

1. Watch the Terraform-06 video.

2. Execute the Script Shown in the Video.

Create a directory named as aws_instance in your terraform folder in that directory I have 2 files one is main.tf and variables.tf

The screenshot shows the VS Code interface with the Explorer sidebar open, displaying a folder structure under 'TERRAFORM BASICS' with subfolders 'aws_instance', 'development', and files 'main.tf' and 'variables.tf'. The 'main.tf' file is selected and its content is shown in the right-hand editor pane:

```
aws_instance > main.tf > resource "aws_instance" "webserver"
1   resource "aws_instance" "webserver" {
2     ami = var.ami
3     instance_type = "t3.micro"
4     subnet_id = "subnet-0a192382de0e2bf6a"
5 }
```

The screenshot shows the VS Code interface with the Explorer sidebar open, displaying a folder structure under 'TERRAFORM BASICS' with subfolders 'aws_instance', 'development', and files 'main.tf' and 'variables.tf'. The 'variables.tf' file is selected and its content is shown in the right-hand editor pane:

```
aws_instance > variables.tf > variable "subnet_id"
1   variable "ami" {
2
3 }
4 variable "subnet_id" {
5
6 }
7
```

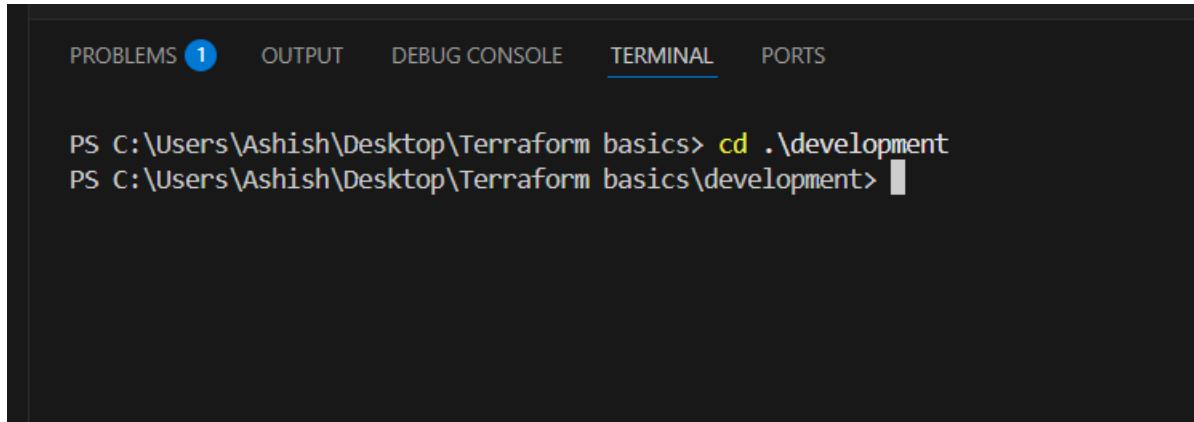
Create another folder named as development in the terraform folder in that I have a file main.tf

The screenshot shows the VS Code interface with the Explorer sidebar open, displaying a folder structure under 'TERRAFORM BASICS' with subfolders 'aws_instance' and 'development'. Within 'development', there are files '.terraform', '.terraform.lock.hcl', 'main.tf', 'terraform.tfstate', and 'terraform.tfstate.backup'. The 'main.tf' file in the 'development' folder is selected and its content is shown in the right-hand editor pane:

```
development > main.tf > module "dev-webserver"
1   module "dev-webserver" [
2     source = "../aws_instance"
3     ami = "ami-0cae6d6fe6048ca2c"
4   ]
5 }
```

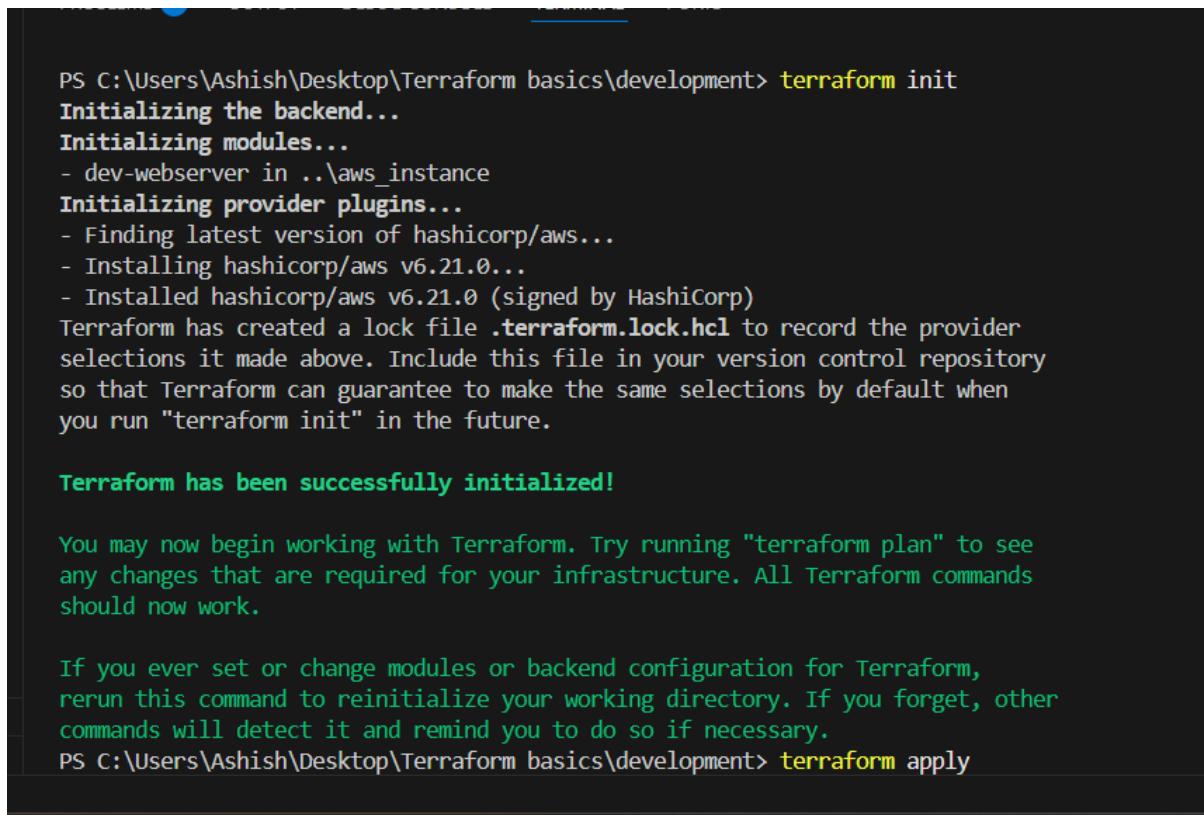
Then change your directory to development by using

```
cd .\development
```



PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ashish\Desktop\Terraform basics> cd .\development
PS C:\Users\Ashish\Desktop\Terraform basics\development>
```



```
PS C:\Users\Ashish\Desktop\Terraform basics\development> terraform init
Initializing the backend...
Initializing modules...
- dev-webserver in ..\aws_instance
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (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.
PS C:\Users\Ashish\Desktop\Terraform basics\development> terraform apply
```

```
rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.  
PS C:\Users\Ashish\Desktop\Terraform basics\development> terraform apply  
  
Terraform used the selected providers to generate the following execution plan. Resource actions  
+ create  
  
Terraform will perform the following actions:  
  
# module.dev-webserver.aws_instance.webserver will be created  
+ resource "aws_instance" "webserver" {  
    + ami = "ami-0cae6d6fe6048ca2c"  
    + 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)
```

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
  
+ network_interface (known after apply)  
+ primary_network_interface (known after apply)  
+ private_dns_name_options (known after apply)  
+ root_block_device (known after apply)  
}  
  
Plan: 1 to add, 0 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.dev-webserver.aws_instance.webserver: Creating...  
module.dev-webserver.aws_instance.webserver: Still creating... [00m10s elapsed]  
module.dev-webserver.aws_instance.webserver: Creation complete after 16s [id=i-037d744ac6494816d]  
  
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.  
PS C:\Users\Ashish\Desktop\Terraform basics\development>
```

An instance will be created.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with 'EC2' selected under 'Instances'. The main area displays 'Instances (1/1) Info' with a table showing one instance: i-037d744ac6494816d, which is 'Running', an 't3.micro' type, has '3/3 checks passed', and is in 'us-east-1a'. There are buttons for 'Connect', 'Actions', and 'Launch instances'.

If you want to create one more instance for production environment you create a directory in terraform folder.

A screenshot of a Windows File Explorer window. The left sidebar shows 'Desktop' selected. The main pane lists several folders: '.git', 'aws_instance', 'development', and 'production'. A red arrow points to the 'production' folder. The file list includes:

Name	Date modified	Type	Size
.git	14-11-2025 18:42	File folder	
aws_instance	17-11-2025 14:53	File folder	
development	17-11-2025 15:26	File folder	
production	17-11-2025 15:46	File folder	

In that create a file.

A screenshot of the VS Code interface. The left sidebar shows the 'EXPLORER' view with a tree structure for 'TERRAFORM BASICS' containing 'aws_instance', 'development', and 'production' folders. The 'PROBLEMS' tab at the bottom shows two errors: '+ metadata_options (known after apply)' and '+ network_interface (known after apply)'. The code editor shows the 'main.tf' file from the 'prod' folder, which contains:

```
module "prod-webserver" {
  source = "../aws_instance"
  ami    = "ami-0cae6d6fe6048ca2c"
  subnet_id = "subnet-0a192382de0e2bf6a"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ashish\Desktop\Terraform basics\production> terraform init
Initializing the backend...
Initializing modules...
- prod-webserver in ..\aws_instance
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (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.

```
PS C:\Users\Ashish\Desktop\Terraform basics\production> terraform apply
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ashish\Desktop\Terraform basics\production> terraform apply
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with
+ create

Terraform will perform the following actions:

# module.prod-webserver.aws_instance.webserver will be created
+ resource "aws_instance" "webserver" {
    + ami                                = "ami-0cae6d6fe6048ca2c"
    + 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)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
+ network_interface (known after apply)
+ primary_network_interface (known after apply)
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}
```

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

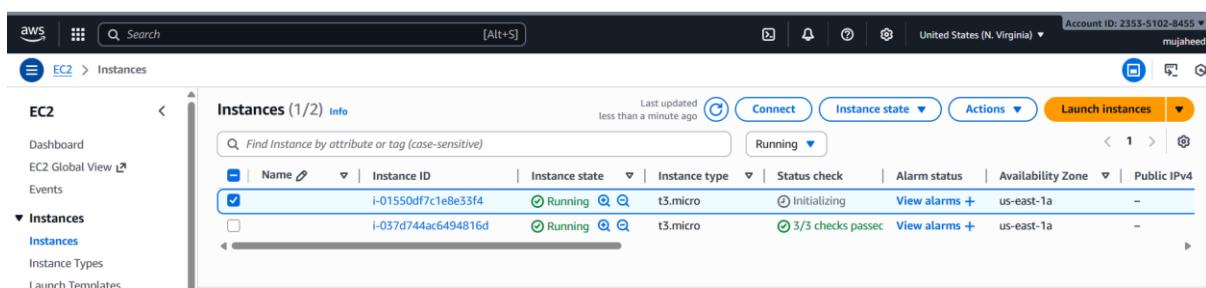
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

```
module.prod-webserver.aws_instance.webserver: Creating...
module.prod-webserver.aws_instance.webserver: Still creating... [00m10s elapsed]
module.prod-webserver.aws_instance.webserver: Creation complete after 17s [id=i-01550df7c1e8e33f4]
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics\production>

Here an instance has been created for production purpose.



The screenshot shows the AWS EC2 Instances page. The left sidebar has 'Instances' expanded, showing 'Instances' and 'Launch Templates'. The main area displays 'Instances (1/2) Info' with a table of running instances. The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. Two instances are listed: one with ID i-01550df7c1e8e33f4 and another with ID i-037d744ac6494816d, both running t3.micro instances in us-east-1a.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
	i-01550df7c1e8e33f4	Running	t3.micro	Initializing	View alarms +	us-east-1a	-
	i-037d744ac6494816d	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	-

Terraform workspace:

- terraform workspace new projectA

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform workspace new projectA
created and switched to workspace "projectA"!
```

```
You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> 
```

- **terraform workspace list**

```
for this configuration.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform workspace list
  default
* projectA
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> 
```

To select different workspace

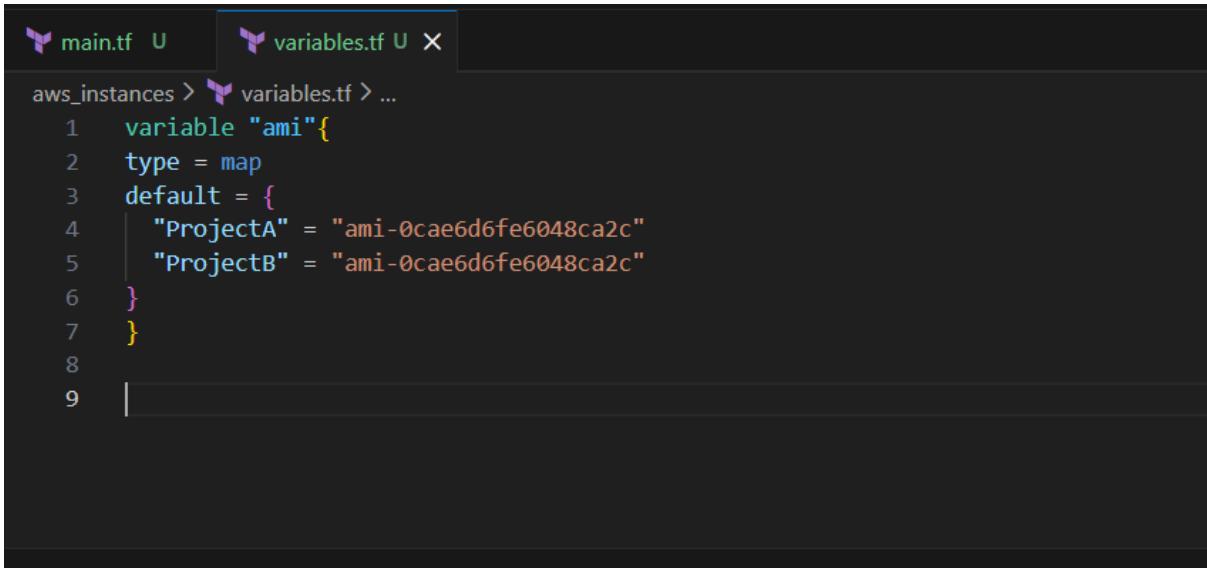
- **terraform workspace select default**

```
projectA
PS C:\Users\Ashish\Desktop\Terraform basics> terraform workspace select default
Switched to workspace "default".
PS C:\Users\Ashish\Desktop\Terraform basics> terraform workspace list
* default
  projectA
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> 
```



```
aws_instances > main.tf > resource "aws_instance" "webserver" > ami
1   resource "aws_instance" "webserver"{
2     ami = "var.ami"
3     instance_type = "t3.micro"
4     subnet_id = "subnet-0a192382de0e2bf6a"
5   }
6
```



```
aws_instances > variables.tf > ...
1   variable "ami"{
2     type = map
3     default = {
4       "ProjectA" = "ami-0cae6d6fe6048ca2c"
5       "ProjectB" = "ami-0cae6d6fe6048ca2c"
6     }
7   }
8
9 |
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> cd .\aws_instances
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instances> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (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.

```
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instances> terraform apply
```

commands will detect it and remind you to do so if necessary.

```
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instances> terraform apply
```

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

Terraform will perform the following actions:

```
# aws_instance.webserver will be created
+ resource "aws_instance" "webserver" {
    + 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
        = "var.ami"
        = (known after apply)
        = false
        = false
        = (known after apply)
        = (known after apply)
        = (known after apply)
```

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER**: Shows the file tree with files: `aws_instance`, `.terraform`, `terraform.tfstate.d`, `.terraform.lock.hcl`, `main.tf` (selected), and `variables.tf`.
- PROBLEMS**: Shows 1 error: `+ private_dns_name_options (known after apply)` and `+ root_block_device (known after apply)`.
- OUTPUT**: Shows the Terraform plan output:

```
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}
```

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

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

Enter a value: yes

aws_instance.webserver: Creating...
aws_instance.webserver: Still creating... [00m10s elapsed]
aws_instance.webserver: Creation complete after 18s [00m18s elapsed]

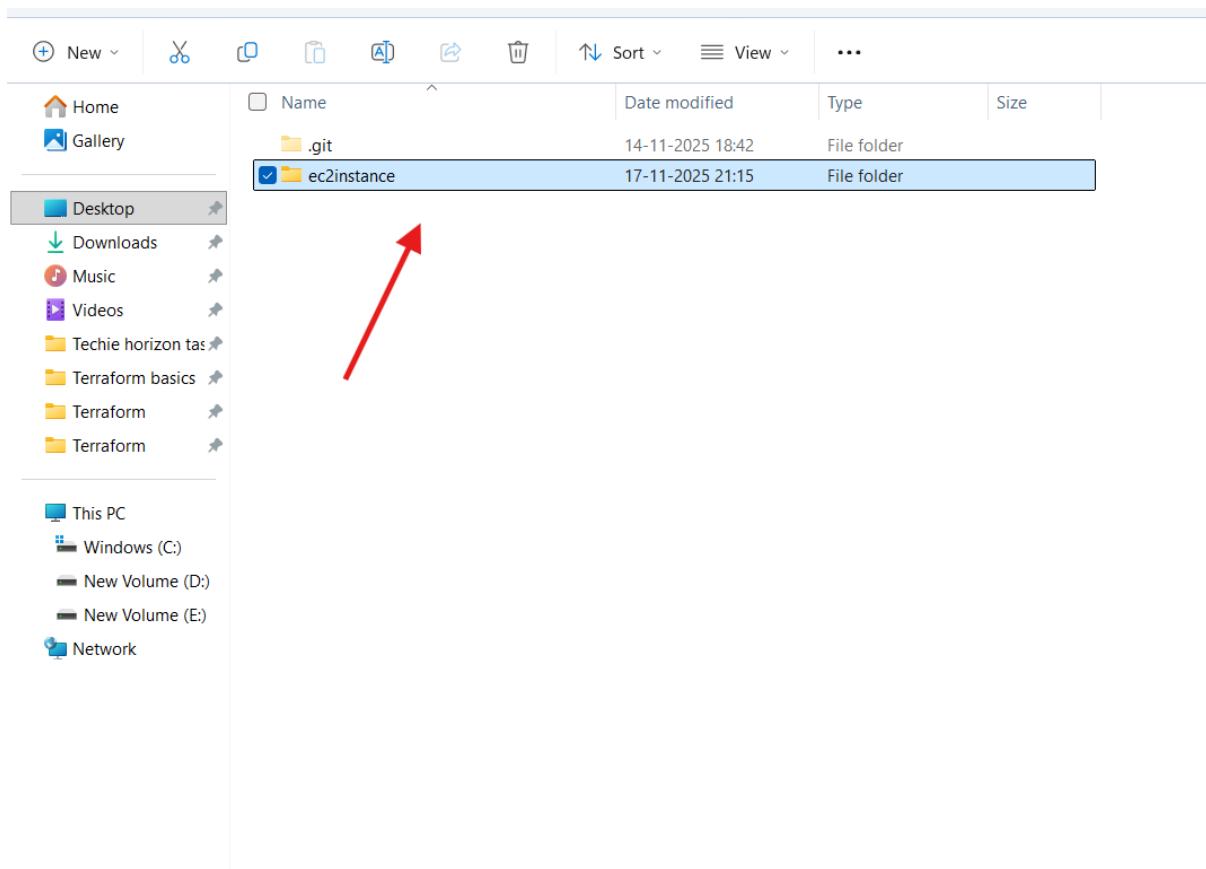
Apply complete! Resources: 1 added, 0 changed, 0 destroyed

The screenshot shows the AWS EC2 Instances page. The left sidebar has 'EC2' selected under 'Instances'. The main area shows 'Instances (1/1) Info' with a table. The table has columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. One row is listed: 'i-01a7d79cfb2aed8a9' with 'Running' status, 't3.micro' type, 'Initializing' check, and 'us-east-1a' zone. A search bar at the top says 'Find Instance by attribute or tag (case-sensitive)'.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
	i-01a7d79cfb2aed8a9	Running	t3.micro	Initializing	View alarms +	us-east-1a	-

3 .Provision EC2, S3, and VPC using Terraform modules.

Create a directory named as ec2instance.



In that directory I have main.tf, variable.tf

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left has sections for 'TERRAFORM BASICS' and 'ec2instance', with 'main.tf' and 'variables.tf' listed under 'ec2instance'. The Editor pane on the right displays the 'main.tf' file content:

```
resource "aws_instance" "webserver" {
  ami = var.ami
  instance_type = "t3.micro"
  subnet_id = "subnet-0a192382de0e2bf6a"
  tags = {
    Name = "webserver"
}
```

Below the Editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS.

```
main.tf U variables.tf U X
ec2instance > variables.tf > variable "subnet_id"
1   variable "ami" {
2
3   }
4   variable "subnet_id" {
5
6 }
```

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

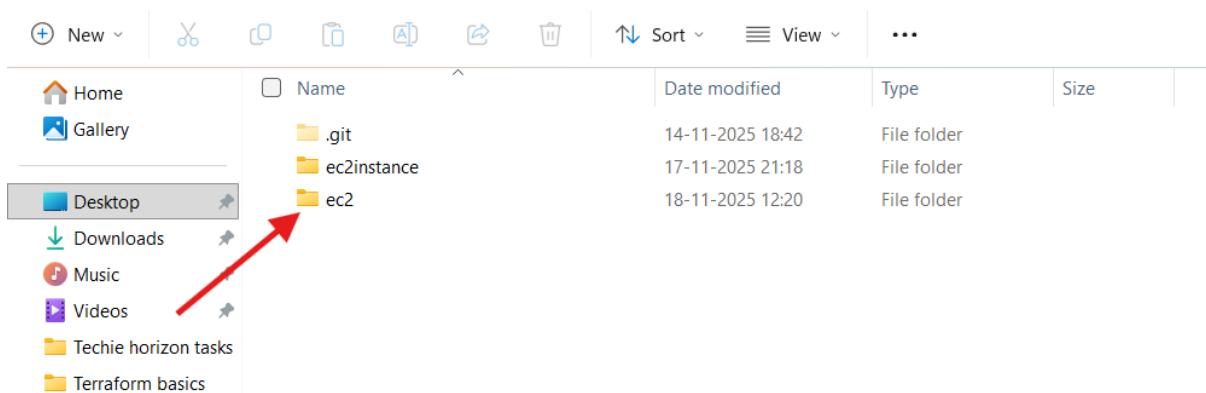
PS C:\Users\Ashish\Desktop\Terraform basics\ec2> cd ..
PS C:\Users\Ashish\Desktop\Terraform basics> cd ./aws_instance
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (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,

Create a directory in your terraform folder named as ec2.



Create a file in ec2 as main.

The screenshot shows the VS Code interface. On the left is the Explorer sidebar with a tree view of files. In the center is the main editor area with three tabs: 'main.tf aws_instance U', 'main.tf ec2 U X', and 'variables.tf U'. The 'main.tf ec2' tab is active, displaying the following Terraform code:

```
ec2 > main.tf > ...
1 module "ec2-webserver" {
2   source = "../aws_instance"
3   ami   = "ami-0cae6d6fe6048ca2c"
4   subnet_id = "subnet-0a192382de0e2bf6a"
5 }
6
```

The screenshot shows the terminal tab in VS Code. The output of the 'terraform init' command is displayed:

```
PS C:\Users\Ashish\Desktop\Terraform basics> cd ./ec2
PS C:\Users\Ashish\Desktop\Terraform basics\ec2> terraform init
Initializing the backend...
Initializing modules...
- ec2-webserver in ..\aws_instance
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (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.

The screenshot shows the terminal tab in VS Code. The output of the 'terraform apply' command is displayed:

```
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.
PS C:\Users\Ashish\Desktop\Terraform basics\ec2> terraform apply

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

Terraform will perform the following actions:

# module.ec2-webserver.aws_instance.webserver will be created
+ resource "aws_instance" "webserver" {
    + ami           = "ami-0cae6d6fe6048ca2c"
    + 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)
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}

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

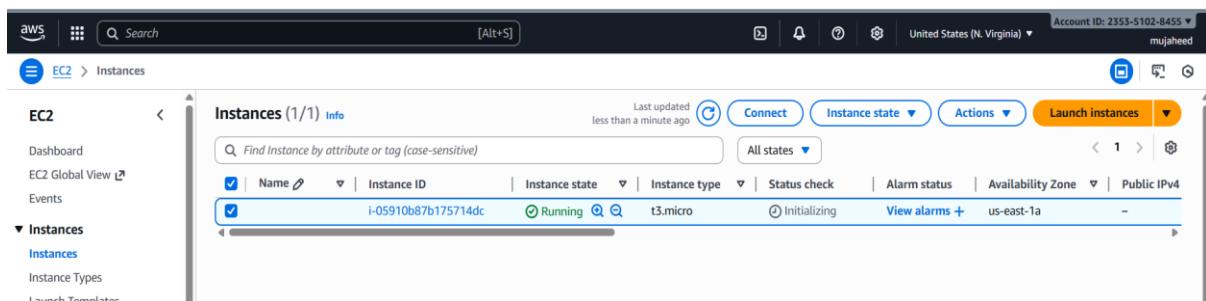
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

module.ec2-webserver.aws_instance.webserver: Creating...
module.ec2-webserver.aws_instance.webserver: Still creating... [00m10s elapsed]
module.ec2-webserver.aws_instance.webserver: Creation complete after 17s [id=i-05910b87b175714dc]

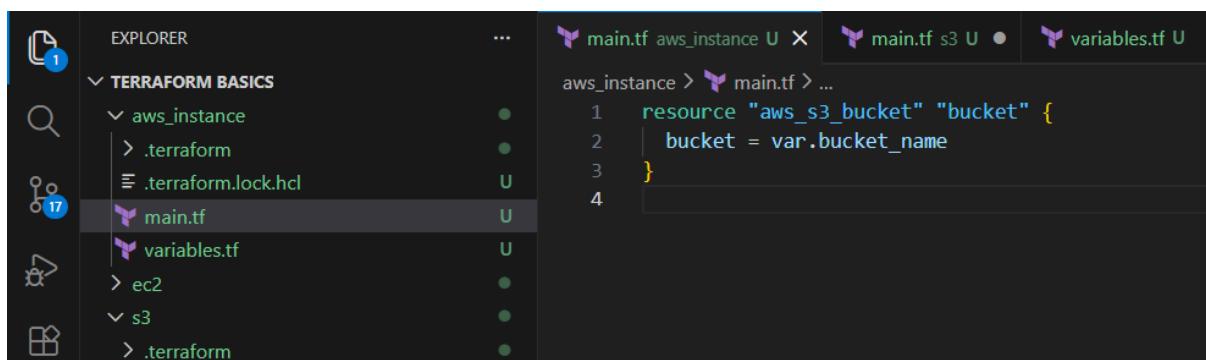
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics\ec2>
```

An ec2 instance has been created using terraform module.



S3:

In the aws_instance folder change the main.tf,variable.tf files.



The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure under "TERRAFORM BASICS". Files include `aws_instance`, `.terraform`, `.terraform.lock.hcl`, `main.tf`, `variables.tf`, `ec2`, `s3`, `.terraform`, `.terraform.lock.hcl`, `main.tf`, and `terraform.tfstate`.
- EDITOR**: The `variables.tf` file is open, showing a variable definition:


```
variable "bucket_name"
```

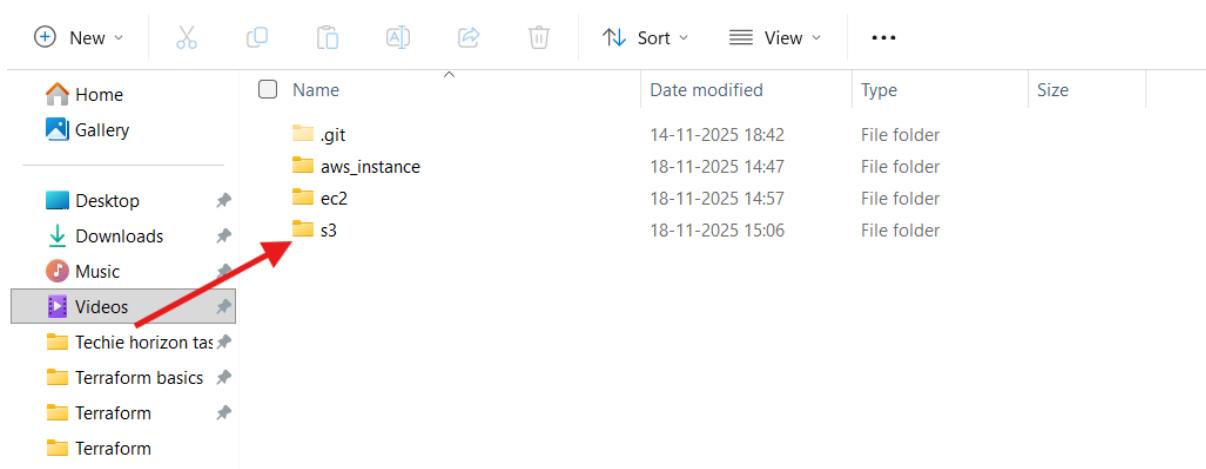
 Line 1: `1 variable "bucket_name"`
 Line 2: `2 }`
 Line 3: `3 }`
 Line 4: `4`
- TERMINAL**: Displays the terminal output of the `terraform init` command:


```
PS C:\Users\Ashish\Desktop\Terraform basics\ec2> cd ..  
PS C:\Users\Ashish\Desktop\Terraform basics> cd ./aws_instance  
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform init  
Initializing the backend...  
Initializing provider plugins...  
- Reusing previous version of hashicorp/aws from the dependency lock file  
- Using previously-installed hashicorp/aws v6.21.0  
  
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.

Create a directory in the terraform folder named as s3.



Create a file in the s3 folder as main.tf

EXPLORER

TERRAFORM BASICS

- aws_instance
- .terraform
- .terraform.lock.hcl
- main.tf
- variables.tf
- ec2
- s3
- .terraform
- .terraform.lock.hcl
- main.tf
- {} terraform.tfstate

... main.tf aws_instance U main.tf s3 U ● variables.tf U

```
s3 > main.tf > module "bucket" > bucket_name
1   module "bucket" {
2     source = "../aws_instance"
3     bucket_name = "terraformmodulebucket123"
4 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

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.

```
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> cd ..
PS C:\Users\Ashish\Desktop\Terraform basics> cd ./s3
PS C:\Users\Ashish\Desktop\Terraform basics\s3> terraform init
Initializing the backend...
Initializing modules...
- bucket in ..\aws_instance
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (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!

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

commands will detect it and remind you to do so if necessary.

```
PS C:\Users\Ashish\Desktop\Terraform basics\s3> terraform apply
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with colors.

+ create

Terraform will perform the following actions:

```
# module.bucket.aws_s3_bucket.bucket will be created
+ resource "aws_s3_bucket" "bucket" {
    + acceleration_status      = (known after apply)
    + acl                      = (known after apply)
    + arn                      = (known after apply)
    + bucket                   = "terraformmodulebucket123"
    + bucket_domain_name       = (known after apply)
    + bucket_prefix             = (known after apply)
    + bucket_region              = (known after apply)
    + bucketRegionalDomainName = (known after apply)
    + force_destroy            = false
    + hosted_zone_id           = (known after apply)
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
+ server_side_encryption_configuration (known after apply)
+ versioning (known after apply)
+ website (known after apply)
}
```

Plan: 1 to add, 0 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.bucket.aws_s3_bucket.bucket: Creating...
module.bucket.aws_s3_bucket.bucket: Creation complete after 6s [id=terraformmodulebucket123]
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics\s3>

An s3 bucket has been created by using terraform modules.

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with navigation links like 'General purpose buckets', 'Directory buckets', etc. The main area is titled 'General purpose buckets' and shows a table with one item:

Name	AWS Region	Creation date
terraformmodulebucket123	US East (N. Virginia) us-east-1	November 18, 2025, 15:20:35 (UTC+05:30)

On the right, there are two informational boxes: 'Account snapshot' and 'External access summary - new'.

Vpc:

Change the data In main.tf,variables.tf in aws_instance directory.

```
aws_instance > main.tf > resource "aws_vpc" "vpcmain"
1   resource "aws_vpc" "vpcmain" {
2     cidr_block = var.cidr_block
3
4     tags = {
5       Name = var.vpc_name
6     }
7   }
```

```
aws_instance > variables.tf > variable "vpc_name"
1   variable "cidr_block" {
2
3   }
4
5   variable "vpc_name" {
6
7   }
```

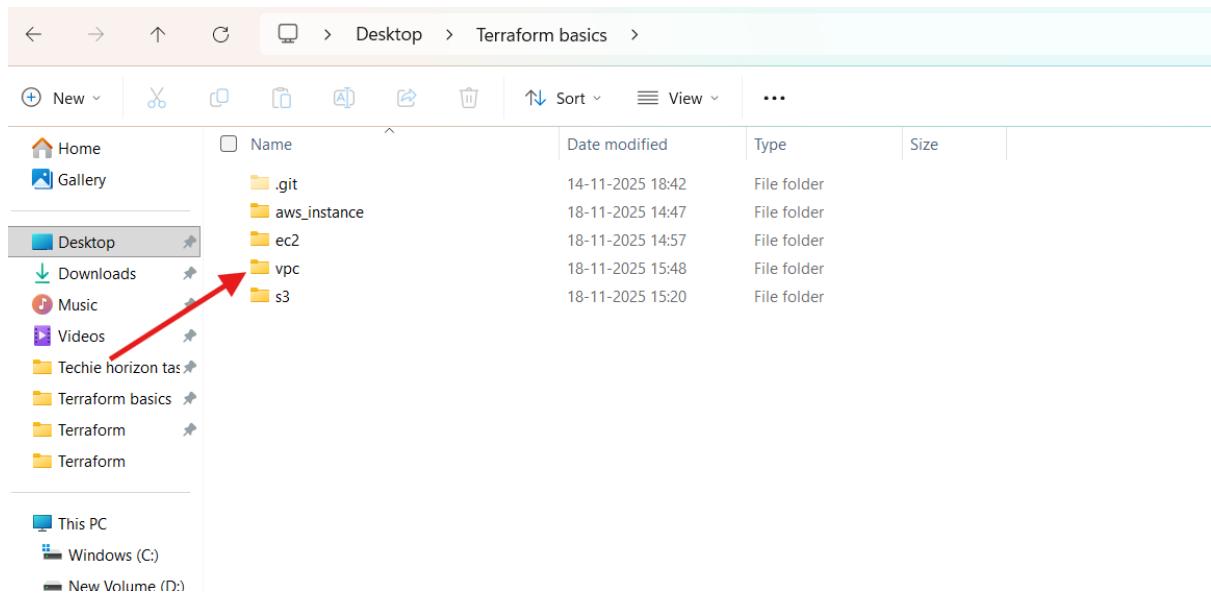
```
PS C:\Users\Ashish\Desktop\Terraform basics> cd ./aws_instance
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

Terraform has been successfully initialized!

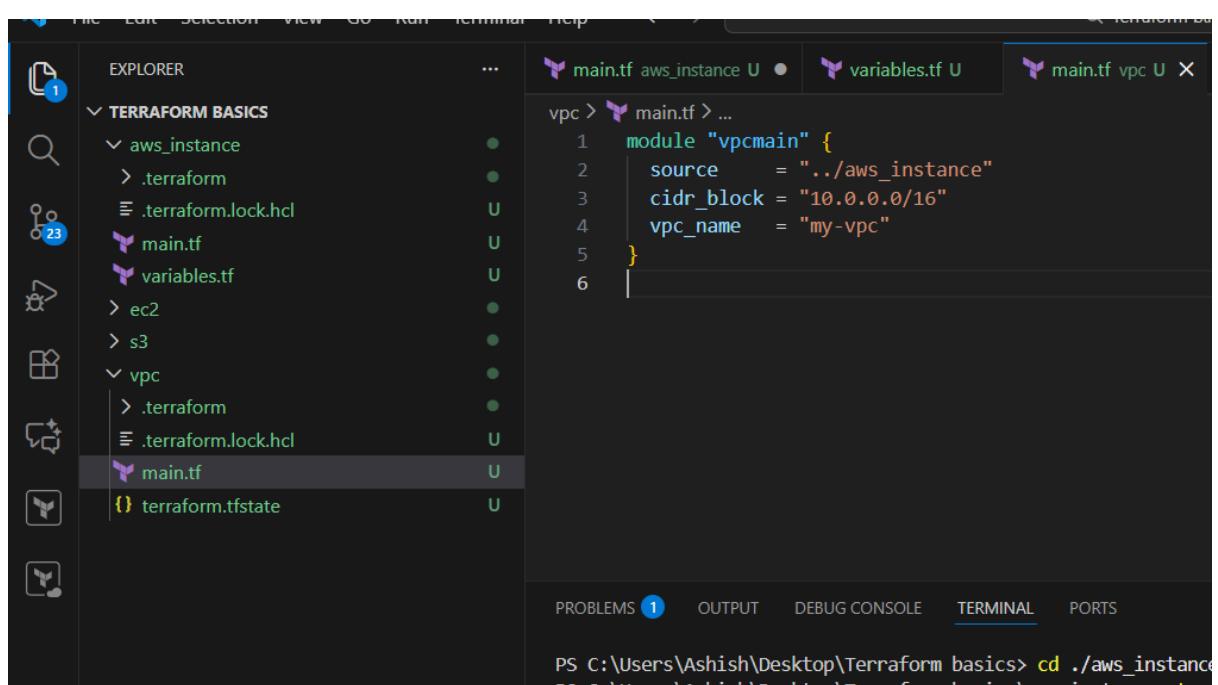
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

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.
```

Create a directory named as vpc in the terraform folder.



Create a file main.tf in vpc.



```
PS C:\Users\Ashish\Desktop\Terraform basics\vpc> terraform init
Initializing the backend...
Initializing modules...
- vpcmain in ..\aws_instance
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (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,

Run this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

```
PS C:\Users\Ashish\Desktop\Terraform basics\vpc> terraform apply
```

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

Terraform will perform the following actions:

```
# module.vpcmain.aws_vpc.vpcmain will be created
+ resource "aws_vpc" "vpcmain" {
    + arn                      = (known after apply)
    + cidr_block               = "10.0.0.0/16"
    + default_network_acl_id   = (known after apply)
    + default_route_table_id   = (known after apply)
    + default_security_group_id = (known after apply)
    + dhcp_options_id          = (known after apply)
    + enable_dns_hostnames     = (known after apply)
    + enable_dns_support        = true
    + enable_network_address_usage_metrics = (known after apply)
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

+ "Name" = "my-vpc"
}
+ tags_all
+ "Name" = "my-vpc"
}
}

Plan: 1 to add, 0 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.vpcmain.aws_vpc.vpcmain: Creating...
module.vpcmain.aws_vpc.vpcmain: Creation complete after 5s [id=vpc-02a9dda495d24bd85]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics\vpc>

```

A vpc has been created by using terraform modules.

Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR
default	vpc-06cf45eaab13624fe	Available	Off	172.31.0.0/16	-
my-vpc	vpc-02a9dda495d24bd85	Available	Off	10.0.0.0/16	-

4. Provision EC2 for 3 different environments (Dev, Staging, and Prod) using Terraform workspaces.

Create 3 workspace

- terraform workspace new dev
- terraform workspace new staging
- terraform workspace new prod

```
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform workspace new dev  
Created and switched to workspace "dev"!
```

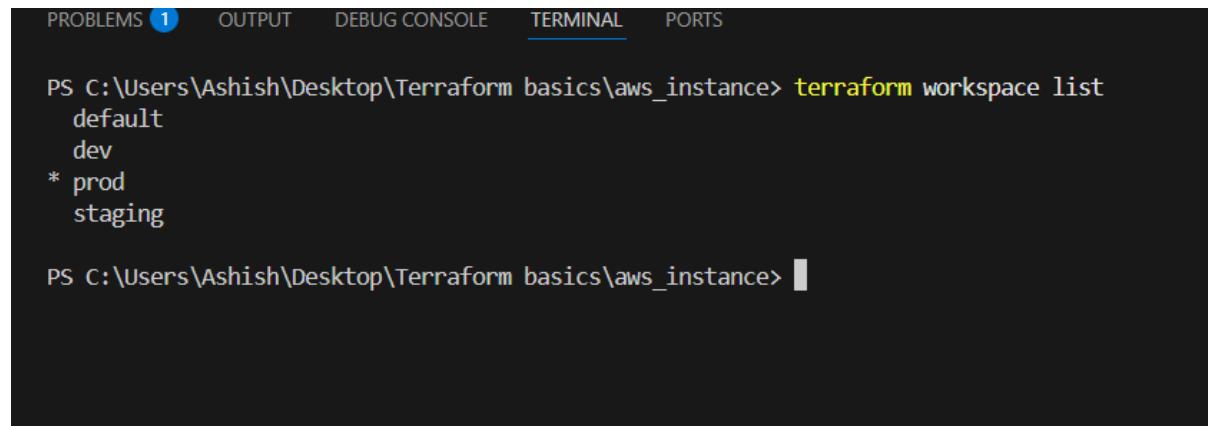
You're now on a new, empty workspace. Workspaces isolate their state, so if you run "terraform plan" Terraform will not see any existing state for this configuration.

```
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform workspace new staging  
Created and switched to workspace "staging"!
```

You're now on a new, empty workspace. Workspaces isolate their state, so if you run "terraform plan" Terraform will not see any existing state for this configuration.

```
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform workspace new prod  
Created and switched to workspace "prod"!
```

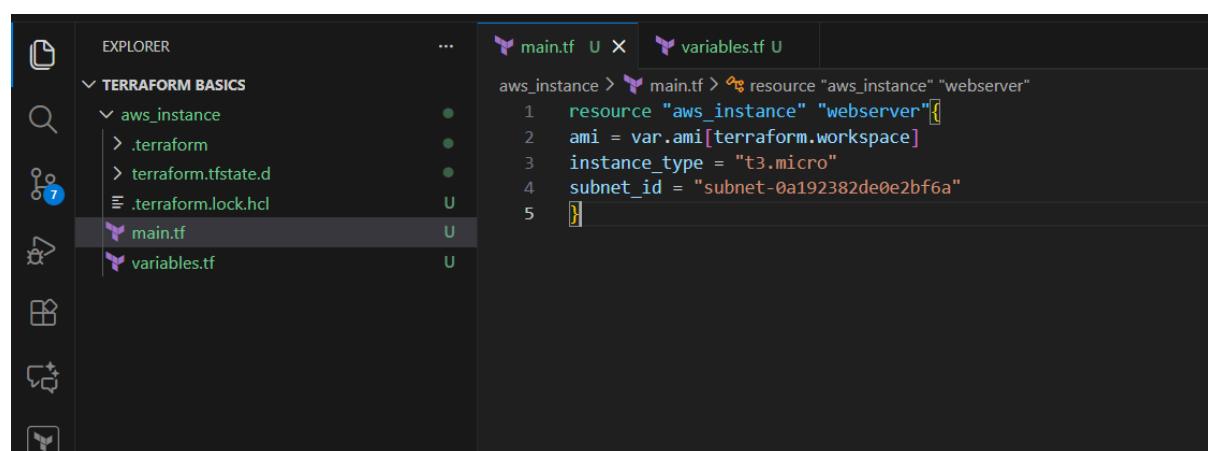
You're now on a new, empty workspace. Workspaces isolate their state, so if you run "terraform plan" Terraform will not see any existing state for this configuration.



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform workspace list
  default
  dev
* prod
  staging
```

Give main.tf, variables.tf as



```
EXPLORER ... main.tf U X variables.tf U
TERRAFORM BASICS aws_instance > main.tf > resource "aws_instance" "webserver" [
  .terraform
  .terraform.lock.hcl
  main.tf
  variables.tf
]

aws_instance > main.tf > resource "aws_instance" "webserver" [
  ami = var.ami[terraform.workspace]
  instance_type = "t3.micro"
  subnet_id = "subnet-0a192382de0e2bf6a"
]
```

```
aws_instance > variables.tf > ...
1 variable "ami" {
2   type = map(string)
3   default = {
4     dev = "ami-0cae6d6fe6048ca2c"
5     staging = "ami-0ecb62995f68bb549"
6     prod = "ami-0b4bc1e90f30ca1ec"
7   }
8 }
9
```

Select dev as workspace.

- terraform workspace select dev

```
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource a
+ create

Terraform will perform the following actions:

# aws_instance.webserver will be created
+ resource "aws_instance" "webserver" {
    + ami
    + arn
    + associate_public_ip_address
    + availability_zone
    + disable_api_stop
    + disable_api_termination
    + ebs_optimized
    + enable_primary_ipv6
    + force_destroy
    + private_dns_name_options (known after apply)
    + root_block_device (known after apply)
}
```

```
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}

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

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

Enter a value: yes

aws_instance.webserver: Creating...
aws_instance.webserver: Still creating... [00m10s elapsed]
aws_instance.webserver: Creation complete after 18s [id=i-01a7d79cfb2aed8a9]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance>
```

An instance has been created with dev workspace.

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table titled 'Instances (1/1) Info' with one row. The row contains the following information: Instance ID (i-01a7d79cfb2aed8a9), Instance state (Running), Instance type (t3.micro), Status check (Initializing), Alarm status (View alarms +), Availability Zone (us-east-1a), and Public IPv4 (not visible). There are filters for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. Buttons for Connect, Instance state, Actions, and Launch instances are at the top right. A search bar and account information (Account ID: 2353-5102-8455, mujahed) are at the top.

Shift to staging branch

- terraform workspace select staging
- terraform apply

```
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform workspace list
default
dev
prod
* staging

PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform apply

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

Terraform will perform the following actions:

# aws_instance.webserver will be created
+ resource "aws_instance" "webserver" {
    + ami
    + arn
    + associate_public_ip_address
    + availability_zone
    + disable_api_stop
        = "ami-0ecb62995f68bb549"
        = (known after apply)
        = (known after apply)
        = (known after apply)
        = (known after apply)
```

```
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}

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

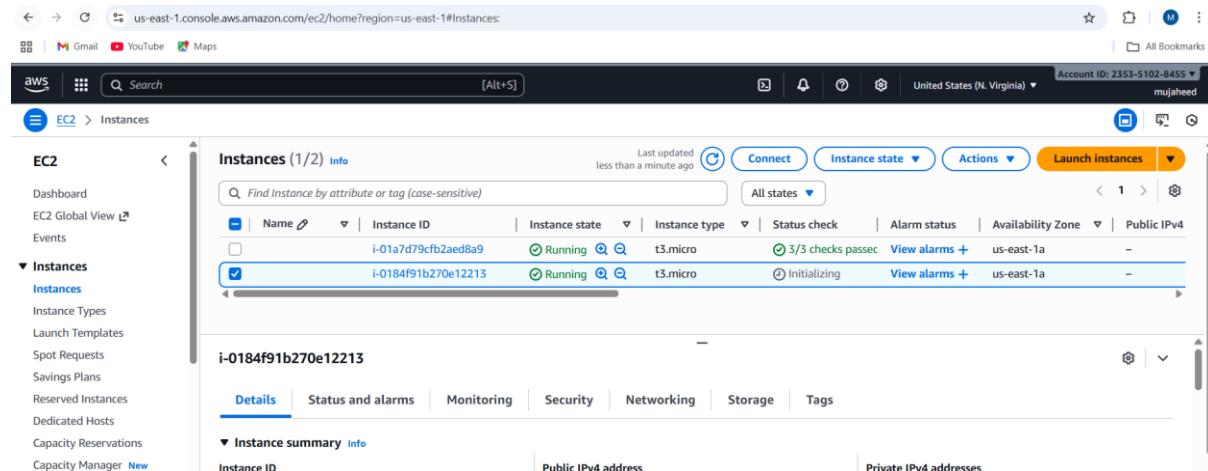
Do you want to perform these actions in workspace "staging"?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.webserver: Creating...
aws_instance.webserver: Still creating... [00m10s elapsed]
aws_instance.webserver: Creation complete after 18s [id=i-0184f91b270e12213]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance>
```

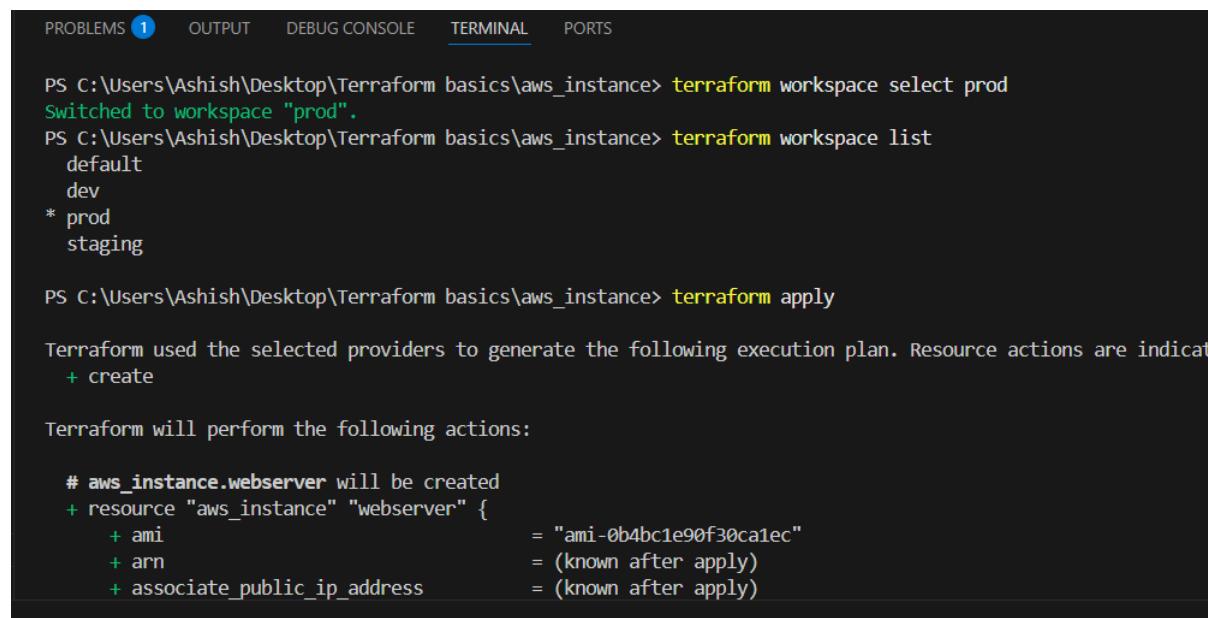
An instance has been created for staging workspace.



The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table of instances. There are two rows: one for an instance with ID i-01a7d79cfb2aed8a9 (status: Running, type: t3.micro) and another for an instance with ID i-0184f91b270e12213 (status: Running, type: t3.micro). The second instance is selected. Below the table, there is a detailed view for the selected instance, showing tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The Details tab is active, showing the Instance ID i-0184f91b270e12213, Public IPv4 address, and Private IPv4 addresses.

Shift to production workspace

- terraform workspace select staging
- terraform apply



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform workspace select prod
Switched to workspace "prod".
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform workspace list
  default
  dev
* prod
  staging

PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated by colors.
+ create

Terraform will perform the following actions:

# aws_instance.webserver will be created
+ resource "aws_instance" "webserver" {
    + ami                               = "ami-0b4bc1e90f30ca1ec"
    + arn                               = (known after apply)
    + associate_public_ip_address       = (known after apply)
```

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

+ private_dns_name_options (known after apply)

+ root_block_device (known after apply)
}

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

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

Enter a value: yes

aws_instance.webserver: Creating...
aws_instance.webserver: Still creating... [00m10s elapsed]
aws_instance.webserver: Creation complete after 17s [id=i-062b878c85aa82e64]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics\aws_instance>
```

An Instance has been created with prod workspace.

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table titled 'Instances (1/3) Info'. The table has columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. There are three rows:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
i-01a7d79cfb2aed8a9	i-01a7d79cfb2aed8a9	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	-
i-018af91b270e12213	i-018af91b270e12213	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	-
i-062b878c85aa82e64	i-062b878c85aa82e64	Running	t3.micro	Initializing	View alarms +	us-east-1a	-

Below the table, the instance ID 'i-062b878c85aa82e64' is highlighted. At the bottom of the page, there are tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags.