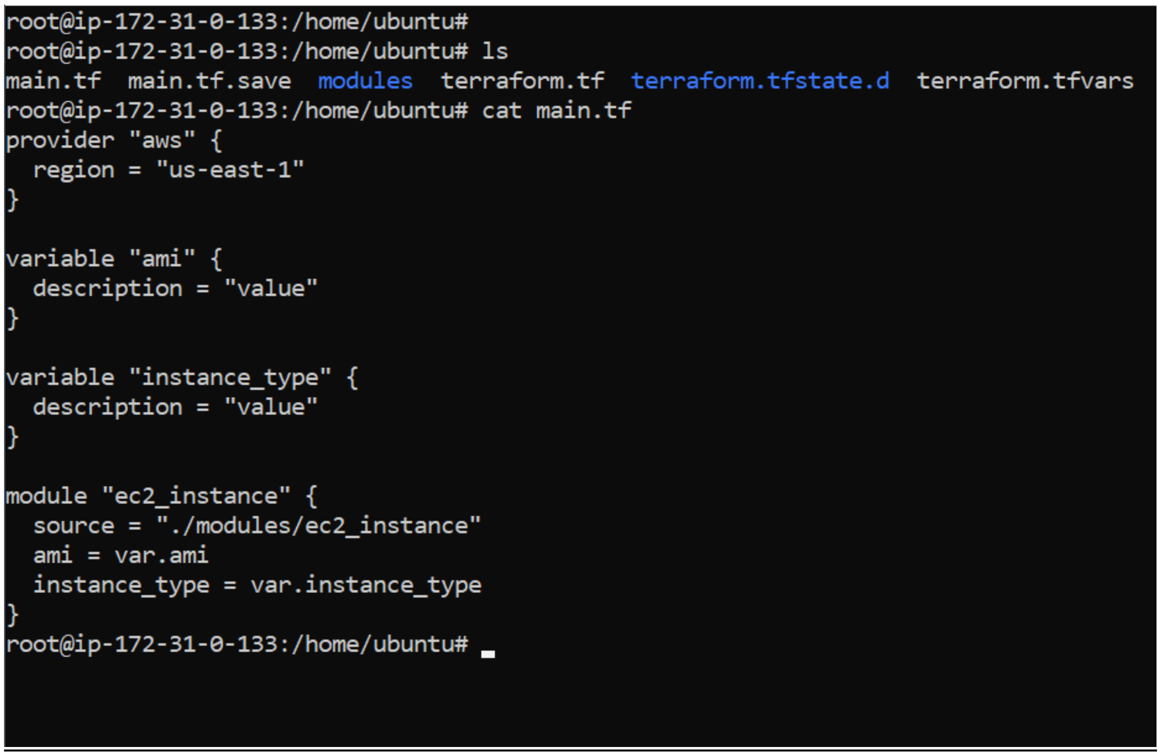
#terraform workspace

step 1: firstly we are going to create three files and two folders as following. the primary files in the root dir will be '**main.tf'** and '**terraform.tfvars**' then will create folder called '**modules'** in that particular folder will create another one folder by '**ec2_instance'** in that will create the same file like '**main.tf'** but the configurable data will be different as follows.

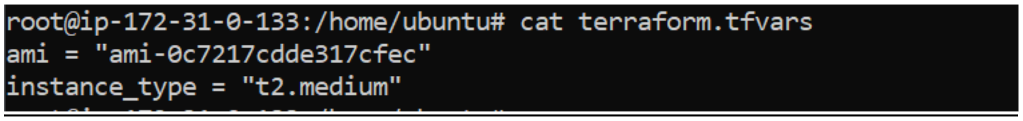
> primary files: main.tf, terraform.tfvars

> secondary: modules folder, inside that another one folder ec2_instance/ main.tf



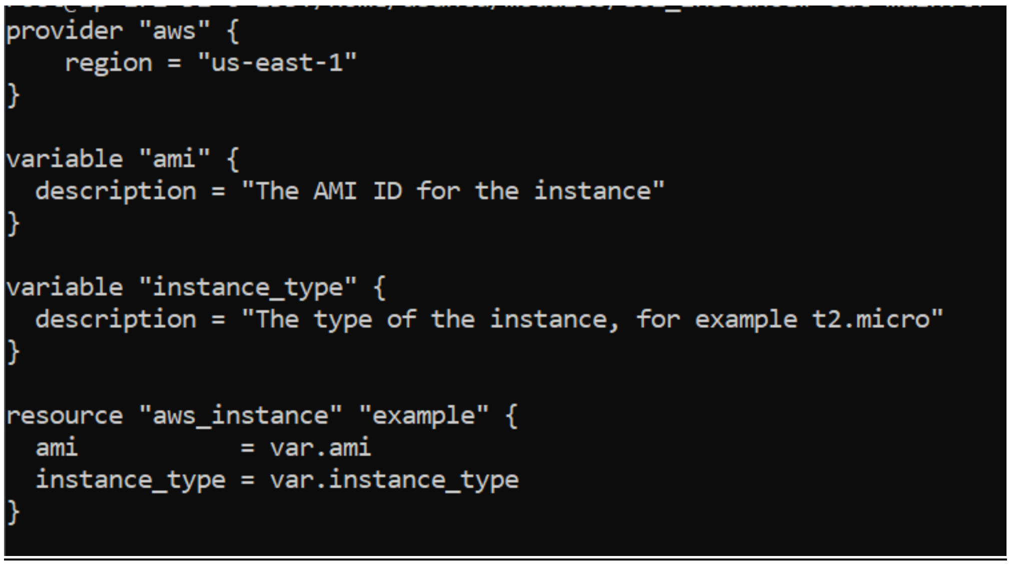
```
root@ip-172-31-0-133:/home/ubuntu#
root@ip-172-31-0-133:/home/ubuntu# ls
main.tf  main.tf.save  modules  terraform.tf  terraform.tfstate.d  terraform.tfvars
root@ip-172-31-0-133:/home/ubuntu# cat main.tf
provider "aws" {
  region = "us-east-1"
}

variable "ami" {
  description = "value"
}

variable "instance_type" {
  description = "value"
}

module "ec2_instance" {
  source = "./modules/ec2_instance"
  ami = var.ami
  instance_type = var.instance_type
}
root@ip-172-31-0-133:/home/ubuntu#
```

main.tf



```
root@ip-172-31-0-133:/home/ubuntu# cat terraform.tfvars
ami = "ami-0c7217cdde317cfec"
instance_type = "t2.medium"
```

terraform.tfvars


```
provider "aws" {
    region = "us-east-1"
}

variable "ami" {
  description = "The AMI ID for the instance"
}

variable "instance_type" {
  description = "The type of the instance, for example t2.micro"
}

resource "aws_instance" "example" {
  ami           = var.ami
  instance_type = var.instance_type
}
```

cd modules/ec2_instance/main.tf <

<u>#commands for workspace</u>

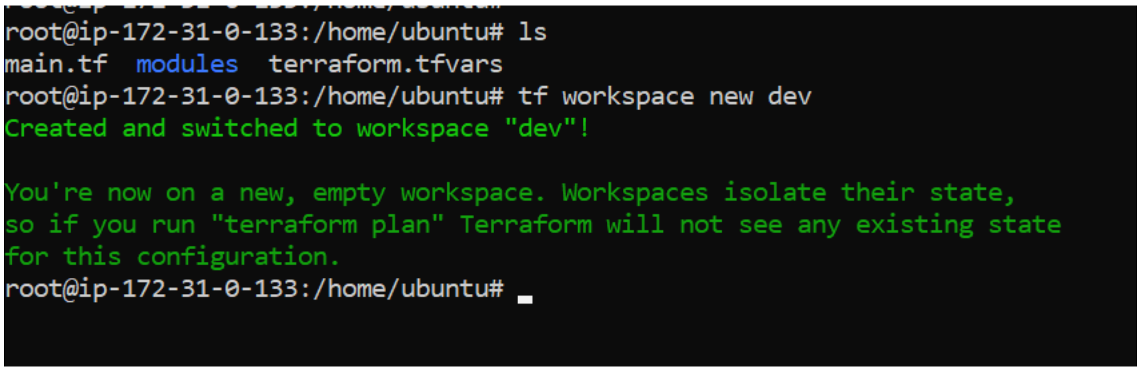
there we are creating **dev, stage, prod** environments with the help of below commands. the purpose is separate '**terraform.tfstate**' file will be created in single infra of **dev**, **stage** & **prod** so the '**state**' file will keep track of changes, modify, destroy records for the specific environment.

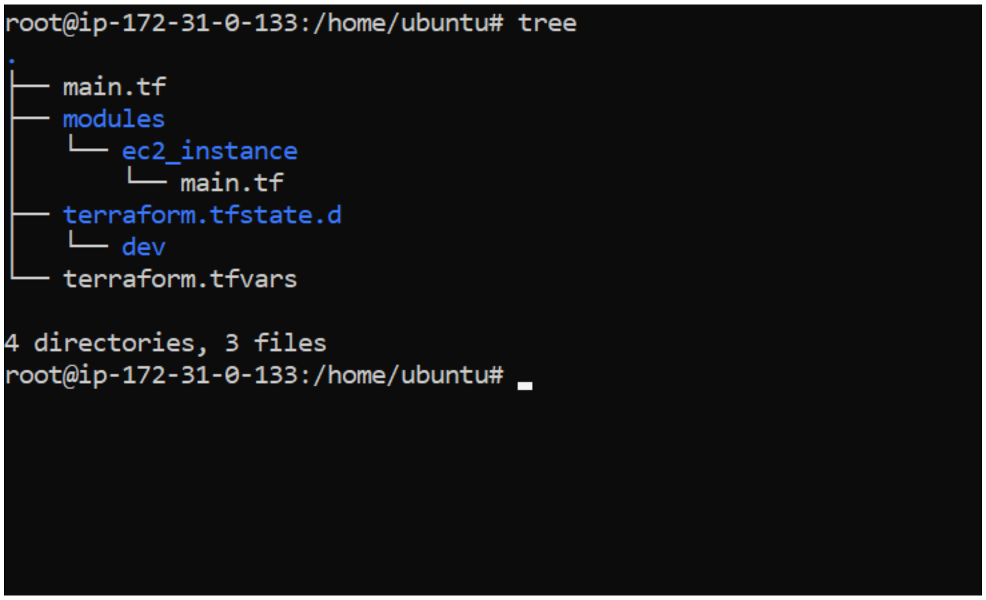
>**terraform workspace new dev**

>terraform workspace new stage

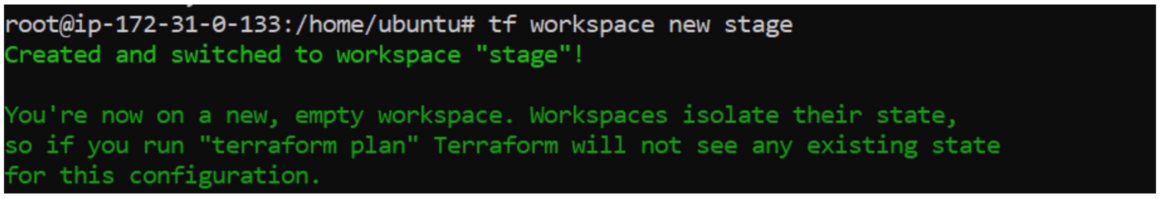
>terraform workspace new prod

[will create workspaces as follows with the help of these commands]

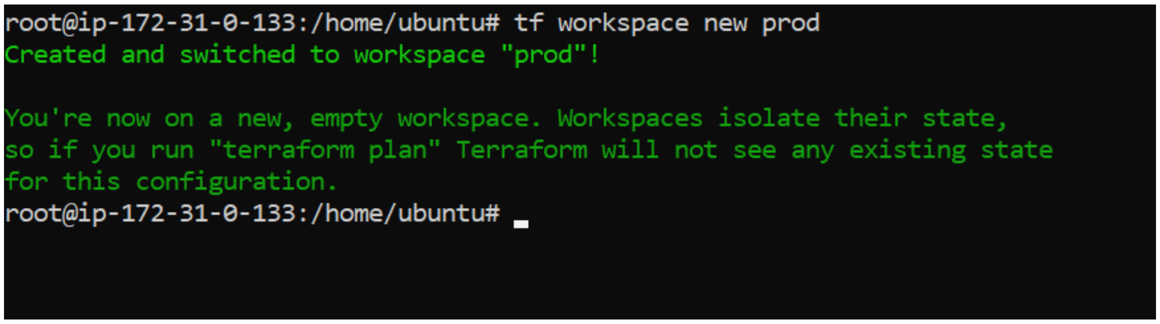

```
root@ip-172-31-0-133:/home/ubuntu# ls
main.tf  modules  terraform.tfvars
root@ip-172-31-0-133:/home/ubuntu# tf workspace new dev
Created and switched to workspace "dev"!

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.
root@ip-172-31-0-133:/home/ubuntu#
```



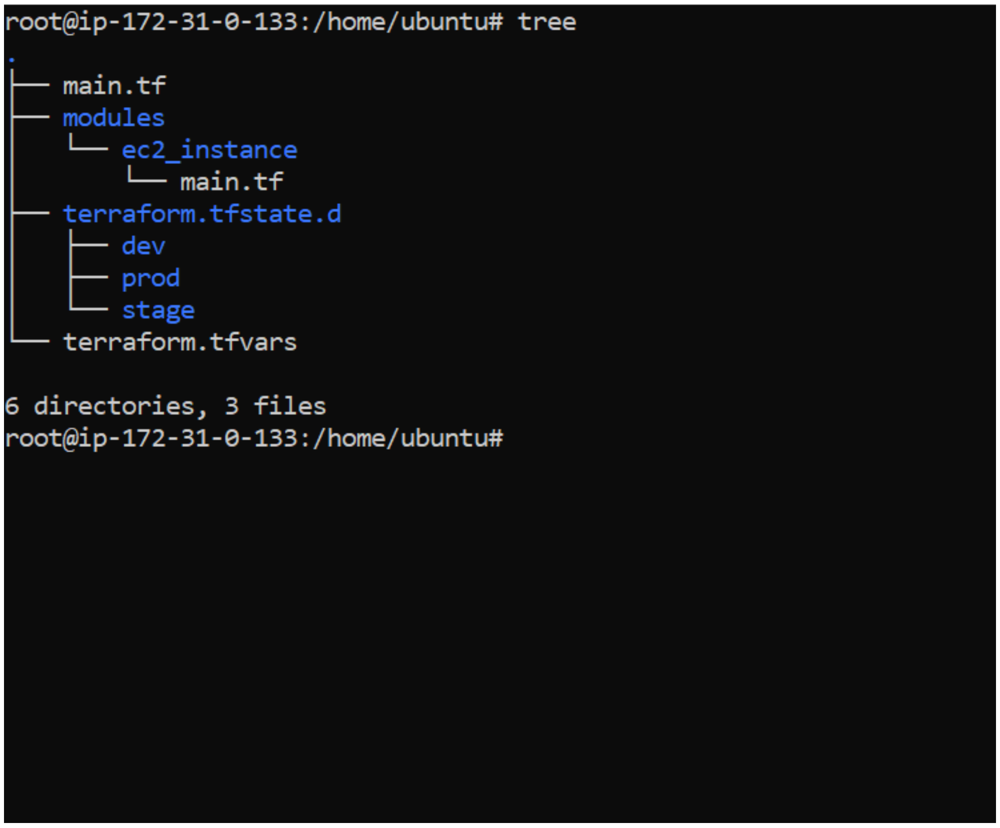
```
root@ip-172-31-0-133:/home/ubuntu# tree
.
├── main.tf
├── modules
│   └── ec2_instance
│       └── main.tf
├── terraform.tfstate.d
│   └── dev
└── terraform.tfvars

4 directories, 3 files
root@ip-172-31-0-133:/home/ubuntu#
```



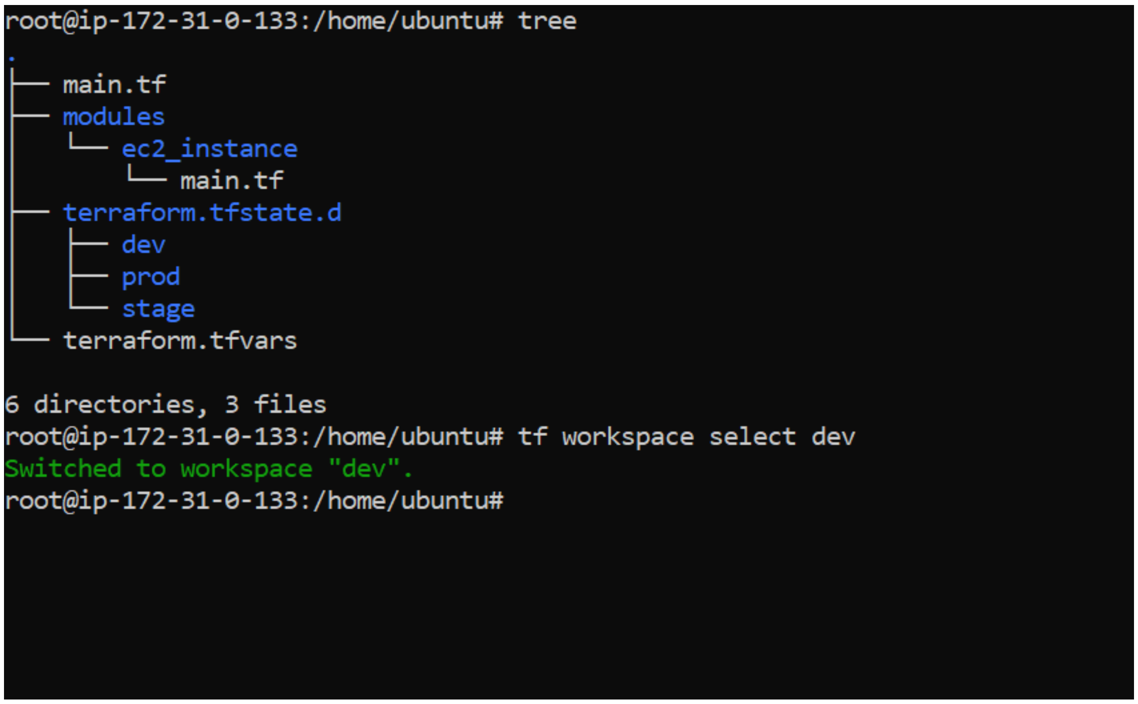
```
root@ip-172-31-0-133:/home/ubuntu# tf workspace new stage
Created and switched to workspace "stage"!

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.
```



```
root@ip-172-31-0-133:/home/ubuntu# tf 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.
root@ip-172-31-0-133:/home/ubuntu#
```

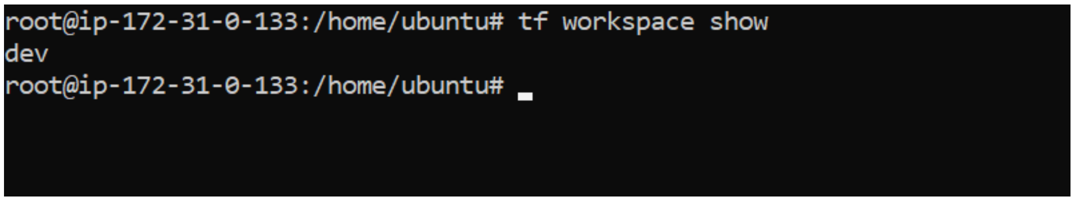


```
root@ip-172-31-0-133:/home/ubuntu# tree
.
├── main.tf
├── modules
│   └── ec2_instance
│       └── main.tf
├── terraform.tfstate.d
│   ├── dev
│   ├── prod
│   └── stage
└── terraform.tfvars

6 directories, 3 files
root@ip-172-31-0-133:/home/ubuntu#
```



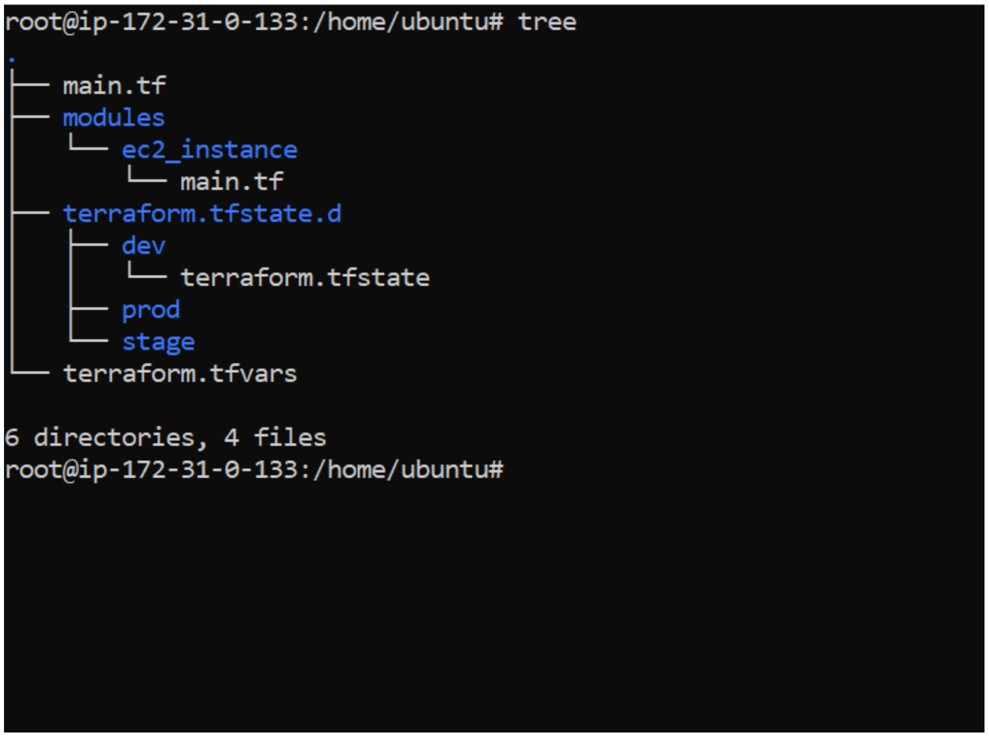
```
root@ip-172-31-0-133:/home/ubuntu# tree
.
├── main.tf
├── modules
│   └── ec2_instance
│       └── main.tf
├── terraform.tfstate.d
│   ├── dev
│   ├── prod
│   └── stage
└── terraform.tfvars

6 directories, 3 files
root@ip-172-31-0-133:/home/ubuntu# tf workspace select dev
Switched to workspace "dev".
root@ip-172-31-0-133:/home/ubuntu#
```

to go inside the dev, here is the command below



```
root@ip-172-31-0-133:/home/ubuntu# tf workspace show
dev
root@ip-172-31-0-133:/home/ubuntu#
```

to view where you are



```
root@ip-172-31-0-133:/home/ubuntu# tree
.
├── main.tf
├── modules
│   └── ec2_instance
│       └── main.tf
├── terraform.tfstate.d
│   ├── dev
│   │   └── terraform.tfstate
│   ├── prod
│   └── stage
└── terraform.tfvars

6 directories, 4 files
root@ip-172-31-0-133:/home/ubuntu#
```

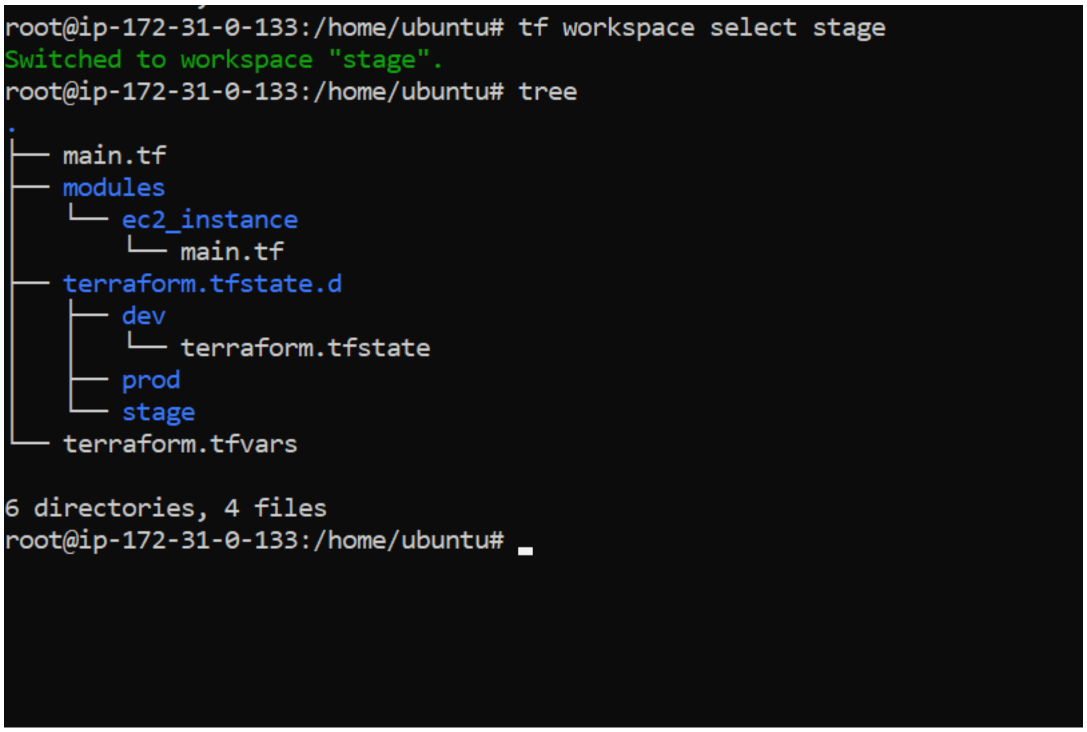
>terraform init (you are in dev)

>terraform apply (infra will be created)

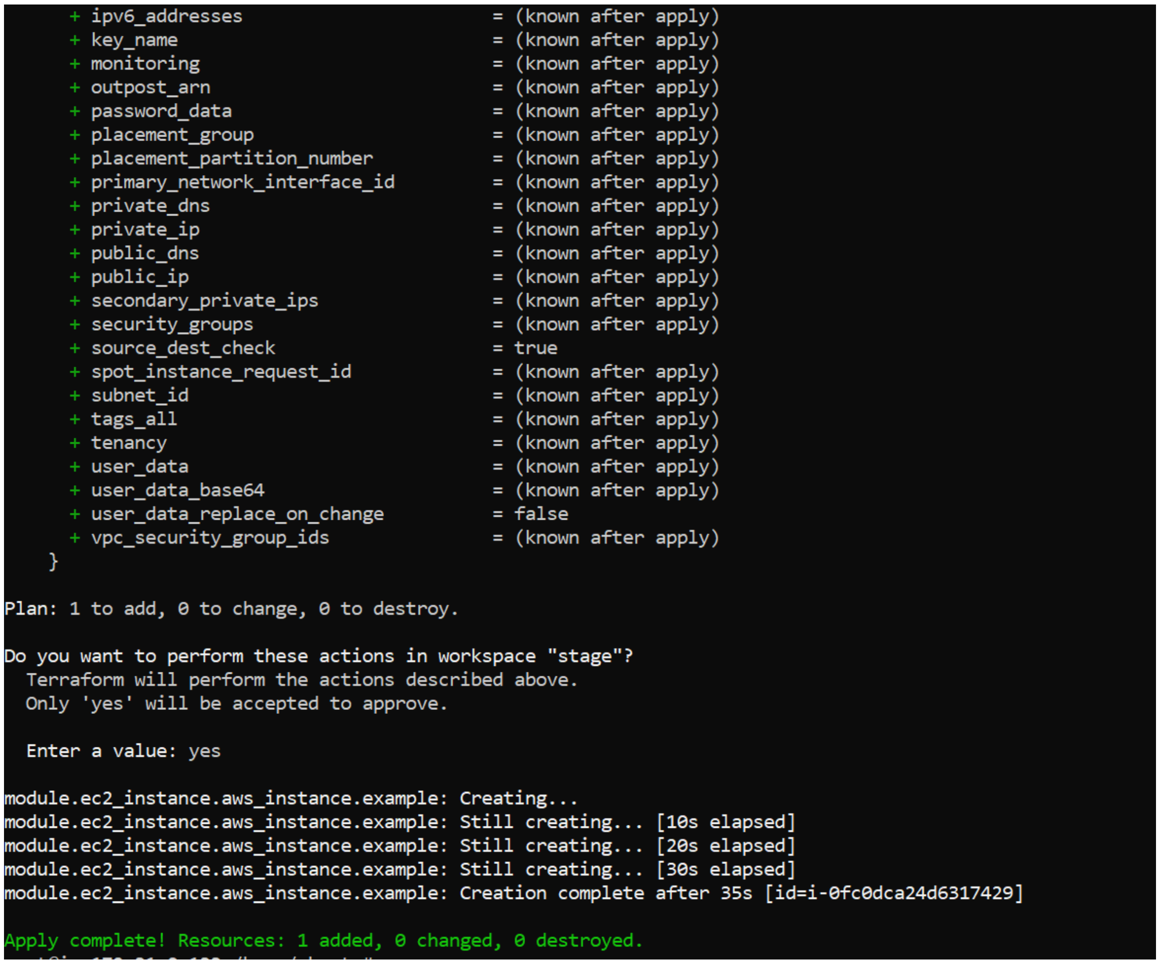
- then you will see instance will be there in dev environment. terraform.tfstate file will not be created in whole root folder it will create in only dev folder separately so that it will prevent conflict. that is what we wanted exactly.

- instance is created is just for dev environment

>>now switching to the stage <<

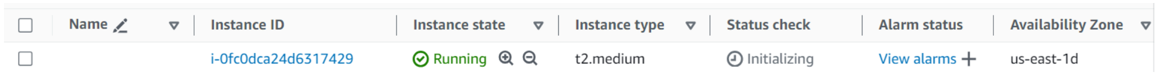

```
root@ip-172-31-0-133:/home/ubuntu# tf workspace select stage
Switched to workspace "stage".
root@ip-172-31-0-133:/home/ubuntu# tree
.
├── main.tf
├── modules
│   └── ec2_instance
│       └── main.tf
├── terraform.tfstate.d
│   ├── dev
│   │   └── terraform.tfstate
│   ├── prod
│   └── stage
└── terraform.tfvars

6 directories, 4 files
root@ip-172-31-0-133:/home/ubuntu#
```

- here will change instance type from '**terraform.tfvars'** file. will choose **'t2.medium'.** so that it will create medium instance in **'stage'** area.

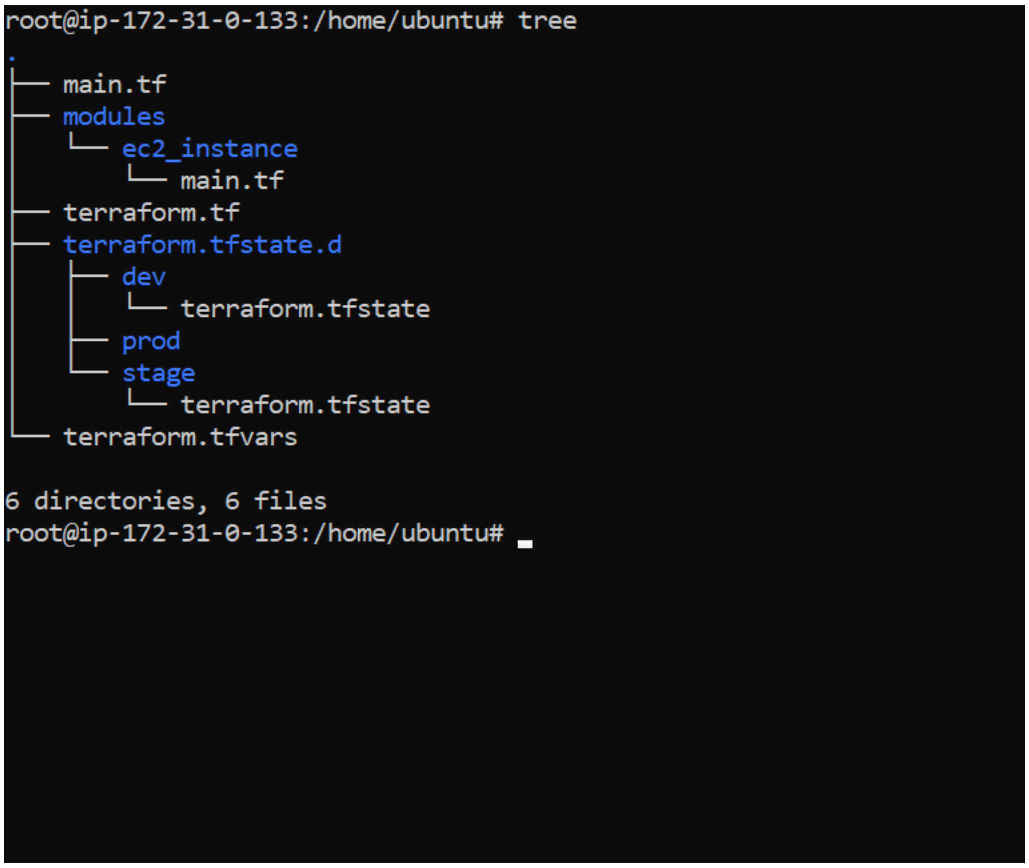


```
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags_all                             = (known after apply)
      + 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)
    }

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

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

  Enter a value: yes

module.ec2_instance.aws_instance.example: Creating...
module.ec2_instance.aws_instance.example: Still creating... [10s elapsed]
module.ec2_instance.aws_instance.example: Still creating... [20s elapsed]
module.ec2_instance.aws_instance.example: Still creating... [30s elapsed]
module.ec2_instance.aws_instance.example: Creation complete after 35s [id=i-0fc0dca24d6317429]

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

- medium size instance has been created in **'stage'** area.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☐		i-0fc0dca24d6317429	Running	t2.medium	Initializing	View alarms +	us-east-1d



```
root@ip-172-31-0-133:/home/ubuntu# tree
.
├── main.tf
├── modules
│   └── ec2_instance
│       └── main.tf
├── terraform.tf
├── terraform.tfstate.d
│   ├── dev
│   │   └── terraform.tfstate
│   ├── prod
│   └── stage
│       └── terraform.tfstate
└── terraform.tfvars

6 directories, 6 files
root@ip-172-31-0-133:/home/ubuntu# _
```

- even separate **'terraform.tftstate'** file has been created for **'stage'** area that is what we want.

is not it a time consuming task to update **'terraform.tfvars'** file again and again for instance size. as a devops engineer we want automation. so now for that instead of updating this file **'terraform.tfvars',** we can add each var files(desired instance type) in our multi deployment areas as follows,

>**Dev.tfvars**

>Prod.tfvars

>Stage.tfvars

*****