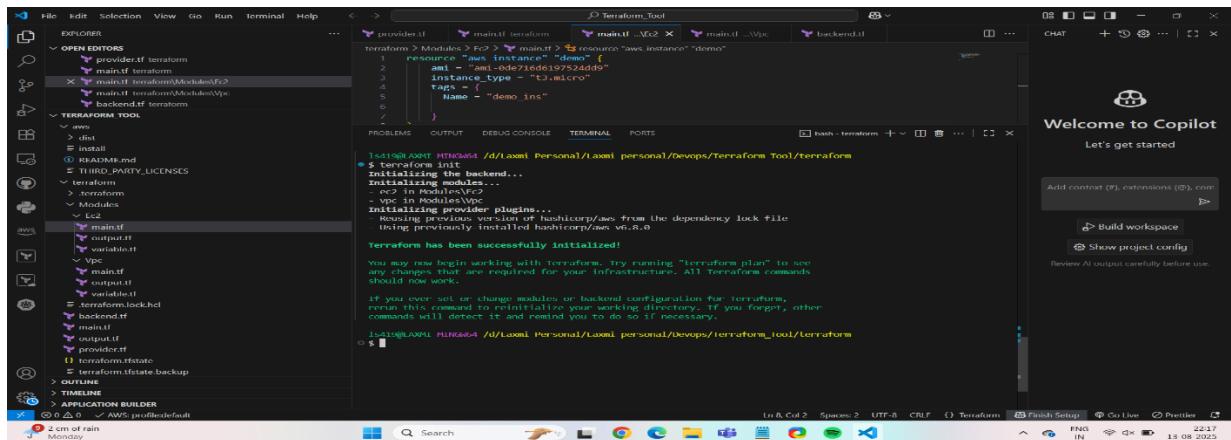


Name :- Laxmi Swami

Topic :- Terraform Variables

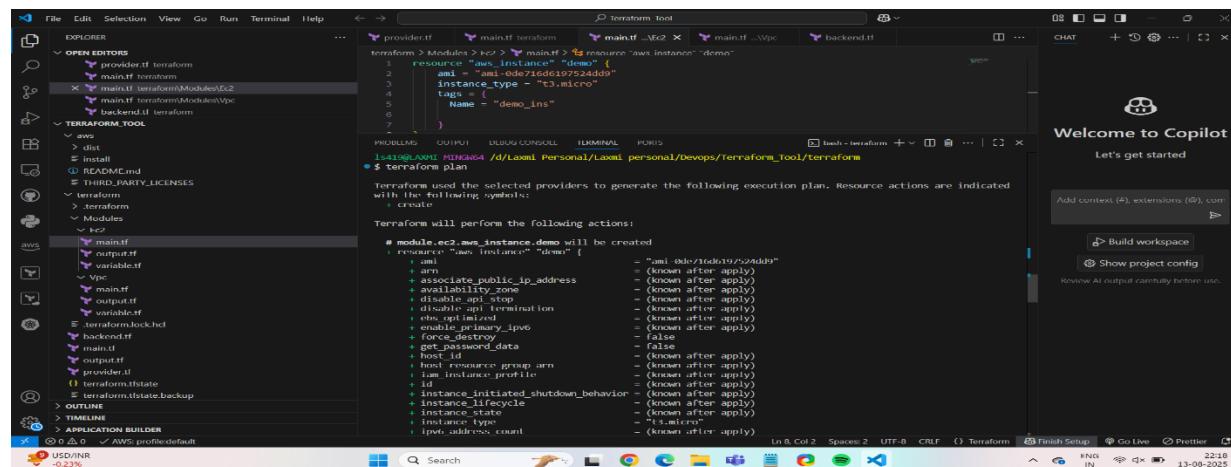
STEP 1 :- First fetch terraform init command as we have destroy everything in last task.



The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the command 'terraform init' being run, followed by its output. The output includes messages about initializing modules and provider plugins, and a success message: 'Terraform has been successfully initialized!'. A note at the bottom of the terminal window suggests running 'terraform init' again if changes are made to modules or backend configurations.

```
15419@LAXMI:~/Desktop/Laxmi Personal/Laxmi personal/Devops/Terraform Tool/terraform$ terraform init
Initializing provider plugins...
- using source 'hashicorp/aws' from version lock file
- using provider 'aws' v3.0.0
Terraform has been successfully initialized!
```

STEP 2 :- Now fetch the command **terraform plan**.



The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the command 'terraform plan' being run, followed by its output. The output shows the execution plan, indicating actions such as creating an AWS instance and setting its IP address. Terraform also lists various resource types and their properties.

```
15419@LAXMI:~/Desktop/Laxmi Personal/Laxmi personal/Devops/Terraform Tool/terraform$ 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:
# module.ec2.aws_instance.demo.will be created
+ resource "aws_instance" "demo" {
  ami = "ami-0de716d6197524dd9"
  instance_type = "t3.micro"
  tags = [
    {
      Name = "demo_ins"
    }
  ]
}
```

STEP 3 :- Now fetch the command **terraform apply**.

The screenshot shows the Terraform Tool interface with the following details:

- EXPLORER** pane: Shows files like provider.tf, main.tf, and backend.tf.
- PROBLEMS** pane: Displays the command: `14:19:56 AM PT [INFO] 4474/TERRAFORM /d/Laxmi Personal/Laxmi personal/devops/Terraform Tool/terraform`.
- TERRAFORM** pane: Shows the execution plan with the following actions:

```

resource "aws_instance" "demo" {
  ami           = "ami-0dc716d619742add9"
  ami_id        = "ami-0dc716d619742add9"
  associate_public_ip_address = true
  availability_zone = "us-east-1a"
  disable_api_termination = true
  ebs_optimized = false
  eni_index     = 0
  force_destroy = false
  get_password_data = true
  host_resource_groupArn = null
  iam_instance_profile = null
  id            = "i-08ef7d622d99c6718"
  instance_initiated_shutdown_behavior = "stop"
  instance_lifecycle = "auto"
  instance_state = "running"
  instance_type = "t3.micro"
}
  
```

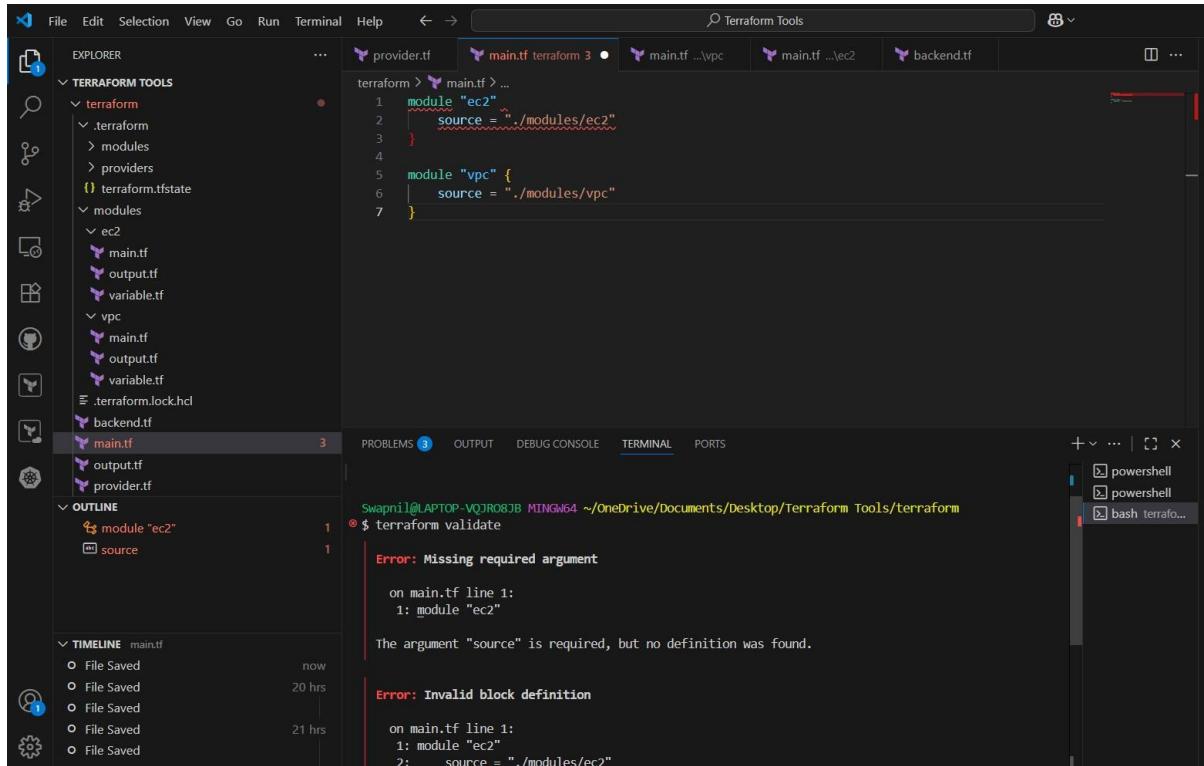
The plan indicates that the `aws_instance.demo` resource will be created.

STEP 4 :- Now created instance , vpc, subnet

The screenshots show the AWS Management Console interface across three different sections:

- EC2 > Instances**: Shows the creation of an EC2 instance named `demo_Ins` (t3.micro, Running). The instance ID is `i-08ef7d622d99c6718`.
- VPC > Your VPCs**: Shows the creation of a VPC named `My-Vpc`. The VPC ID is `vpc-0490878c502185d6a`, and its IPv4 CIDR is `172.31.0.0/16`.
- VPC > Subnets**: Shows the creation of two subnets: `MySubnet1` and `MySubnet2`. The subnet IDs are `subnet-0b61b75bd4e5a85ec` and `subnet-0830fc7bc4e39e4dd` respectively, both associated with the VPC `vpc-0490878c502185d6a`.

STEP 5 :- If we delete something in main.tf and fetch **terraform validate**. It will show where is error which line, column and also give a suggestion how to solve this error.



The screenshot shows the VS Code interface with the Terraform Tools extension. The Explorer sidebar shows files like provider.tf, main.tf, and backend.tf. The main editor window displays a portion of main.tf:

```

1 module "ec2"
2   source = "./modules/ec2"
3 }
4
5 module "vpc" {
6   source = "./modules/vpc"
7 }

```

The Problems panel shows two errors from running `terraform validate`:

- Error: Missing required argument**
on main.tf line 1:
1: module "ec2"
- Error: Invalid block definition**
on main.tf line 1:
1: module "ec2"
2: source = "./modules/ec2"

The terminal below shows the command and the validation results:

```

Swapnil@LAPTOP-VQJRO8JB MINGW64 ~/OneDrive/Documents/Desktop/Terraform Tools/terraform
$ terraform validate

```

The output shows the validation errors:

```

Error: Missing required argument
on main.tf line 1:
1: module "ec2"

The argument "source" is required, but no definition was found.

Error: Invalid block definition
on main.tf line 1:
1: module "ec2"
2:   source = "./modules/ec2"

```




The screenshot shows the VS Code interface with the Terraform Tools extension. The Explorer sidebar shows files like provider.tf, main.tf, and backend.tf. The main editor window displays a portion of main.tf:

```

1 module "ec2"
2   source = "./modules/ec2"
3 }
4
5 module "vpc" {
6   source = "./modules/vpc"
7 }

```

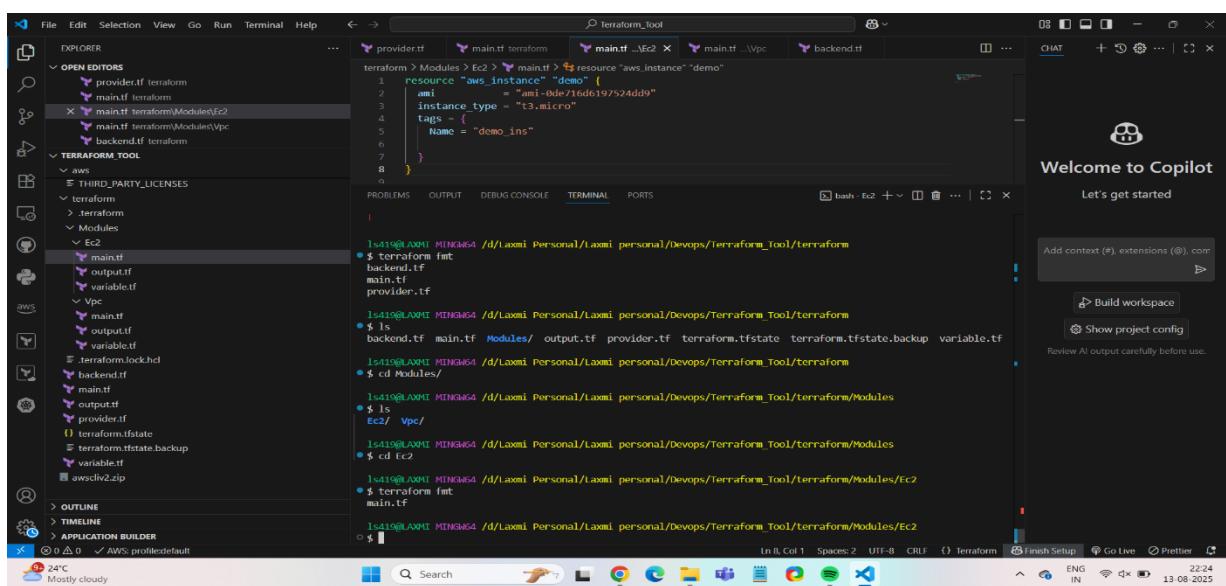
The Problems panel shows a single success message from running `terraform validate`:

```

Swapnil@LAPTOP-VQJRO8JB MINGW64 ~/OneDrive/Documents/Desktop/Terraform Tools/terraform
$ terraform validate
Success! The configuration is valid.

```

STEP 6 :- If our format is not as per standard and we fetch **terraform fmt** command we will get the required format automatically.



The screenshot shows the VS Code interface with the Terraform Tools extension. The Explorer sidebar shows files like provider.tf, main.tf, and backend.tf. The main editor window displays a portion of main.tf:

```

1 resource "aws_instance" "demo" {
2   ami           = "ami-0de710d6197524dd9"
3   instance_type = "t3.micro"
4   tags          = {
5     Name = "demo_ins"
6   }
7 }

```

The terminal below shows the command and the formatted output:

```

1s@192LAVNI MINGW64 ~/d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$ terraform fmt
backend.tf
main.tf
provider.tf

1s@192LAVNI MINGW64 ~/d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$ ls
backend.tf  main.tf  provider.tf  terraform.tstate  terraform.tstate.backup  variable.tf

1s@192LAVNI MINGW64 ~/d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$ cd Modules/
1s@192LAVNI MINGW64 ~/d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform/Modules
$ ls
Ec2/  Vpc/

1s@192LAVNI MINGW64 ~/d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform/Modules
$ cd Ec2/
1s@192LAVNI MINGW64 ~/d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform/Modules/Ec2
$ terraform fmt
main.tf

1s@192LAVNI MINGW64 ~/d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform/Modules/Ec2
$ ls
main.tf

```

STEP 7 :- For variable go to ec2 folder then main.tf and we will change the script as below

```

resource "aws_instance" "sample" {
  ami = "ami-ID"
  instance_type = "var.instance_type"
  tags = {
    Name = "Sample-Instance"
  }
}

```

```

resource "aws_instance" "demo" {
  ami           = "ami-0de716ddc793add0"
  instance_type = var.instance_type
  tags = {
    Name = "demo-ins"
  }
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

module.ec2.aws_instance.demo: Modifying... [id=i-08ef7d622d99c6718]
 module.ec2.aws_instance.demo: Still modifying... [id=i-08ef7d622d99c6718, 00m10s elapsed]
 module.ec2.aws_instance.demo: Still modifying... [id=i-08ef7d622d99c6718, 00m26s elapsed]

Error: updating EC2 Instance (i-08ef7d622d99c6718) type: modifying EC2 Instance (i-08ef7d622d99c6718) InstanceType (var.instance_type) attribute: operation error EC2: ModifyInstanceStateAttribute, https response error StatusCode 400, StatusMessage InvalidParameterValue: The following supplied instance types do not exist: [var.instance_type]

with module.ec2.aws_instance.demo,
 on Modules/Ec2/main.tf line 1, in resource "aws_instance" "demo":
 1: resource "aws_instance" "demo" {

STEP 8 :- Now if we change type from t2.small to t3.micro and initiate the terraform plan we will get 1 change for same instance.

```

variable "instance_type" {
  type = string
  default = "t3.micro"
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[1x19@0:2001] HINDE64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform

\$ cd /terraform
 bash: cd: /terraform: No such file or directory

[1x19@0:2001] HINDE64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool

\$ cd terraform/

[1x19@0:2001] HINDE64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform

\$ ls

backend.tf main.tf output.tf provider.tf terraform.tstate terraform.tfstate.backup variable.tf

[1x19@0:2001] HINDE64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform

```

variable "instance_type" {
  type = string
  default = "t2.small"
}

```

The screenshot shows the VS Code interface with the Terraform extension active. The left sidebar shows the project structure with files like provider.tf, main.tf, variable.tf, and backend.tf. The main editor shows the variable.tf file with the above code. The terminal tab shows the command line output of running 'terraform plan' in the root directory.

STEP 9 :- Now go to terraform folder terraform plan.

```

$ terraform plan
module.vpc.aws_vpc.my-vpc: Refreshing state... [id=vpc-e13fe09ff95ce9e03]
module.vpc.aws_subnet.my-subnet1: Refreshing state... [id=subnet-0b61b79d45a85ec]
module.vpc.aws_subnet.my-subnet2: Refreshing state... [id=subnet-08976fe4ab0826361]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- update in-place

Terraform will perform the following actions:

# module.ec2.aws_instance.demo will be updated in-place
- resource "aws_instance" "demo" {
    id: ...
    ~ instance_type: "t2.micro" -> "var.instance_type"
    ~ public_dns: ...
    ~ public_ip: ...
    tags: ...
    ~ "Name": "demo_ins"
  }
  # (36 unchanged attributes hidden)
}

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

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.

$ 

```

This screenshot shows the VS Code interface again, but the terminal tab now displays the output of the 'terraform plan' command. It shows the execution plan for the 'aws_ec2_instance' resource, indicating an update in-place.

```

$ terraform plan
# module.ec2.aws_instance.demo will be updated in-place
- resource "aws_instance" "demo" {
    id: ...
    ~ instance_type: "t2.micro" -> "var.instance_type"
    ~ public_dns: ...
    ~ public_ip: ...
    tags: ...
    ~ "Name": "demo_ins"
  }
  # (36 unchanged attributes hidden)
}

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- update in-place

Terraform will perform the following actions:

# module.ec2.aws_instance.demo will be updated in-place
- resource "aws_instance" "demo" {
    id: ...
    ~ instance_type: "t2.micro" -> "var.instance_type"
    ~ public_dns: ...
    ~ public_ip: ...
    tags: ...
    ~ "Name": "demo_ins"
  }
  # (36 unchanged attributes hidden)
}

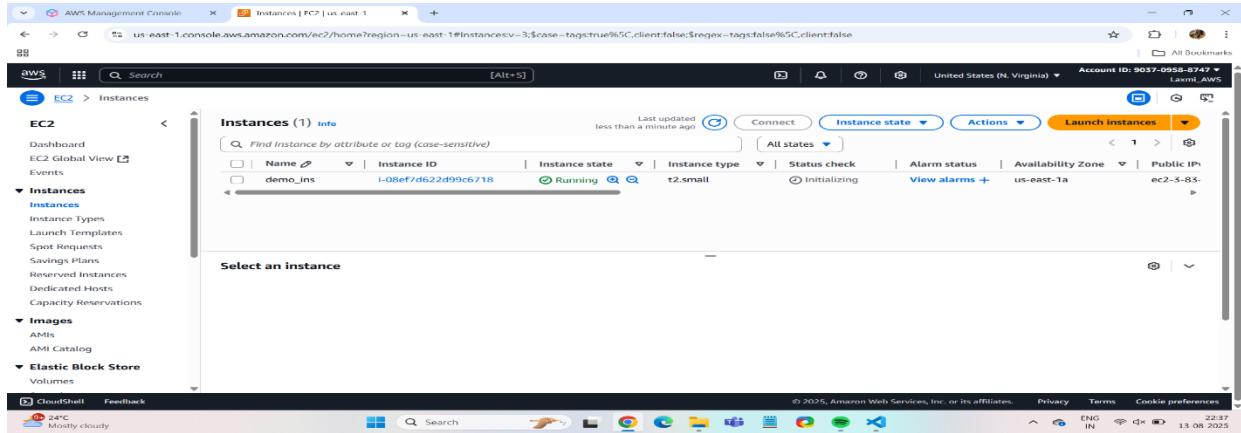
$ 

```

This screenshot shows the VS Code interface again, but the terminal tab now displays the output of the 'terraform plan' command. It shows the execution plan for the 'aws_ec2_instance' resource, providing more detailed information about the resources involved.

Now go to same folder's variable.tf file and write below script.

```
variable "instance_type" {
    type = string
    default = "t2.small"
}
```



STEP 10 :- Now go to create variable variable

```
"instance_type" {
    type = string
    default = ""
}
```

- Here we have not written any instance type so when we execute terraform plan or terraform apply command we have to fetch instance type.
- We will get instance type as we fetch it while terraform apply command.

```

provider.tf
main.tf
variable.tf
main.tf_ec2
variable.tf_ec2
main.tf_vpc
backend.tf
variable.tf_vpc

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

```

tags = [
  "Name" = "demo_ins"
]
# (36 unchanged attributes hidden)
# (8 unchanged blocks hidden)
}

```

Do you want to perform these actions?
Terraform will perform the actions described above.

bash - terraform

File Edit Selection View Go Run Terminal Help

File Edit Selection View Go Run Terminal Help

24°C Mostly cloudy

STEP 11 :- Now go to main.tf in terraform folder and mention instance_type same as we mention in ec2 folder.

```

provider.tf
main.tf
variable.tf
main.tf_ec2
variable.tf_ec2
main.tf_vpc
backend.tf
variable.tf_vpc

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

```

tags = [
  "Name" = "demo_ins"
]
# (36 unchanged attributes hidden)
# (8 unchanged blocks hidden)
}

```

Do you want to perform these actions?
Terraform will perform the actions described above.

bash - terraform

File Edit Selection View Go Run Terminal Help

File Edit Selection View Go Run Terminal Help

24°C Mostly cloudy

STEP 12 :- In terraform > variable.tf file we will not give any instance type.

```

provider.tf
main.tf
variable.tf
main.tf_ec2
variable.tf_ec2
main.tf_vpc
backend.tf
variable.tf_vpc

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

```

tags = [
  "Name" = "demo_ins"
]
# (37 unchanged attributes hidden)
# (8 unchanged blocks hidden)
}

```

Do you want to perform these actions?
Terraform will perform the actions described above.

bash - terraform

File Edit Selection View Go Run Terminal Help

File Edit Selection View Go Run Terminal Help

24°C Mostly cloudy

STEP 13 :- When we run terraform plan command we will have to fetch the value. Similarly we have to fetch it when we fetch terraform apply and terraform destroy command.

```

variable "instance_type" {
  type = string
}

```

```

Only 'yes' will be accepted to approve.

Enter a value: yes

```

```

module.ec2.aws_instance.demo: Modifying... [id=1-08ef7d622d99c6718]
module.ec2.aws_instance.demo: Modifications complete after 9s [id=1-08ef7d622d99c6718]

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

```

13:45@LAXMI MINGW64 ~/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform

\$ terraform plan

var.instance_type

Enter a value: t3.micro

STEP 14 :- Now create one staging.tfvars file and mention t2.micro as a instance type here.

```

variable "instance_type" {
  type = string
  value = "t2.micro"
}

```

```

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these

```

13:45@LAXMI MINGW64 ~/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform

\$ terraform plan

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

STEP 15 :- Now create one production.tfvars file and mention t2.medium as a instance type here.

The screenshot shows the VS Code interface with the Terraform Tool extension open. In the Explorer sidebar, there is a new file named 'production.tfvars' under the 'TERRAFORM_TOOL' section. The code editor shows the following content:

```
provider.tf  main.tf  terraform  staging.tfvars  production.tfvars  variable.tf  terraform
1  instance_type = "t2.medium"
```

The terminal tab shows the command line output of the current session:

```
ls
backedn.tf  main.tf  Modules/  output.tf  provider.tf  terraform.tfstate  terraform.tfstate.backup  variable.tf
ls@LAXMI MINGW64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$ touch staging.tfvars production.tfvars
ls@LAXMI MINGW64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$ ls
backedn.tf  Modules/  production.tfvars  staging.tfvars  terraform.tfstate.backup
main.tf  output.tf  provider.tf  terraform.tfstate  variable.tf
ls@LAXMI MINGW64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$ [ ]
```

STEP 16 :- Now fetch **terraform plan -var-file="staging.tfvars"** check the instance type it will be t2.micro.

The screenshot shows the VS Code interface with the Terraform Tool extension open. The terminal tab displays the output of the 'terraform plan -var-file="staging.tfvars"' command:

```
ls@LAXMI MINGW64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$ terraform plan -var-file="staging.tfvars"
module.vpc.aws_vpc.my-Vpc: Refreshing state... [id=vpc-e13f05ff95ce9a93]
module.ec2.aws_instance.demo: Refreshing state... [id=e8ef7d622d99e6718]
module.vpc.aws_subnet.my-Subnet2: Refreshing state... [id=subnet-8b61b75bd045a85ec]
module.vpc.aws_subnet.my-Subnet1: Refreshing state... [id=subnet-08576fe4ab8826361]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
~ update in-place

Terraform will perform the following actions:

# module.ec2.aws_instance.demo will be updated in-place
~ resource "aws_instance" "demo" {
    id          = "e8ef7d622d99e6718"
    instance_type = "t2.micro" -> "t2.micro"
    public_dns  = "ec2-3-82-269-195.compute-1.amazonaws.com" -> (known after apply)
    public_ip   = "3.82.269.195" -> (known after apply)
    tags        = {
        "Name" = "demo_ins"
    }
    # (36 unchanged attributes hidden)

    # (8 unchanged blocks hidden)
}
```

The terminal also shows the plan summary and a note about the lack of a save option:

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

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.
```

STEP 17 :- Now fetch the command **terraform plan -varfile="production.tfvars"** here t2.small will be you're instance type.

```

ls@192:~/AXML MINGW64 ~/laxmi Personal/laxmi personal/Devops/Terraform_Tool/terraform
$ terraform plan -varfile=production.tfvars
module.vpc.aws_vpc.MyVpc: Refreshing state... [id=vpc-e13f059ff95ceca993]
module.ec2.aws_instance.demo: Refreshing state... [id=i-08ef7d622d99c6718]
module.vpc.aws_subnet.mySubnet2: Refreshing state... [id=subnet-0b61b75bde45a85ec]
module.vpc.aws_subnet.mySubnet1: Refreshing state... [id=subnet-08976fe4ab0826361]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
~ update in-place

Terraform will perform the following actions:

# module.ec2.aws_instance.demo will be updated in-place
~ resource "aws_instance" "demo" {
    id                               = "i-08ef7d622d99c6718"
    ~ instance_type                  = "t3.micro" -> "t2.medium"
    ~ public_dns                     = "ec2-3-82-209-195.compute-1.amazonaws.com" -> (known after apply)
    ~ public_ip                      = "3.82.209.195" -> (known after apply)
    tags                            = {
        "Name" = "demo_ins"
    }
    # (36 unchanged attributes hidden)
}

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

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.

```

STEP 18 :- Now fetch the command

terraform apply -var-file="staging.tfvars"

```

ls@192:~/AXML MINGW64 ~/laxmi Personal/laxmi personal/Devops/Terraform_Tool/terraform
$ terraform apply -varfile=staging.tfvars
module.vpc.aws_vpc.MyVpc: Refreshing state... [id=vpc-e13f059ff95ceca993]
module.ec2.aws_instance.demo: Refreshing state... [id=i-08ef7d622d99c6718]
module.vpc.aws_subnet.mySubnet2: Refreshing state... [id=subnet-08976fe4ab0826361]
module.vpc.aws_subnet.mySubnet1: Refreshing state... [id=subnet-0b61b75bde45a85ec]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
~ update in-place

Terraform will perform the following actions:

# module.ec2.aws_instance.demo will be updated in-place
~ resource "aws_instance" "demo" {
    id                               = "i-08ef7d622d99c6718"
    ~ instance_type                  = "t3.micro" -> "t2.micro"
    ~ public_dns                     = "ec2-3-82-209-195.compute-1.amazonaws.com" -> (known after apply)
    ~ public_ip                      = "3.82.209.195" -> (known after apply)
    tags                            = {
        "Name" = "demo_ins"
    }
    # (36 unchanged attributes hidden)
}

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

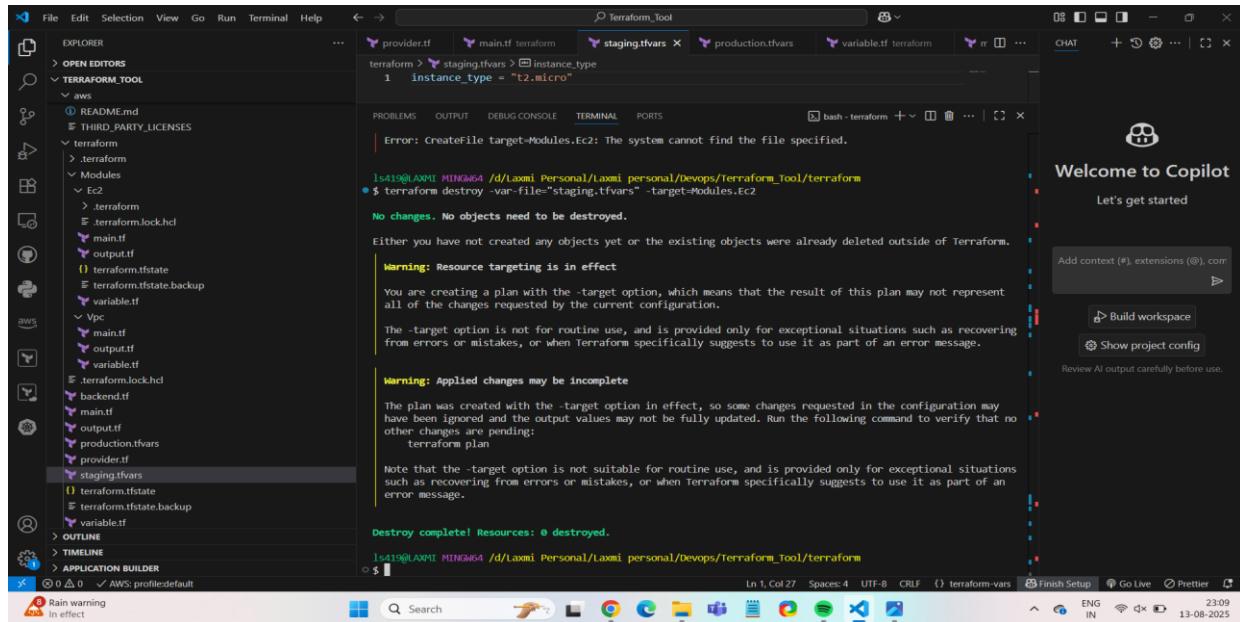
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

```

STEP 19 :- Now I will fetch the command **terraform destroy -varfile="staging.tfvars" -target=module.ec2**

It will delete all the data inside ec2 folder. i.e. instance. And vpc and subnet will remain as it is.



```
terraform > staging.tfvars > instance_type
1 instance_type = "t2.micro"

Error: CreateFile target\Modules.Ec2: The system cannot find the file specified.

15419@LAXMI MINGW64 /d/Laxmi Personal/laxmi personal/devops/terraform_tool/terraform
$ terraform destroy -var-file="staging.tfvars" -target=Module.Ec2
No changes. No objects need to be destroyed.

Either you have not created any objects yet or the existing objects were already deleted outside of Terraform.

Warning: Resource targeting is in effect

You are creating a plan with the -target option, which means that the result of this plan may not represent all of the changes requested by the current configuration.

The -target option is not for routine use, and is provided only for exceptional situations such as recovering from errors or mistakes, or when Terraform specifically suggests to use it as part of an error message.

Warning: Applied changes may be incomplete

The plan was created with the -target option in effect, so some changes requested in the configuration may have been ignored and the output values may not be fully updated. Run the following command to verify that no other changes are pending:
  terraform plan

Note that the -target option is not suitable for routine use, and is provided only for exceptional situations such as recovering from errors or mistakes, or when Terraform specifically suggests to use it as part of an error message.

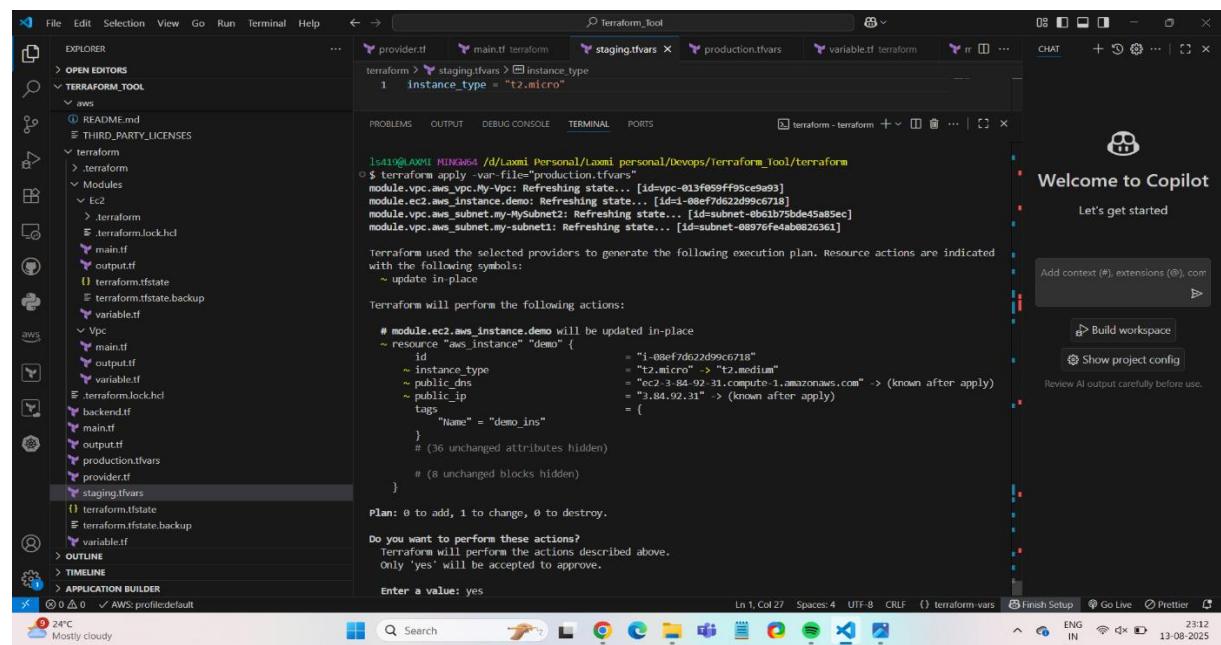
Destroy complete! Resources: 0 destroyed.

15419@LAXMI MINGW64 /d/Laxmi Personal/laxmi personal/devops/terraform_tool/terraform

```

STEP 20 :- Now fetch the command

terraform apply -var-file="production.tfvars" this command will create only one instance but will not create any vpc and if instance is already created in staging environment it will change instance type from t2.micro to t2.small.



```
terraform > staging.tfvars > instance_type
1 instance_type = "t2.micro"

15419@LAXMI MINGW64 /d/Laxmi Personal/laxmi personal/Devops/Terraform_Tool/terraform
$ terraform apply -var-file="production.tfvars"
module.vpc.aws_vpc.MyVpc: Refreshing state... [id=vpc-e13fe69ff95ce9e93]
module.ec2.aws_instance.demo: Refreshing state... [id=i-08ef7d622d99c6718]
module.vpc.aws_subnet.myMySubnet2: Refreshing state... [id=subnet-e6b1b79bd45a85ec]
module.vpc.aws_subnet.myMySubnet1: Refreshing state... [id=subnet-e887fe4a0026363]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  ~ update in place

Terraform will perform the following actions:

# module.ec2.aws_instance.demo will be updated in-place
~ resource "aws_instance" "demo" {
    id                               = "i-08ef7d622d99c6718"
    ~ instance_type                   = "t2.micro" -> "t2.medium"
    ~ public_dns                      = "ec2-3-84-92-31.compute-1.amazonaws.com" -> (known after apply)
    ~ public_ip                        = "3.84.92.31" -> (known after apply)
    tag["Name"] = "demo_ins"
}
# (36 unchanged attributes hidden)

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

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes
```

```

# module.ec2.aws_instance.demo will be updated in place
~ resource "aws_instance" "demo" {
    ~ instance_type = "t2.micro"
    ~ public_dns   = "ec2-3-84-92-31.compute-1.amazonaws.com" -> (known after apply)
    ~ public_ip    = "3.84.92.31" -> (known after apply)
    ~ tags         = [
        { "Name": "demo_ins" }
    ]
}

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

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

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

1:12:19 [LAXMI] MINGW64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_tool/terraform
$ terraform apply

```

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
demo_ins	i-08ef7d622d99c6718	Running	t2.medium	Initializing	View alarms	us-east-1a	ec2-5-87-
demo_ins	i-0511b2593997ce0cf	Terminated	m1.small	-	View alarms	us-east-1a	-

STEP 21 :- Now fetch the command

terraform destroy -var-file="staging.tfvars" It will destroy all the resources created.

```

1:14:19 [LAXMI] MINGW64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_tool/terraform
$ terraform destroy -var-file=staging.tfvars
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# module.ec2.aws_instance.demo will be destroyed
- resource "aws_instance" "demo" {
    - arn           = null
    - associate_public_ip_address = null
    - availability_zone      = null
    - ebs_optimized       = null
    - disable_api_termination = null
    - ebs_optimized       = null
    - force_destroy        = null
    - get_password_data   = null
    - ignore_errors        = null
    - id                = "i-08ef7d622d99c6718"
    - instance_initiated_shutdown_behavior = null
    - instance_type        = "t2.micro" -> null
    - ipv4_address_count   = null
    - ipv6_address_count   = null
    - monitoring          = null
    - placement_group       = null
    - primary_network_interface_id = "eni-000edda0a00ca0a" -> null
}

```

STEP 22 :- Now fetch the command

terraform destroy -var-file="production.tfvars" It won't delete anything as we have already deleted files by fetching the command in staging.tf and it will overwrite the same data and will show 0 file destroyed.

```
File Edit Selection View Go Run Terminal Help ← → Terraform_Tool
EXPLORER ... provider.tf main.tf terraform staging.tfvars production.tfvars variable.tf terraform ...
TERRAFORM_TOOL
aws README.md THIRD_PARTY_LICENSES
terraform
  terraform
    Modules
      Ec2
        terraform
          terraform.lock.hcl
        main.tf
        output.tf
        terraform.state
        terraform.state.backup
        variable.tf
      Vpc
        main.tf
        output.tf
        variable.tf
      .terraform.lock.hcl
        backend.tf
        main.tf
        output.tf
        production.tfvars
        provider.tf
        staging.tfvars
      terraform.state
      terraform.state.backup
      variable.tf
  OUTLINE
  TIMELINE
  APPLICATION_BUILDER
  0 0 AWS: profiledefault
  24°C Mostly cloudy
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - terraform + ... x
Welcome to Copilot
Let's get started
Add context (#), extensions (@), com ...
Build workspace
Show project config
Review AI output carefully before use.
In 1, Col 27 Spaces:4 UTF-8 CRLF {} terraform-vars Finish Setup Go Live Prettier
ENG IN 23:15 13-08-2025
```

```
terraform > staging.tfvars > instance_type
1 instance_type = "t2.micro"

# module.vpc.aws_subnet.my-MySubnet2 will be destroyed
- resource "aws_subnet" "my-MySubnet2" {
  - arn = "arn:aws:ec2:us-east-1:983709588747:subnet/subnet-0b61b75bde45a85ec"
  - assign_ipv6_address_on_creation = false -> null
  - availability_zone = "us-east-1b" -> null
  - availability_zone_id = "use1-az4" -> null
  - cidr_block = "10.0.2.0/24" -> null
  - enable_dns4 = false -> null
  - enable_dns64 = false -> null
  - enable_ami_association_index = @ -> null
  - enable_resource_name_dns_a_record_on_launch = false -> null
  - enable_resource_name_dns_aaaa_record_on_launch = false -> null
  - id = "subnet-0b61b75bde45a85ec" -> null
  - ipv6_native = false -> null
  - map_customer_owned_ip_on_launch = false -> null
  - map_public_ip_on_launch = false -> null
  - owner_id = "983709588747" -> null
  - private_dns_hostname_type_on_launch = "ip-name" -> null
  - region = "us-east-1" -> null
  - tags = {
    "Name" = "MySubnet2"
  } -> null
  - tags_all = {
    "Name" = "MySubnet2"
  } -> null
  - vpc_id = "vpc-013f059ff95ce9a93" -> null
  # (4 unchanged attributes hidden)
}

# module.vpc.aws_subnet.my-subnet1 will be destroyed
- resource "aws_subnet" "my-subnet1" {
  - arn = "arn:aws:ec2:us-east-1:983709588747:subnet/subnet-08976faab826361"
  - assign_ipv6_address_on_creation = false -> null
  - availability_zone = "us-east-1b" -> null
}

Plan: 0 to add, 0 to change, 4 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo, only 'yes' will be accepted to confirm.

Enter a value: yes

module.vpc.aws_subnet.my-MySubnet2: Destroying... [id=subnet-0b61b75bde45a85ec]
module.vpc.aws_subnet.my-subnet1: Destroying... [id=subnet-08976faab826361]
module.ec2.aws_instance.demo: Destroying... [id=i-08ef7de22d99c6718]
module.vpc.aws_subnet.my-Subnet2: Destruction complete after 2s
module.vpc.aws_subnet.my-subnet1: Destruction complete after 2s
module.vpc.aws_vpc.my-Vpc: Destroying... [id=vpc-013f059ff95ce9a93]
module.vpc.aws_vpc.my-Vpc: Destruction complete after 2s
module.ec2.aws_instance.demo: Still destroying... [id=i-08ef7de22d99c6718, 00m10s elapsed]
module.ec2.aws_instance.demo: Still destroying... [id=i-08ef7de22d99c6718, 00m20s elapsed]
module.ec2.aws_instance.demo: Still destroying... [id=i-08ef7de22d99c6718, 00m30s elapsed]
module.ec2.aws_instance.demo: Destruction complete after 3zs

Destroy complete! Resources: 4 destroyed.

ls419@LAXMI MINGW64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$ terraform destroy -var-file="production.tfvars"

No changes. No objects need to be destroyed.

Either you have not created any objects yet or the existing objects were already deleted outside of Terraform.

Destroy complete! Resources: 0 destroyed.

ls419@LAXMI MINGW64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$
```

```
File Edit Selection View Go Run Terminal Help ← → Terraform_Tool
EXPLORER ... provider.tf main.tf terraform staging.tfvars production.tfvars variable.tf terraform ...
TERRAFORM_TOOL
aws README.md THIRD_PARTY_LICENSES
terraform
  terraform
    Modules
      Ec2
        terraform
          terraform.lock.hcl
        main.tf
        output.tf
        terraform.state
        terraform.state.backup
        variable.tf
      Vpc
        main.tf
        output.tf
        variable.tf
      .terraform.lock.hcl
        backend.tf
        main.tf
        output.tf
        production.tfvars
        provider.tf
        staging.tfvars
      terraform.state
      terraform.state.backup
      variable.tf
  OUTLINE
  TIMELINE
  APPLICATION_BUILDER
  0 0 AWS: profiledefault
  24°C Mostly cloudy
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - terraform + ... x
Welcome to Copilot
Let's get started
Add context (#), extensions (@), com ...
Build workspace
Show project config
Review AI output carefully before use.
In 1, Col 27 Spaces:4 UTF-8 CRLF {} terraform-vars Finish Setup Go Live Prettier
ENG IN 23:16 13-08-2025
```

```
terraform > staging.tfvars > instance_type
1 instance_type = "t2.micro"

# module.vpc.aws_subnet.my-MySubnet2 will be destroyed
- resource "aws_subnet" "my-MySubnet2" {
  - arn = "arn:aws:ec2:us-east-1:983709588747:subnet/subnet-0b61b75bde45a85ec"
  - assign_ipv6_address_on_creation = false -> null
  - availability_zone = "us-east-1b" -> null
  - availability_zone_id = "use1-az4" -> null
  - cidr_block = "10.0.2.0/24" -> null
  - enable_dns4 = false -> null
  - enable_dns64 = false -> null
  - enable_ami_association_index = @ -> null
  - enable_resource_name_dns_a_record_on_launch = false -> null
  - enable_resource_name_dns_aaaa_record_on_launch = false -> null
  - id = "subnet-0b61b75bde45a85ec" -> null
  - ipv6_native = false -> null
  - map_customer_owned_ip_on_launch = false -> null
  - map_public_ip_on_launch = false -> null
  - owner_id = "983709588747" -> null
  - private_dns_hostname_type_on_launch = "ip-name" -> null
  - region = "us-east-1" -> null
  - tags = {
    "Name" = "MySubnet2"
  } -> null
  - tags_all = {
    "Name" = "MySubnet2"
  } -> null
  - vpc_id = "vpc-013f059ff95ce9a93" -> null
  # (4 unchanged attributes hidden)
}

# module.vpc.aws_subnet.my-subnet1 will be destroyed
- resource "aws_subnet" "my-subnet1" {
  - arn = "arn:aws:ec2:us-east-1:983709588747:subnet/subnet-08976faab826361"
  - assign_ipv6_address_on_creation = false -> null
  - availability_zone = "us-east-1b" -> null
}

Plan: 0 to add, 0 to change, 4 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo, only 'yes' will be accepted to confirm.

Enter a value: yes

module.vpc.aws_subnet.my-MySubnet2: Destroying... [id=subnet-0b61b75bde45a85ec]
module.vpc.aws_subnet.my-subnet1: Destroying... [id=subnet-08976faab826361]
module.ec2.aws_instance.demo: Destroying... [id=i-08ef7de22d99c6718]
module.vpc.aws_subnet.my-Subnet2: Destruction complete after 2s
module.vpc.aws_subnet.my-subnet1: Destruction complete after 2s
module.vpc.aws_vpc.my-Vpc: Destroying... [id=vpc-013f059ff95ce9a93]
module.vpc.aws_vpc.my-Vpc: Destruction complete after 2s
module.ec2.aws_instance.demo: Still destroying... [id=i-08ef7de22d99c6718, 00m10s elapsed]
module.ec2.aws_instance.demo: Still destroying... [id=i-08ef7de22d99c6718, 00m20s elapsed]
module.ec2.aws_instance.demo: Still destroying... [id=i-08ef7de22d99c6718, 00m30s elapsed]
module.ec2.aws_instance.demo: Destruction complete after 3zs

Destroy complete! Resources: 4 destroyed.

ls419@LAXMI MINGW64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$ terraform destroy -var-file="production.tfvars"

No changes. No objects need to be destroyed.

Either you have not created any objects yet or the existing objects were already deleted outside of Terraform.

Destroy complete! Resources: 0 destroyed.

ls419@LAXMI MINGW64 /d/Laxmi Personal/Laxmi personal/Devops/Terraform_Tool/terraform
$
```