a3a729556143fa24**, Security group rule ID **sg-0a3a729556143fa24**, Port range **80**, Protocol **TCP**, Source **0.0.0.0/0**, Security groups **srinidhi-sg**, and Description **Allow HTTP**.
  - Rule 2: Name **sg-0992873cad770ba0**, Security group rule ID **sg-0992873cad770ba0**, Port range **22**, Protocol **TCP**, Source **0.0.0.0/0**, Security groups **srinidhi-sg**, and Description **Allow SSH**.
- Outbound rules:** A single rule is listed with Name **sg-0c2745efe48da909**, Security group rule ID **sg-0c2745efe48da909**, Port range **All**, Protocol **All**, Destination **0.0.0.0/0**, Security groups **srinidhi-sg**, and Description **None**.

## 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.



The screenshot shows the VS Code interface with the Explorer pane on the left displaying the project structure for 'SRINIDHI-WEEK13-ASSIGNMENT'. The main editor shows the 'main.tf' file with Terraform configuration for an AWS security group. The 'PROBLEMS' pane is empty, and the 'OUTPUT' pane shows the plan output. The 'TERMINAL' pane shows the command 'terraform plan -destroy' being executed. The plan output indicates that the EC2 instance and security group will be destroyed.

```
main.tf
1 resource "aws_security_group" "srinidhi_sg" {
2   ingress = [
3     {
4       from_port = 22
5       to_port   = 22
6       protocol  = "tcp"
7       security_groups = [
8         "sg-0c7eb0898e4abc58e"
9       ]
10    }
11  ]
12  name        = "srinidhi-sg"
13  description = "Allow SSH and HTTP access"
14  vpc_id      = null # If using default VPC, leave null
15}
```

```
Plan: 0 to add, 0 to change, 2 to destroy.
Changes to Outputs:
  instance_id = "i-0785be4073669a099" -> null
  public_ip   = "3.15.159.251" -> null
```

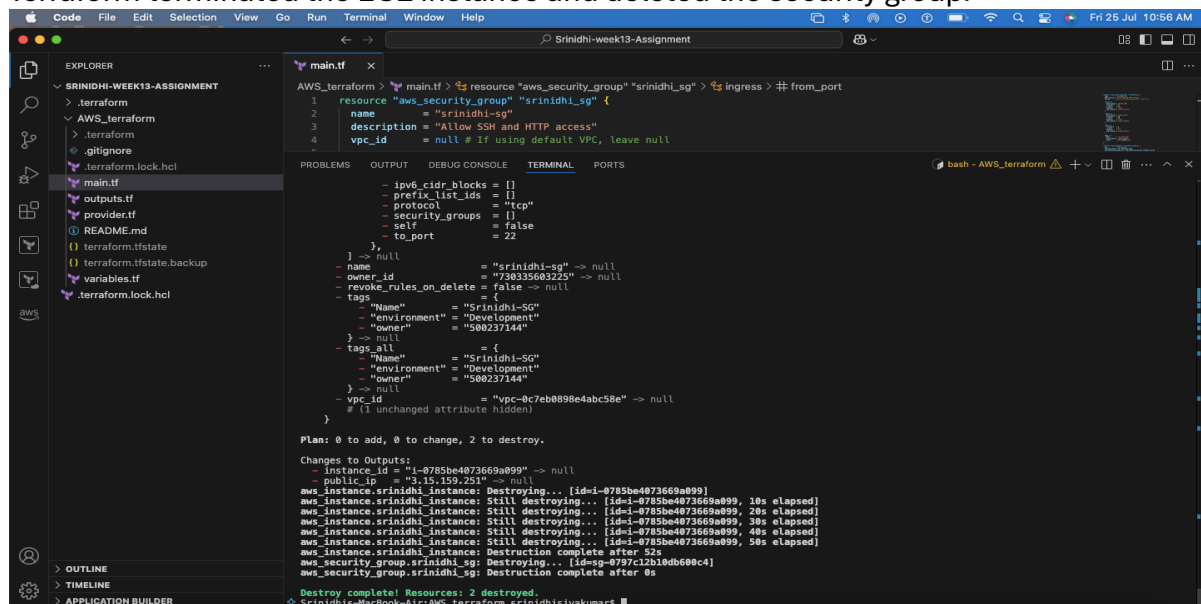
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

## 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.



The screenshot shows the VS Code interface with the Explorer pane on the left displaying the project structure for 'SRINIDHI-WEEK13-ASSIGNMENT'. The main editor shows the 'main.tf' file with Terraform configuration for an AWS security group. The 'PROBLEMS' pane is empty, and the 'OUTPUT' pane shows the plan output. The 'TERMINAL' pane shows the command 'terraform destroy -auto-approve' being executed. The output indicates that the EC2 instance and security group were successfully destroyed.

```
main.tf
1 resource "aws_security_group" "srinidhi_sg" {
2   ingress = [
3     {
4       from_port = 22
5       to_port   = 22
6       protocol  = "tcp"
7       security_groups = [
8         "sg-0c7eb0898e4abc58e"
9       ]
10    }
11  ]
12  name        = "srinidhi-sg"
13  description = "Allow SSH and HTTP access"
14  vpc_id      = null # If using default VPC, leave null
15}
```

```
Plan: 0 to add, 0 to change, 2 to destroy.
Changes to Outputs:
  instance_id = "i-0785be4073669a099" -> null
  public_ip   = "3.15.159.251" -> null

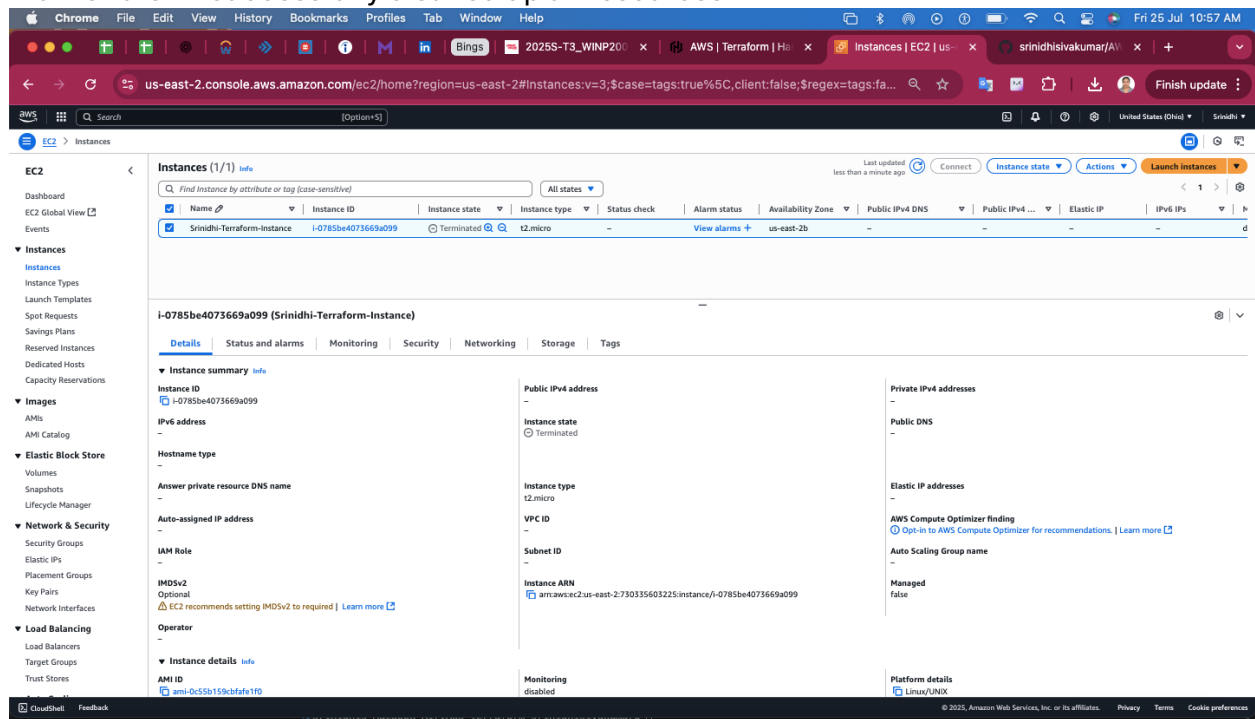
aws_instance.srinidhi_instance: Destroying... [id=i-0785be4073669a099]
aws_instance.srinidhi_instance: Still destroying... [id=i-0785be4073669a099, 18s elapsed]
aws_instance.srinidhi_instance: Still destroying... [id=i-0785be4073669a099, 28s elapsed]
aws_instance.srinidhi_instance: Still destroying... [id=i-0785be4073669a099, 38s elapsed]
aws_instance.srinidhi_instance: Still destroying... [id=i-0785be4073669a099, 48s elapsed]
aws_instance.srinidhi_instance: Still destroying... [id=i-0785be4073669a099, 58s elapsed]
aws_instance.srinidhi_instance: Destruction complete after 52s
aws_security_group.srinidhi_sg: Destroying... [id=sg-0c7eb0898e4abc58e]
aws_security_group.srinidhi_sg: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```



## 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.



The screenshot displays the AWS Management Console interface for the EC2 Instances page. The browser address bar shows the URL: `us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#instances:v=3,$case=tags:true%5C,client:false,$regex=tags:fa...`. The console header includes the AWS logo, a search bar, and the user's name 'Srinidhi'. The left sidebar shows the navigation menu with categories like EC2, Instances, Images, Elastic Block Store, Network & Security, and Load Balancing. The main content area shows a list of instances with the following columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4, Elastic IP, and IPv6 IPs. A single instance is listed: 'Srinidhi-Terraform-Instance' with ID 'i-0785be4073669a099', state 'Terminated', type 't2.micro', and availability zone 'us-east-2b'. Below the list, the details for the selected instance are shown, including the Instance summary, Instance ID, IP addresses, Hostname type, Answer private resource DNS name, Auto-assigned IP address, IAM Role, IMDSv2 status, Operator, Instance details, AMI ID, Public IPv4 address, Instance state, Instance type, VPC ID, Subnet ID, Instance ARN, Private IPv4 addresses, Public DNS, Elastic IP addresses, AWS Compute Optimizer finding, Auto Scaling Group name, Managed status, and Platform details.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4	Elastic IP	IPv6 IPs
Srinidhi-Terraform-Instance	i-0785be4073669a099	Terminated	t2.micro	-	-	us-east-2b	-	-	-	-

**Instance summary**

**Instance ID**  
i-0785be4073669a099

**IPV6 address**  
-

**Hostname type**  
-

**Answer private resource DNS name**  
-

**Auto-assigned IP address**  
-

**IAM Role**  
-

**IMDSv2**  
Optional  
EC2 recommends setting IMDSv2 to required | [Learn more](#)

**Operator**  
-

**Instance details**

**AMI ID**  
ami-0c55b159cbfafa1f0

**Public IPv4 address**  
-

**Instance state**  
Terminated

**Instance type**  
t2.micro

**VPC ID**  
-

**Subnet ID**  
-

**Instance ARN**  
arn:aws:ec2:us-east-2:730135603225:instance/i-0785be4073669a099

**Private IPv4 addresses**  
-

**Public DNS**  
-

**Elastic IP addresses**  
-

**AWS Compute Optimizer finding**  
Opt-in to AWS Compute Optimizer for recommendations. | [Learn more](#)

**Auto Scaling Group name**  
-

**Managed**  
false

**Platform details**  
Linux/UNIX