

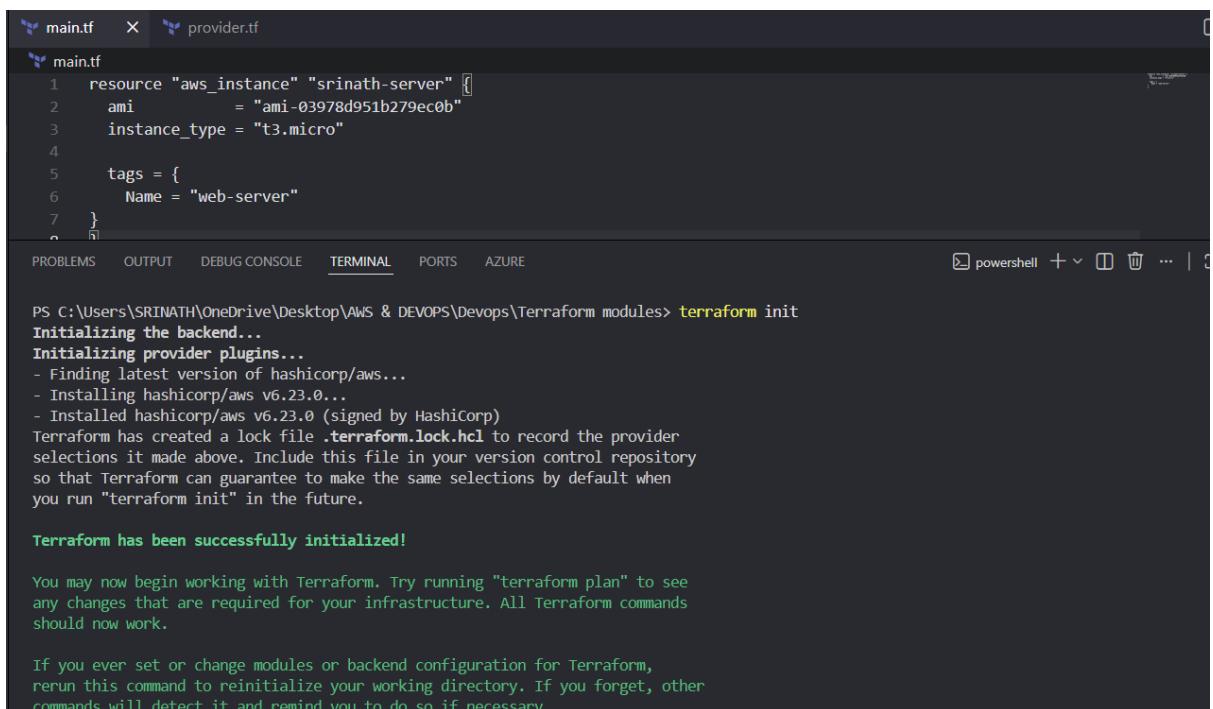
## Task 6

### Terraform:

- Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp that allows users to define and provision cloud and on-premises resources using a declarative configuration language.
- It enables users to manage infrastructure throughout its lifecycle by writing human-readable configuration files that can be versioned, reused, and shared.

### Terraform Command:

1) terraform init:



The screenshot shows a terminal window in a dark-themed IDE interface. The terminal tab is active, displaying the command 'terraform init' being run in a PowerShell session. The output shows the initialization process, including the download and installation of provider plugins for AWS. A success message at the end indicates that Terraform has been initialized successfully, and it suggests running 'terraform plan' to see changes.

```
PS C:\Users\SRINATH\OneDrive\Desktop\AWS & DEVOPS\Devops\Terraform modules> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.23.0...
- Installed hashicorp/aws v6.23.0 (signed by HashiCorp)
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.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.
```

2) terraform plan:

The screenshot shows a code editor with two tabs: 'main.tf' and 'provider.tf'. The 'main.tf' tab contains the following Terraform code:

```
resource "aws_instance" "srinath-server" [
  ami          = "ami-03978d951b279ec0b"
  instance_type = "t3.micro"
  tags = {
    Name = "web-server"
  }
]
```

The 'TERMINAL' tab shows a PowerShell window with the command `terraform plan` executed in the directory `C:\Users\SRINATH\OneDrive\Desktop\AWS & DEVOPS\Devops\Terraform modules`. The output indicates that Terraform will create a new AWS instance named 'srinath-server'.

```
PS C:\Users\SRINATH\OneDrive\Desktop\AWS & DEVOPS\Devops\Terraform modules> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.srinath-server will be created
+ resource "aws_instance" "srinath-server" {
    + ami
    + arn
    + associate_public_ip_address
    + availability_zone
    + disable_api_stop
    + disable_api_termination
    + ebs_optimized
    + enable_primary_ipv6
    + force_destroy
    + get_password_data
    + host_id
    + host_resource_group_arn
    + iam_instance_profile
    + id
    + instance_initiated_shutdown_behavior
    + instance_lifecycle
    + instance_state
    + instance_type
    + ipv6_address_count
    + ip_address_count
    + subnet_id
}
```

### 3) terraform apply

The screenshot shows a Visual Studio Code interface with two tabs open: 'main.tf' and 'provider.tf'. The 'main.tf' tab contains the following Terraform code:

```
resource "aws_instance" "srinath-server" {
  ami           = "ami-03978d951b279ec0b"
  instance_type = "t3.micro"

  tags = {
    Name = "web-server"
  }
}
```

The 'TERMINAL' tab is active, showing a PowerShell window with the command `terraform apply` entered. Below the terminal, the output of the Terraform plan is displayed:

```
PS C:\Users\SRINATH\OneDrive\Desktop\AWS & DEVOPS\Devops\Terraform modules> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.srinath-server will be created
+ resource "aws_instance" "srinath-server" {
    + ami           = "ami-03978d951b279ec0b"
    + arn          = (known after apply)
    + associate_public_ip_address = (known after apply)
    + availability_zone      = (known after apply)
    + disable_api_stop       = (known after apply)
    + disable_api_termination = (known after apply)
    + ebs_optimized          = (known after apply)
    + enable_primary_ipv6    = (known after apply)
    + force_destroy          = false
    + get_password_data      = false
    + host_id                = (known after apply)
    + host_resource_group_arn = (known after apply)
    + iam_instance_profile   = (known after apply)
    + id                     = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
    + instance_lifecycle     = (known after apply)
    + instance_state          = (known after apply)
```

# Output

The screenshot shows the AWS EC2 Instances page. The left sidebar has 'EC2' selected under 'Instances'. The main area displays a table titled 'Instances (1) Info' with one row. The row details a single instance named 'web-server' with the following specifications:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
web-server	i-048811b0ee167bec3	Running	t3.micro	Initializing	View alarms +	us-west-1a	ec2-54-1...

Below the table, a section titled 'Select an instance' is visible. The bottom of the screen includes standard AWS navigation links like CloudShell, Feedback, and Console Mobile App, along with copyright information and links for Privacy, Terms, and Cookie preferences.