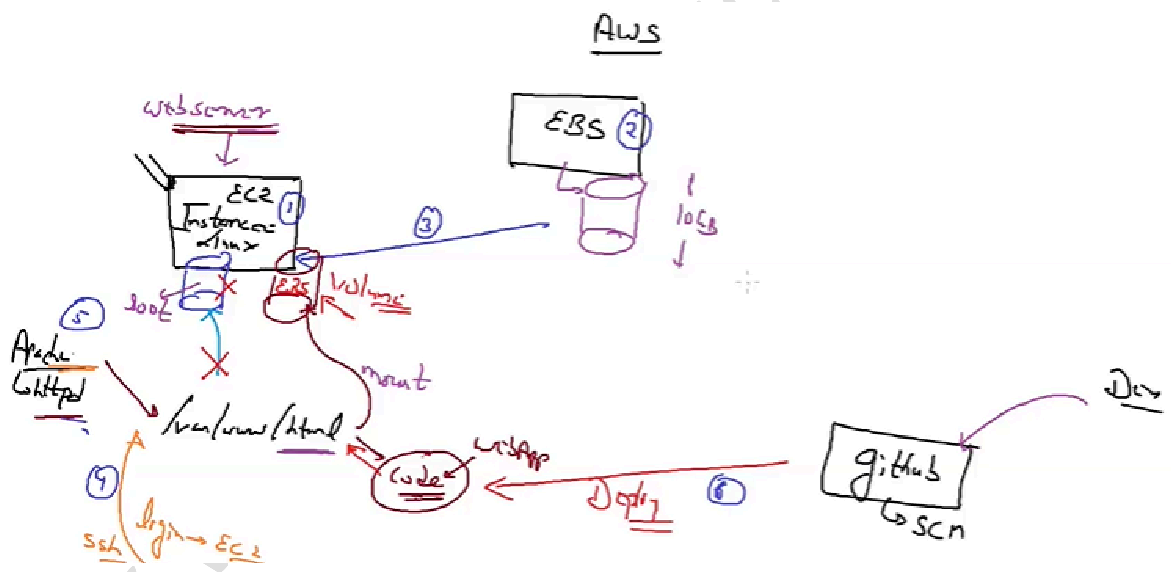
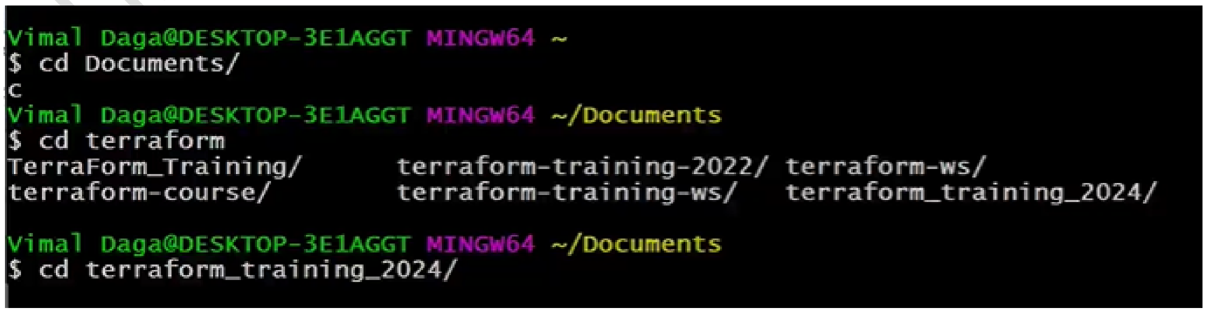
Terraform Session No. 03

Summary 25/04/2024

- Terraform is a tool that lets you manage your cloud infrastructure using code, making it easier to create, modify, and maintain digital resources like servers and networks. It simplifies the process by allowing you to describe your desired infrastructure in code, which Terraform then executes to bring your infrastructure to life.

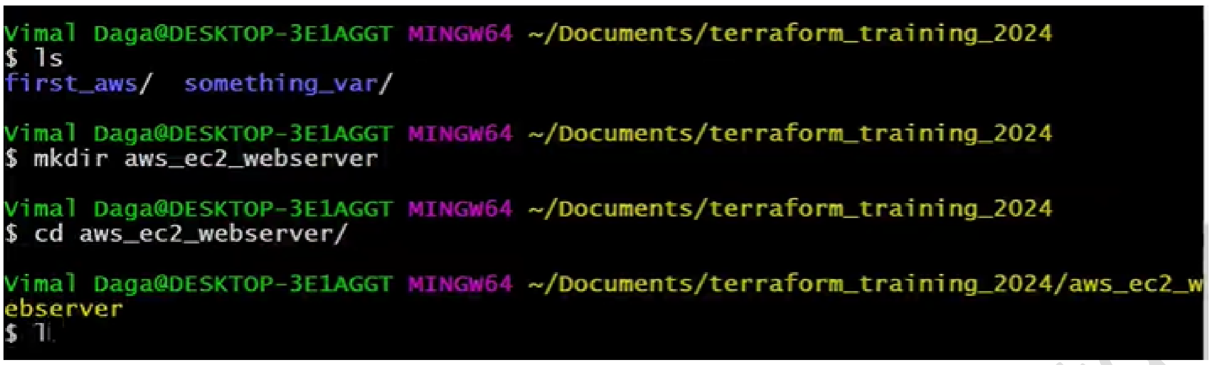


Go to inside the directory



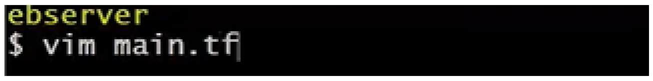
```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
$ cd Documents/
c
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents
$ cd terraform
TerraForm_Training/      terraform-training-2022/ terraform-ws/
terraform-course/        terraform-training-ws/   terraform_training_2024/

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents
$ cd terraform_training_2024/
```

Create a new directory for this project.

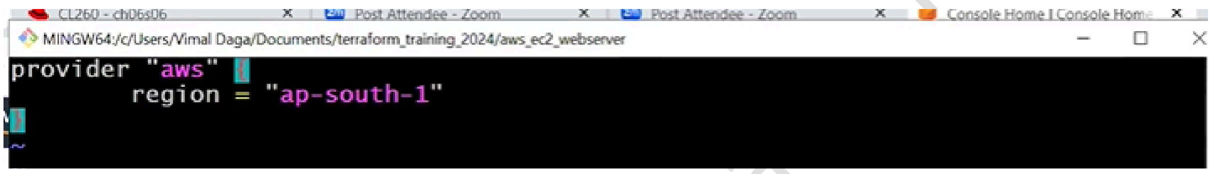


```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024
$ ls
first_aws/  something_var/

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024
$ mkdir aws_ec2_webserver

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024
$ cd aws_ec2_webserver/

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024/aws_ec2_w
ebserver
$ l
```

This command will create an empty file named "main" in your current directory.



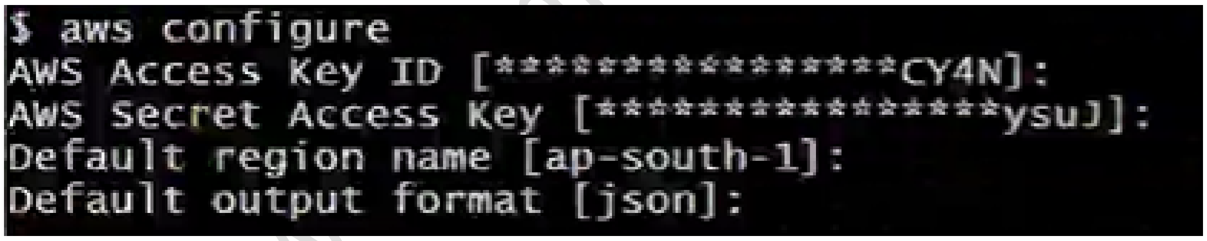
```
ebserver
$ vim main.tf
```



CL260 - ch06s06
Post Attendee - Zoom
Post Attendee - Zoom
Console Home | Console Home
MINGW64:/c/Users/Vimal Daga/Documents/terraform_training_2024/aws_ec2_webserver

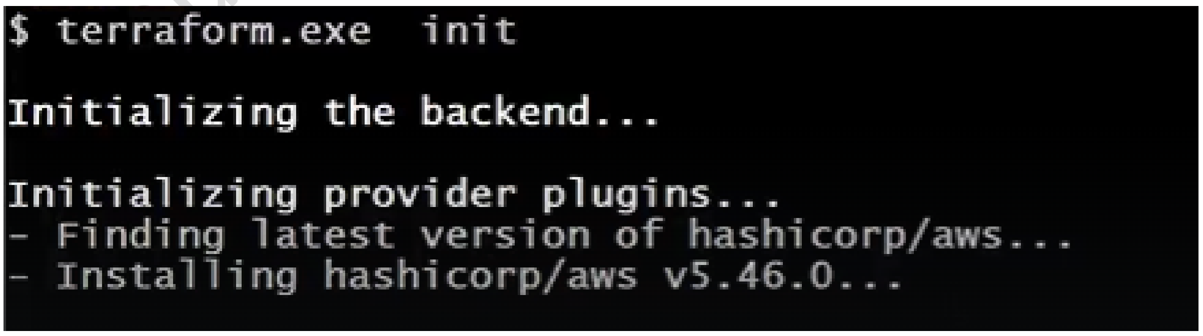
```
provider "aws" {
        region = "ap-south-1"
}
~
```

"aws configure" is the command used to set up AWS credentials on your computer, allowing you to interact with AWS services through the command line.



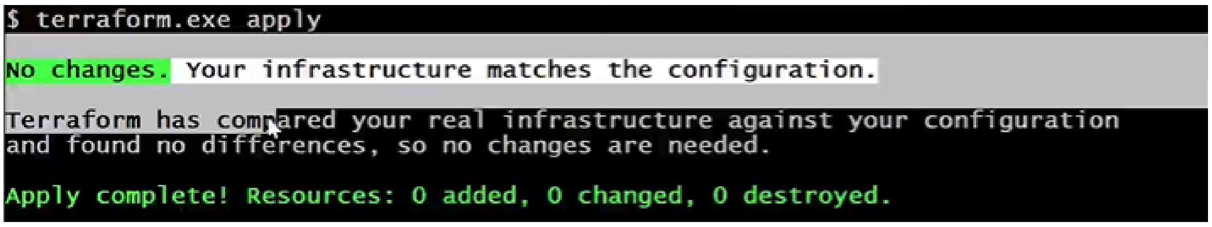
```
$ aws configure
AWS Access Key ID [****************CY4N]:
AWS Secret Access Key [****************ysuJ]:
Default region name [ap-south-1]:
Default output format [json]:
```

"terraform.exe init" initializes a Terraform project by downloading necessary plugins and modules.



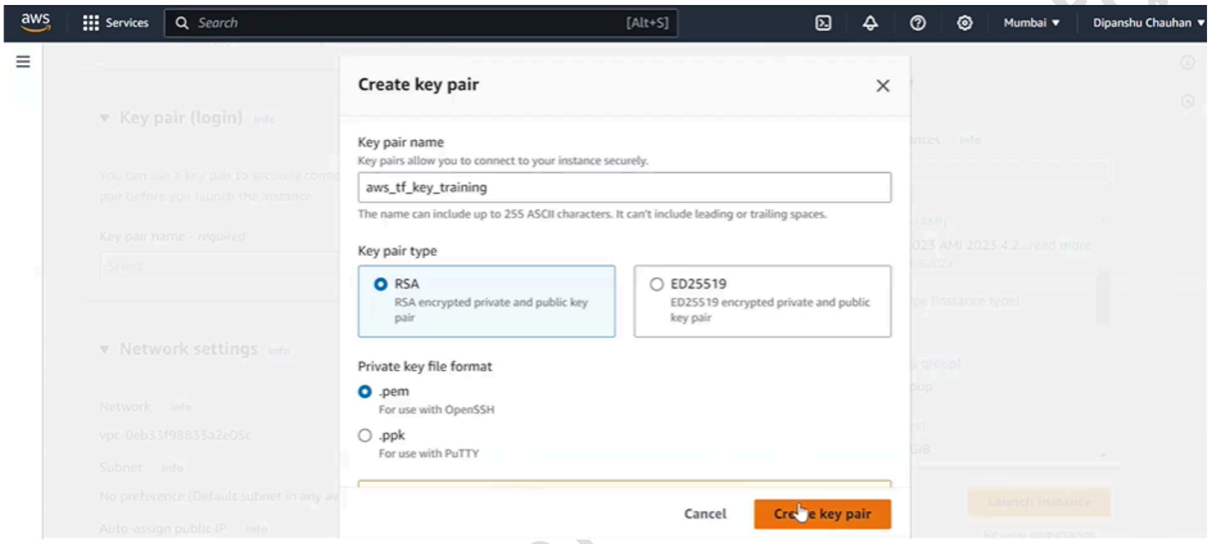
```
$ terraform.exe  init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.46.0...
```

"terraform.exe apply" deploys the infrastructure described in your Terraform configuration files.



```
$ terraform.exe apply

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration
and found no differences, so no changes are needed.

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

Let's create a key pair manually.



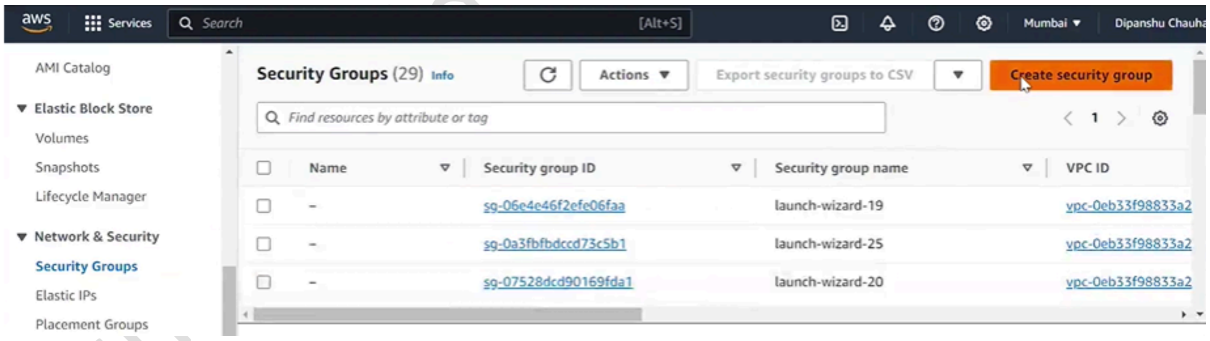
Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.
aws_tf_key_training
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type
[✓ RSA] RSA encrypted private and public key pair
ED25519 ED25519 encrypted private and public key pair

Private key file format
[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

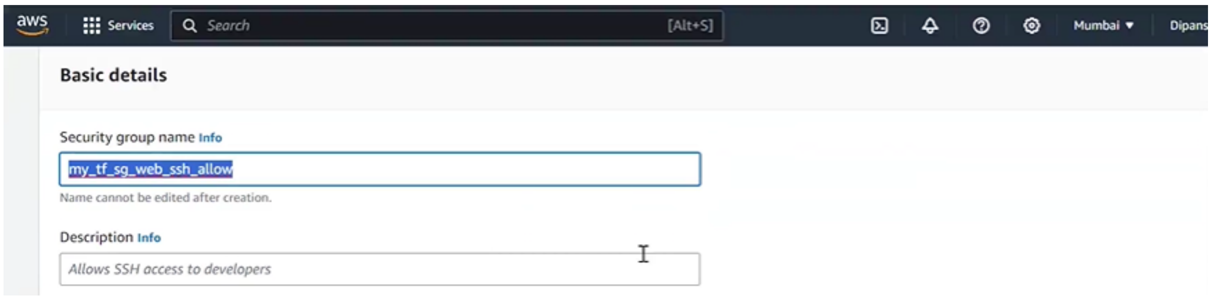
Also create a Security group.



Security Groups (29) Info

Name	Security group ID	Security group name	VPC ID
-	sg-06e4e46f2efe06faa	launch-wizard-19	vpc-0eb33f98833a2
-	sg-0a3fbfbdccd73c5b1	launch-wizard-25	vpc-0eb33f98833a2
-	sg-07528dcd90169fda1	launch-wizard-20	vpc-0eb33f98833a2

Give the name of Security group, you can give any name

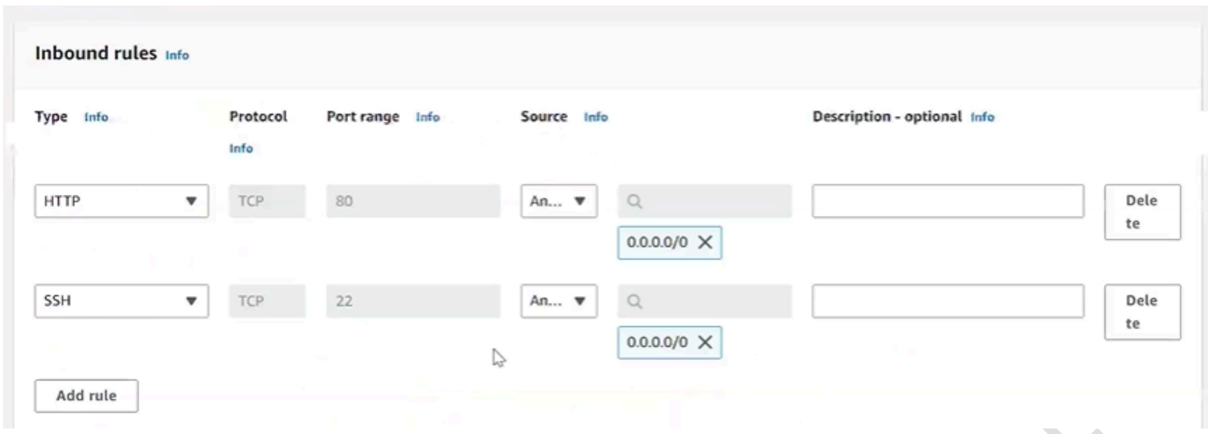


Basic details

Security group name Info
my_tf_sg_web_ssh_allow
Name cannot be edited after creation.

Description Info
Allows SSH access to developers

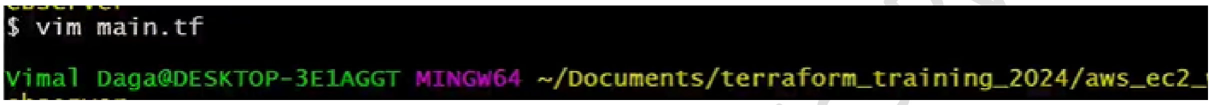
Here add two rules one is HTTP, and SSH



Inbound rules

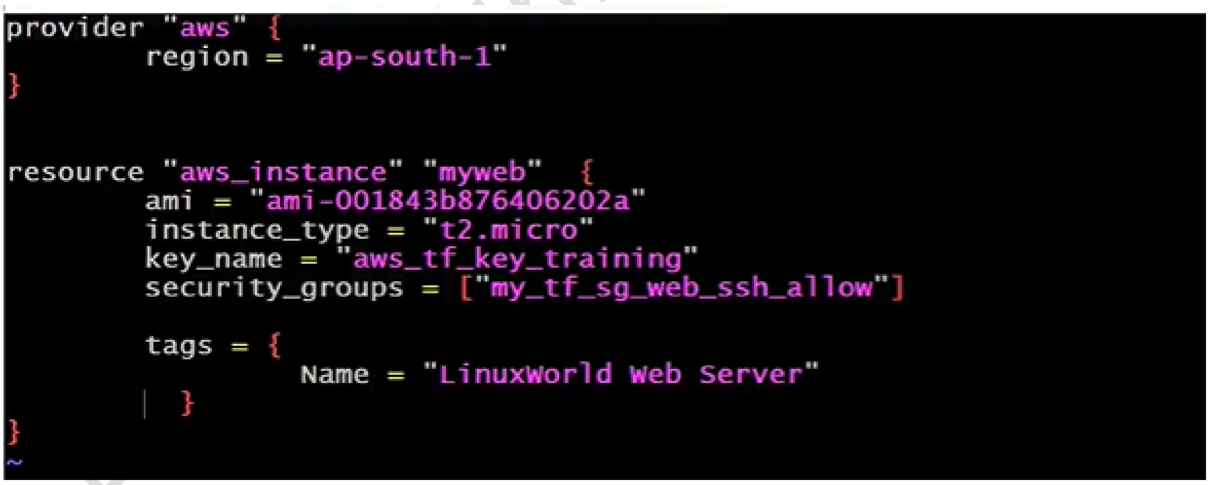
Type	Protocol	Port range	Source		Description - optional	
HTTP	TCP	80	An...	0.0.0.0/0		Delete
SSH	TCP	22	An...	0.0.0.0/0		Delete

Now open this previous created file.



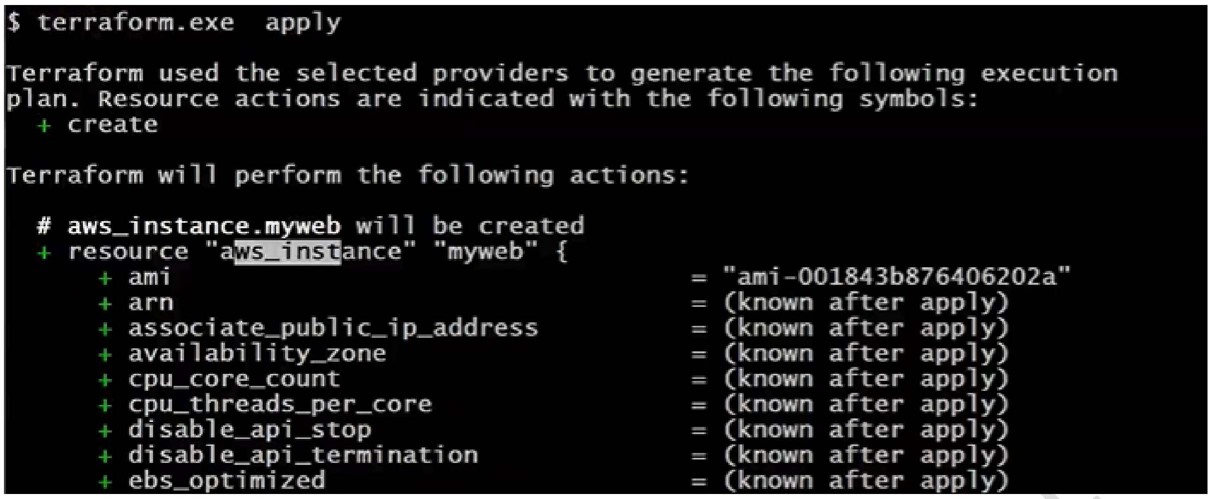
```
$ vim main.tf

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024/aws_ec2_
```

This Terraform code configures AWS as the provider in the "ap-south-1" region and creates an EC2 instance named "myweb". It specifies the AMI, instance type, key pair, security group, and adds a tag with the name "Linuxworld Web Server".



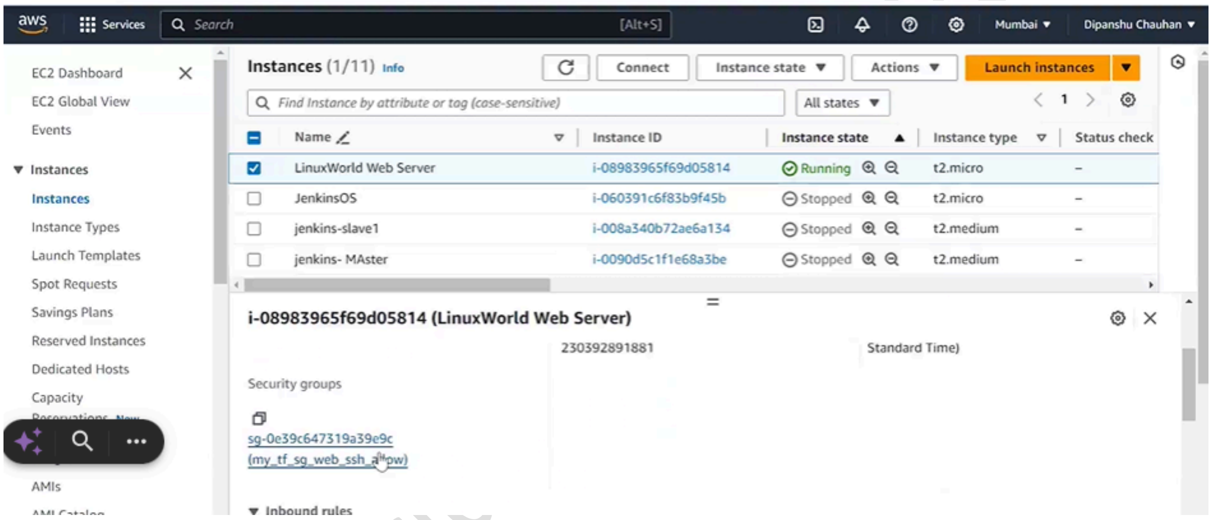
```
provider "aws" {
        region = "ap-south-1"
}

resource "aws_instance" "myweb"  {
        ami = "ami-001843b876406202a"
        instance_type = "t2.micro"
        key_name = "aws_tf_key_training"
        security_groups = ["my_tf_sg_web_ssh_allow"]

        tags = {
                Name = "LinuxWorld Web Server"
          }
}
~
```

"terraform.exe apply" this command will deploy infrastructure defined in Terraform configuration files.



```
$ terraform.exe  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.myweb will be created
  + resource "aws_instance" "myweb" {
      + ami                                  = "ami-001843b876406202a"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
```

Here you can see our instance is launched in the aws cloud.



Instances (1/11)

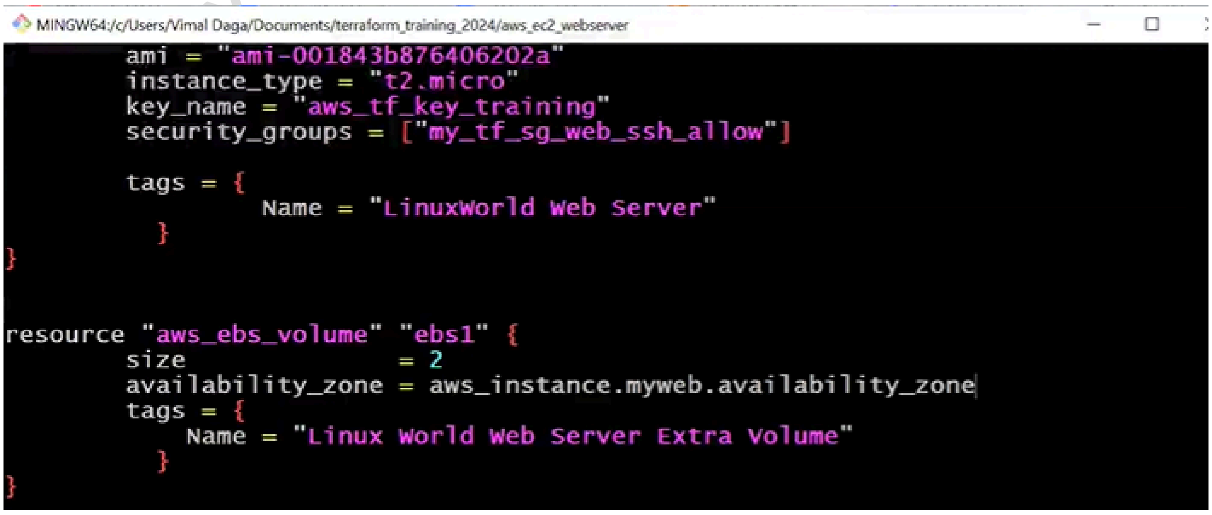
	Name	Instance ID	Instance state	Instance type	Status check
☑	LinuxWorld Web Server	i-08983965f69d05814	Running	t2.micro	–
☐	JenkinsOS	i-060391c6f83b9f45b	Stopped	t2.micro	–
☐	jenkins-slave1	i-008a340b72ae6a134	Stopped	t2.medium	–
☐	jenkins- MAster	i-0090d5c1f1e68a3be	Stopped	t2.medium	–

i-08983965f69d05814 (LinuxWorld Web Server)

Security groups

sg-0e39c647319a39e9c (my_tf_sg_web_ssh_allow)

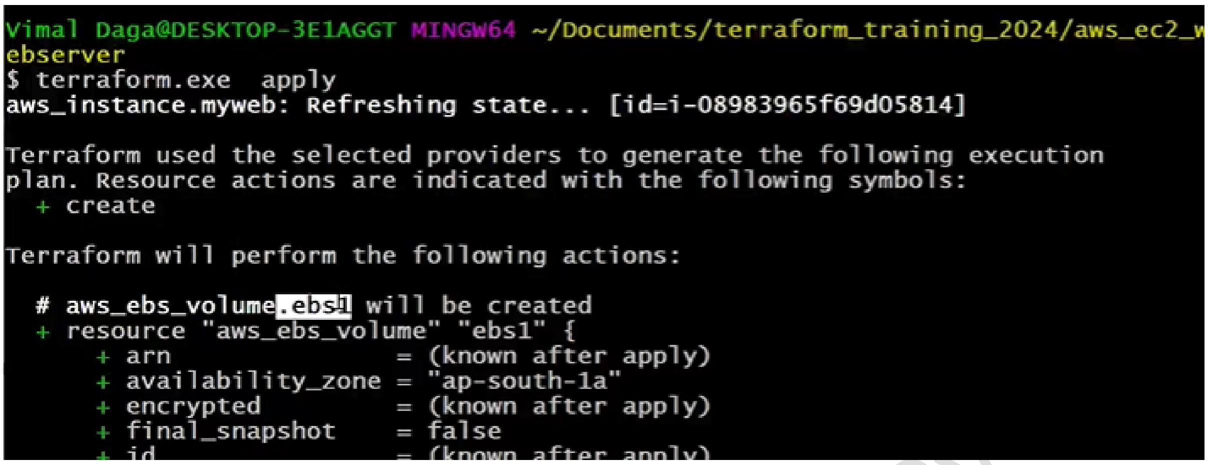
The new line added code creates an AWS EBS volume named "ebs1". It sets the size of the volume, specifies the availability zone where it should be created, and adds a tag with the name "Linux World Web Server Extra Volume".



MINGW64:/c/Users/Vimal Daga/Documents/terraform_training_2024/aws_ec2_webserver

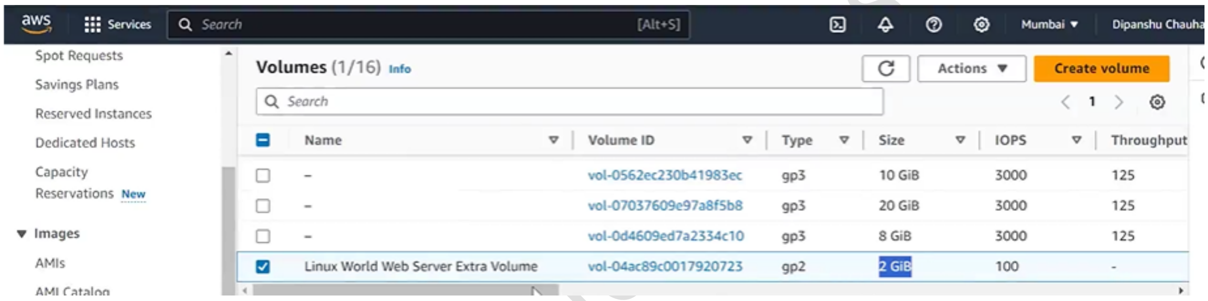
```
        ami = "ami-001843b876406202a"
        instance_type = "t2.micro"
        key_name = "aws_tf_key_training"
        security_groups = ["my_tf_sg_web_ssh_allow"]

        tags = {
                 Name = "LinuxWorld Web Server"
          }
}


resource "aws_ebs_volume" "ebs1" {
        size                 = 2
        availability_zone = aws_instance.myweb.availability_zone
        tags = {
             Name = "Linux World Web Server Extra Volume"
          }
}
```

"Whenever we make changes to the Terraform code, we need to execute the 'terraform.exe apply' command."



```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024/aws_ec2_w
ebserver
$ terraform.exe  apply
aws_instance.myweb: Refreshing state... [id=i-08983965f69d05814]

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_ebs_volume.ebs1 will be created
  + resource "aws_ebs_volume" "ebs1" {
      + arn               = (known after apply)
      + availability_zone = "ap-south-1a"
      + encrypted         = (known after apply)
      + final_snapshot    = false
      + id                = (known after apply)
```

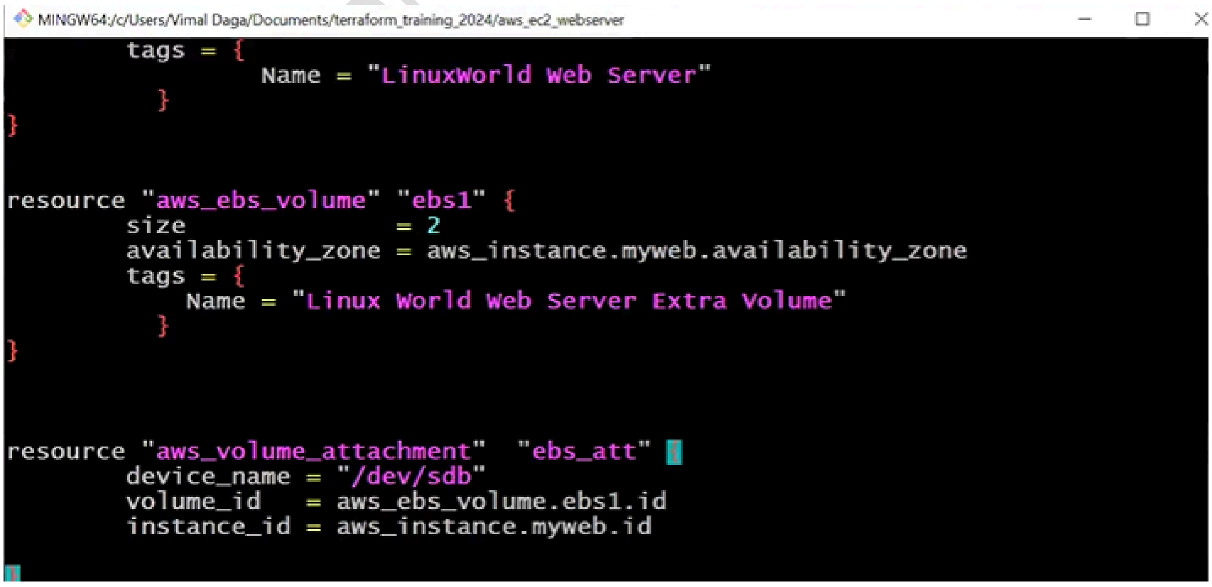
Here you can see 2GB size of EBS volume has been created



Volumes (1/16)

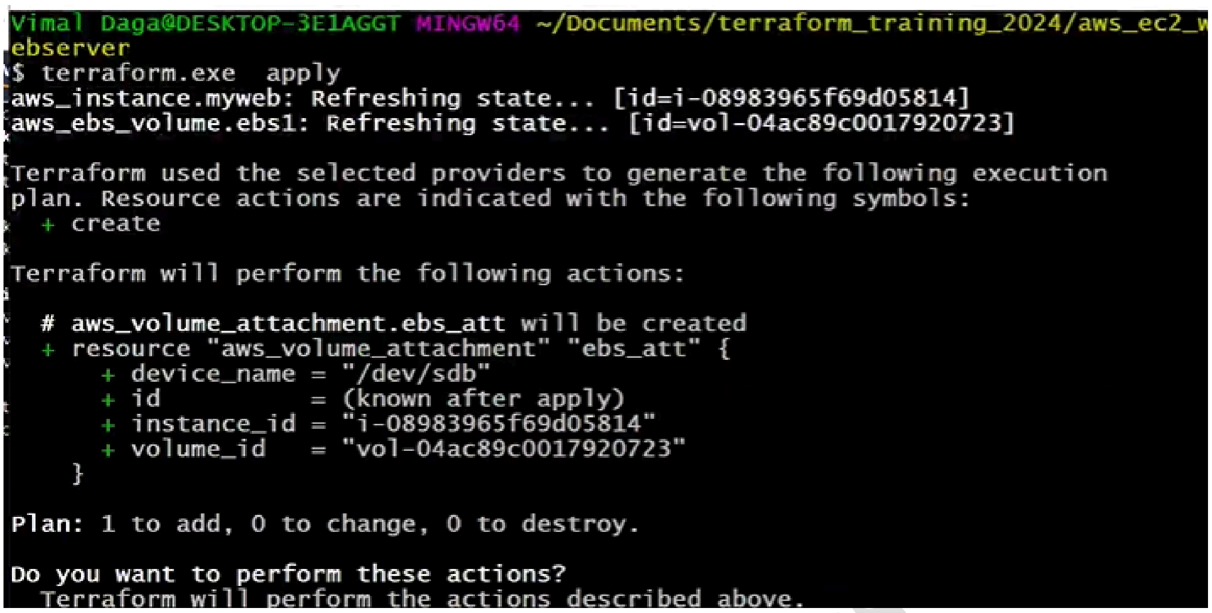
	Name	Volume ID	Type	Size	IOPS	Throughput
☐	-	vol-0562ec230b41983ec	gp3	10 GiB	3000	125
☐	-	vol-07037609e97a8f5b8	gp3	20 GiB	3000	125
☐	-	vol-0d4609ed7a2334c10	gp3	8 GiB	3000	125
☑	Linux World Web Server Extra Volume	vol-04ac89c0017920723	gp2	2 GiB	100	-

The added line attaches the AWS EBS volume "ebs1" to the EC2 instance "myweb" using the resource "aws_volume_attachment". It specifies the device name ("/dev/sdb"), the volume ID of "ebs1", and the instance ID of "myweb".



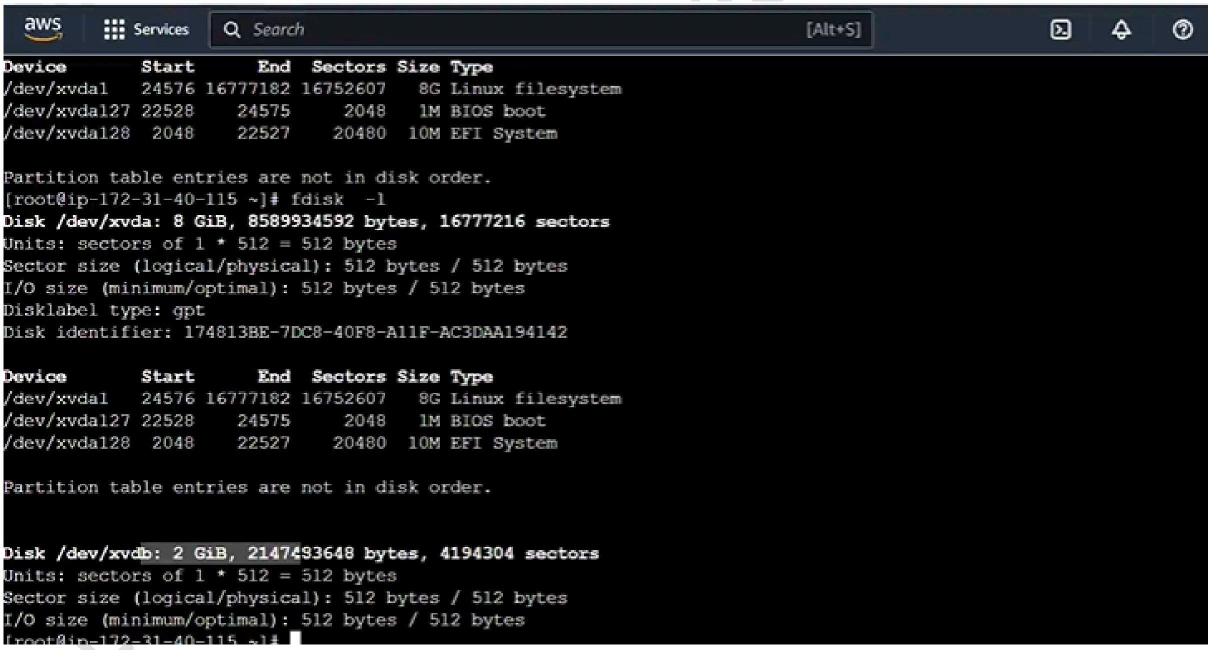
```
MINGW64:/c/Users/Vimal Daga/Documents/terraform_training_2024/aws_ec2_webserver
        tags = {
                 Name = "LinuxWorld Web Server"
          }
}


resource "aws_ebs_volume" "ebs1" {
        size              = 2
        availability_zone = aws_instance.myweb.availability_zone
        tags = {
            Name = "Linux World Web Server Extra Volume"
          }
}



resource "aws_volume_attachment"  "ebs_att" {
        device_name = "/dev/sdb"
        volume_id   = aws_ebs_volume.ebs1.id
        instance_id = aws_instance.myweb.id
```

Again run this apply command.



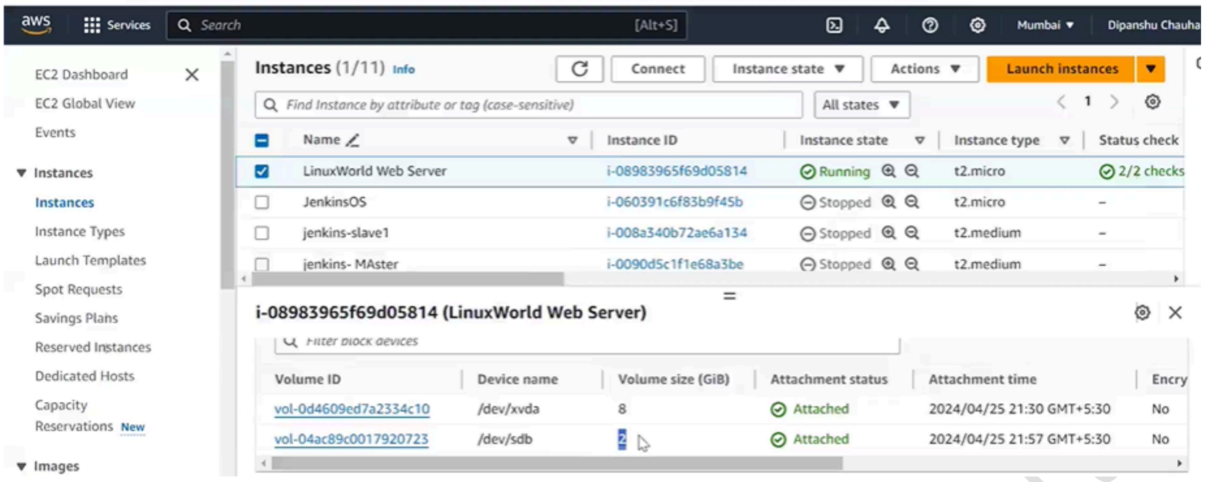
```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024/aws_ec2_w
ebserver
$ terraform.exe  apply
aws_instance.myweb: Refreshing state... [id=i-08983965f69d05814]
aws_ebs_volume.ebs1: Refreshing state... [id=vol-04ac89c0017920723]

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_volume_attachment.ebs_att will be created
  + resource "aws_volume_attachment" "ebs_att" {
      + device_name = "/dev/sdb"
      + id          = (known after apply)
      + instance_id = "i-08983965f69d05814"
      + volume_id   = "vol-04ac89c0017920723"
    }

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

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

You can verify that the EBS volume has been attached by running the "fdisk -l" command.



```
Device         Start      End  Sectors Size Type
/dev/xvda1     24576 16777182 16752607   8G Linux filesystem
/dev/xvda127   22528    24575     2048   1M BIOS boot
/dev/xvda128    2048    22527    20480  10M EFI System

Partition table entries are not in disk order.
[root@ip-172-31-40-115 ~]# fdisk  -l
Disk /dev/xvda: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 174813BE-7DC8-40F8-A11F-AC3DAA194142

Device         Start      End  Sectors Size Type
/dev/xvda1     24576 16777182 16752607   8G Linux filesystem
/dev/xvda127   22528    24575     2048   1M BIOS boot
/dev/xvda128    2048    22527    20480  10M EFI System

Partition table entries are not in disk order.


Disk /dev/xvdb: 2 GiB, 2147483648 bytes, 4194304 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[root@ip-172-31-40-115 ~]#
```

Alternatively, you can visually confirm the attachment of the EBS volume by checking the AWS Management Console.



Instances (1/11) Info

	Name	Instance ID	Instance state	Instance type	Status check
✓	LinuxWorld Web Server	i-08983965f69d05814	Running	t2.micro	2/2 checks
	JenkinsOS	i-060391c6f83b9f45b	Stopped	t2.micro	–
	jenkins-slave1	i-008a340b72ae6a134	Stopped	t2.medium	–
	jenkins- MAster	i-0090d5c1f1e68a3be	Stopped	t2.medium	–

i-08983965f69d05814 (LinuxWorld Web Server)

Volume ID	Device name	Volume size (GiB)	Attachment status	Attachment time	Encry
vol-0d4609ed7a2334c10	/dev/xvda	8	Attached	2024/04/25 21:30 GMT+5:30	No
vol-04ac89c0017920723	/dev/sdb	2	Attached	2024/04/25 21:57 GMT+5:30	No

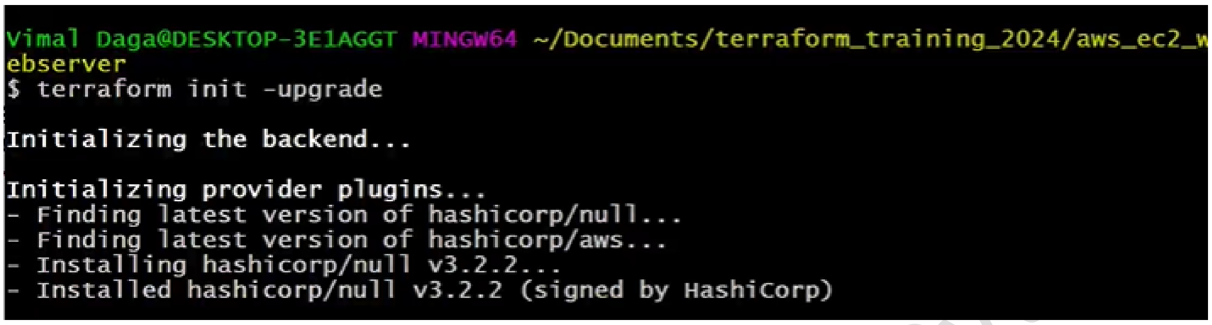
The new code block introduces a "null_resource" resource named "nullremote1", which allows executing remote commands on the EC2 instance "myweb". It uses the "remote-exec" provisioner to run commands like formatting the volume, installing Apache web server, mounting the volume, creating an index.html file, and restarting the httpd service. The "connection" block specifies SSH connection details including the user, private key, and host IP.

```
resource "null_resource" "nullremote1"  {
```

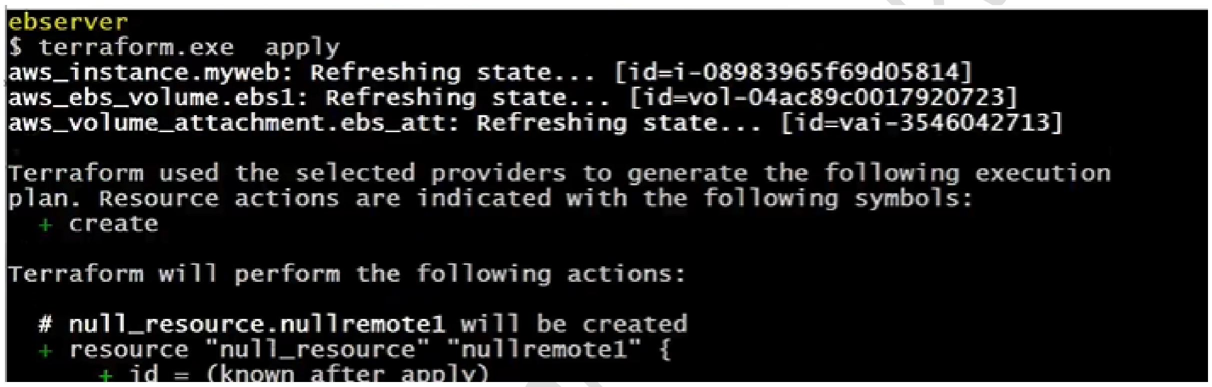
```
        provisioner "remote-exec"  {
                inline = [
                        "sudo mkfs.xfs  /dev/xvdb",
                        "sudo yum  install  httpd -y",
                        "sudo mount  /dev/xvdb  /var/www/html",
                        "sudo sh -c "echo 'welcome to LW' > /var/www/html/index.
html'"",
                        "sudo systemctl restart httpd"
                        ]
        }
```

```
        connection {
            type        = "ssh"
            user        = "ec2-user"
            private_key = file("C:/Users/Vimal Daga/Downloads/aws_tf_key_trainin
g.pem")
            host        = aws_instance.myweb.public_ip
                }

}
```

"terraform init -upgrade" is the command to initialize Terraform with plugin upgrades.

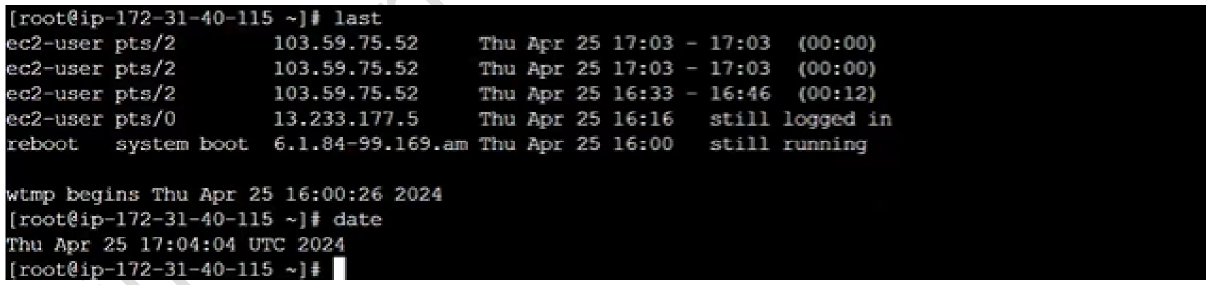


```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024/aws_ec2_w
ebserver
$ terraform init -upgrade

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/null...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/null v3.2.2...
- Installed hashicorp/null v3.2.2 (signed by HashiCorp)
```



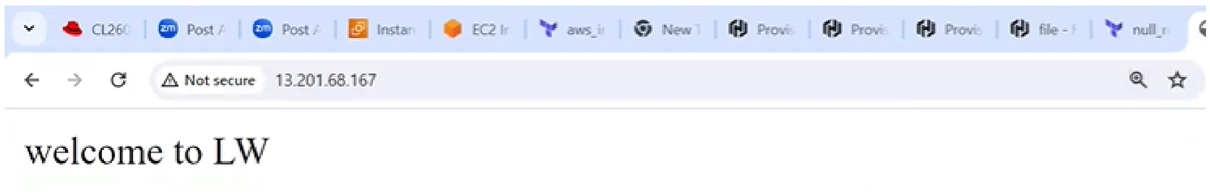
```
ebserver
$ terraform.exe  apply
aws_instance.myweb: Refreshing state... [id=i-08983965f69d05814]
aws_ebs_volume.ebs1: Refreshing state... [id=vol-04ac89c0017920723]
aws_volume_attachment.ebs_att: Refreshing state... [id=vai-3546042713]

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:

  # null_resource.nullremote1 will be created
  + resource "null_resource" "nullremote1" {
      + id = (known after apply)
```

The "last" command in Linux displays a list of recent login sessions, including information about users, terminal sessions, login times, and durations.



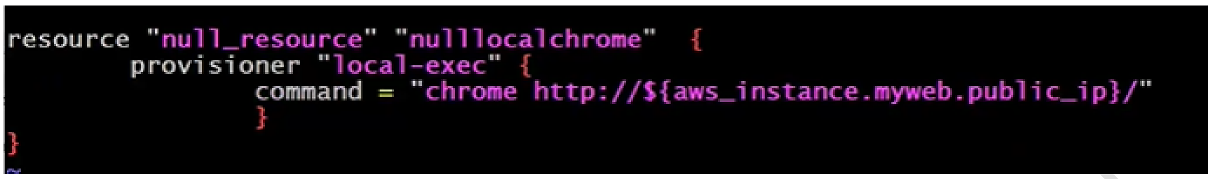
```
[root@ip-172-31-40-115 ~]# last
ec2-user pts/2        103.59.75.52     Thu Apr 25 17:03 - 17:03  (00:00)
ec2-user pts/2        103.59.75.52     Thu Apr 25 17:03 - 17:03  (00:00)
ec2-user pts/2        103.59.75.52     Thu Apr 25 16:33 - 16:46  (00:12)
ec2-user pts/0        13.233.177.5     Thu Apr 25 16:16   still logged in
reboot   system boot  6.1.84-99.169.am Thu Apr 25 16:00   still running

wtmp begins Thu Apr 25 16:00:26 2024
[root@ip-172-31-40-115 ~]# date
Thu Apr 25 17:04:04 UTC 2024
[root@ip-172-31-40-115 ~]#
```

With the EC2 instance's public IP, anyone can access the web page hosted on that instance using a web browser.

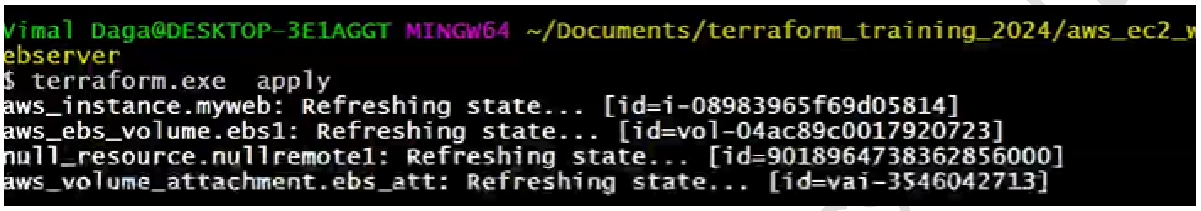


welcome to LW

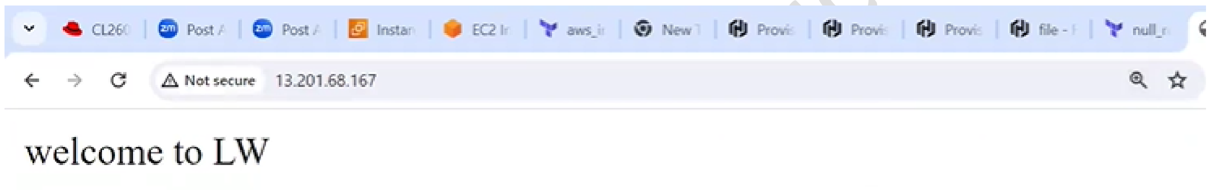
The new code block introduces another "null_resource" named "nulllocalchrome", which utilizes a "local-exec" provisioner. This provisioner executes a local command to launch the Chrome browser and access the web page hosted on the EC2 instance using its public IP address.



```
resource "null_resource" "nulllocalchrome"  {
        provisioner "local-exec" {
                command = "chrome http://${aws_instance.myweb.public_ip}/"
                }
}
```



```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/terraform_training_2024/aws_ec2_w
ebserver
$ terraform.exe  apply
aws_instance.myweb: Refreshing state... [id=i-08983965f69d05814]
aws_ebs_volume.ebs1: Refreshing state... [id=vol-04ac89c0017920723]
null_resource.nullremote1: Refreshing state... [id=9018964738362856000]
aws_volume_attachment.ebs_att: Refreshing state... [id=vai-3546042713]
```



welcome to LW

This Terraform project sets up an AWS EC2 instance ("myweb") with a specified AMI and instance type in the "ap-south-1" region. It attaches an EBS volume ("ebs1") to the instance and configures an Apache web server on it. The project then launches Chrome locally to access the web page hosted on the EC2 instance via its public IP address.