Module 8: Terraform   Assignment-2

Tasks To Be Performed:

1. Destroy the previous deployment

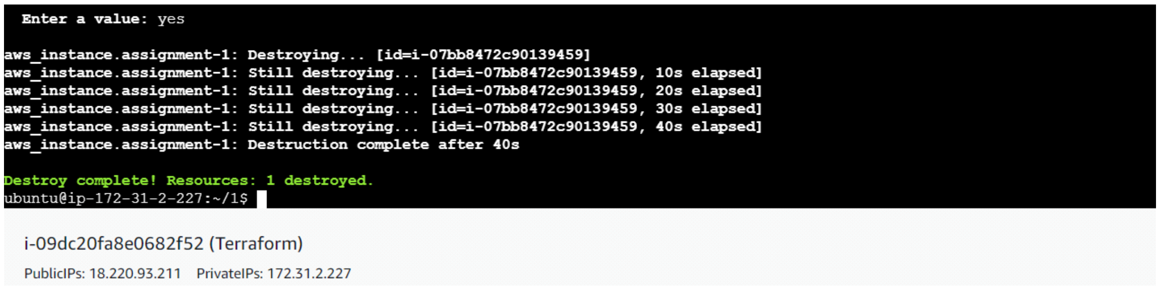
2. Create a new EC2 instance with an Elastic IP

Solution:

Step 1: Destroy the Previous Deployment

To destroy the previous deployment, navigate to the directory containing your Terraform configuration files and run the following command:

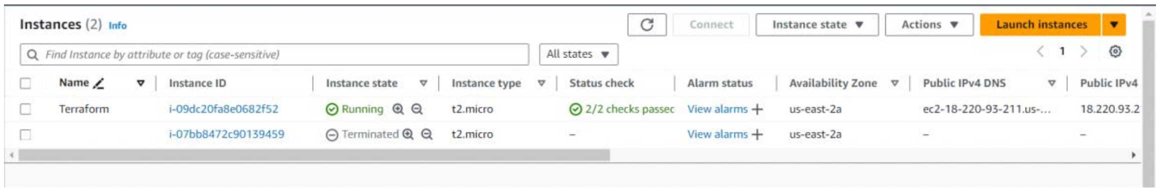
terraform destroy



```
  Enter a value: yes

aws_instance.assignment-1: Destroying... [id=i-07bb8472c90139459]
aws_instance.assignment-1: Still destroying... [id=i-07bb8472c90139459, 10s elapsed]
aws_instance.assignment-1: Still destroying... [id=i-07bb8472c90139459, 20s elapsed]
aws_instance.assignment-1: Still destroying... [id=i-07bb8472c90139459, 30s elapsed]
aws_instance.assignment-1: Still destroying... [id=i-07bb8472c90139459, 40s elapsed]
aws_instance.assignment-1: Destruction complete after 40s

Destroy complete! Resources: 1 destroyed.
ubuntu@ip-172-31-2-227:~/1$
```

i-09dc20fa8e0682f52 (Terraform)

PublicIPs: 18.220.93.211   PrivateIPs: 172.31.2.227



Instances (2)

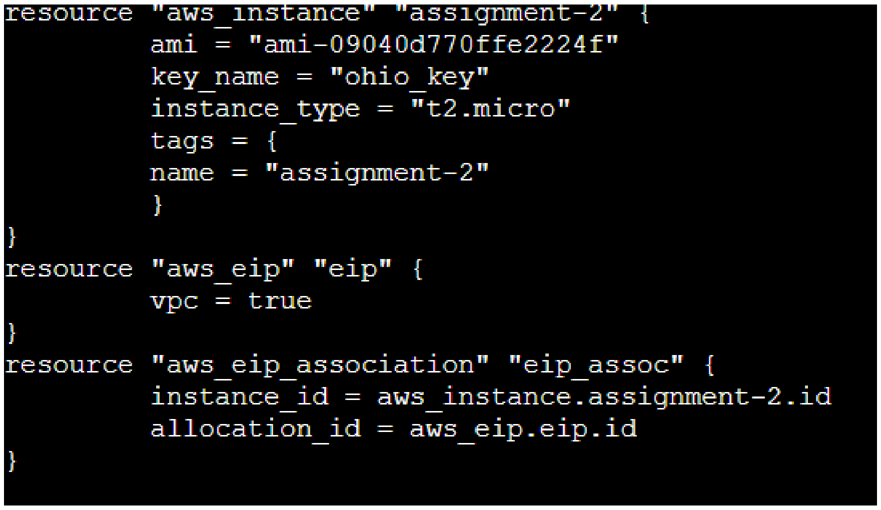
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
☐	Terraform	i-09dc20fa8e0682f52	Running	t2.micro	2/2 checks passec	View alarms	us-east-2a	ec2-18-220-93-211.us-...	18.220.93.2
☐		i-07bb8472c90139459	Terminated	t2.micro	–	View alarms	us-east-2a	–	–

Step 2: Write the Terraform Configuration

Create a directory for your Terraform configuration files. Inside this directory, create a file named tf2.tf and add the following configuration:

provider "aws" {

```
        region = "us-east-2"

        access_key = " "

        secret_key = " "


}

resource "aws_instance" "assignment-2" {

        ami = "ami-09040d770ffe2224f"

        key_name = "ohio_key"

        instance_type = "t2.micro"

        tags = {

        name = "assignment-2"

        }

}

resource "aws_eip" "eip" {

        vpc = true

}

resource "aws_eip_association" "eip_assoc" {

        instance_id = aws_instance.assignment-2.id

        allocation_id = aws_eip.eip.id

}
```

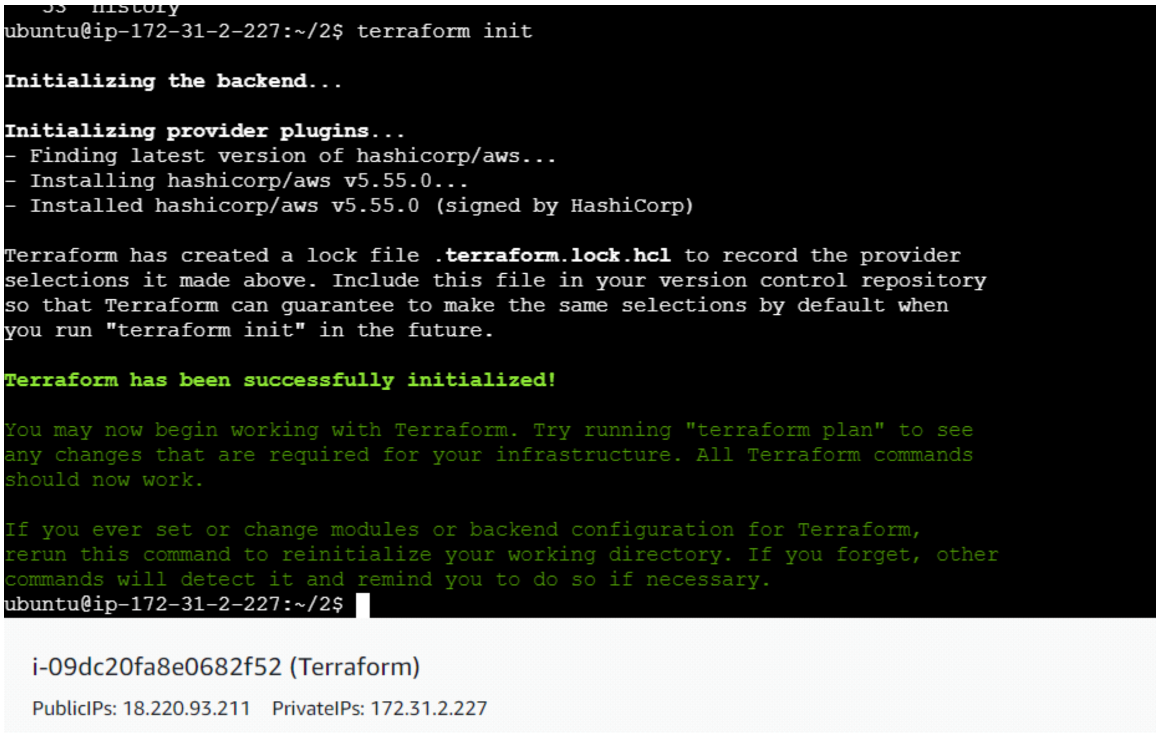


```
resource "aws_instance" "assignment-2" {
         ami = "ami-09040d770ffe2224f"
         key_name = "ohio_key"
         instance_type = "t2.micro"
         tags = {
         name = "assignment-2"
         }
}
resource "aws_eip" "eip" {
         vpc = true
}
resource "aws_eip_association" "eip_assoc" {
         instance_id = aws_instance.assignment-2.id
         allocation_id = aws_eip.eip.id
}
```

Step 3: Initialize Terraform

Run the following command to initialize Terraform. This will download the necessary provider plugins:

terraform init



```
   55  history
ubuntu@ip-172-31-2-227:~/2$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.55.0...
- Installed hashicorp/aws v5.55.0 (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.
ubuntu@ip-172-31-2-227:~/2$
```

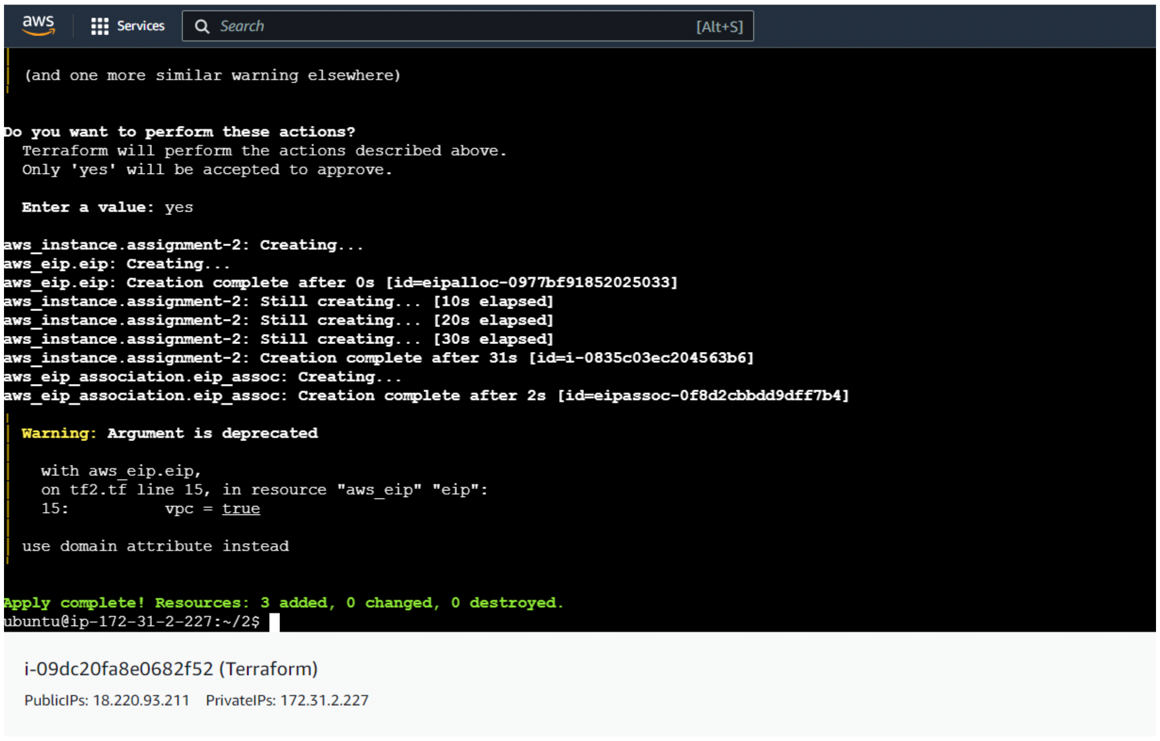
i-09dc20fa8e0682f52 (Terraform)

PublicIPs: 18.220.93.211   PrivateIPs: 172.31.2.227

Step 4: Apply the Terraform Configuration

Run the following command to create the EC2 instance:

terraform apply



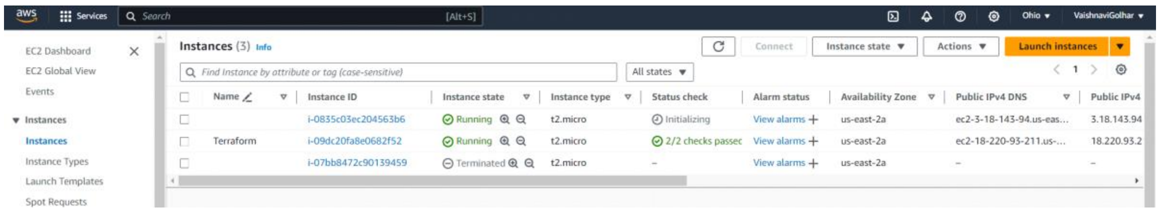
```
 (and one more similar warning elsewhere)


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

aws_instance.assignment-2: Creating...
aws_eip.eip: Creating...
aws_eip.eip: Creation complete after 0s [id=eipalloc-0977bf91852025033]
aws_instance.assignment-2: Still creating... [10s elapsed]
aws_instance.assignment-2: Still creating... [20s elapsed]
aws_instance.assignment-2: Still creating... [30s elapsed]
aws_instance.assignment-2: Creation complete after 31s [id=i-0835c03ec204563b6]
aws_eip_association.eip_assoc: Creating...
aws_eip_association.eip_assoc: Creation complete after 2s [id=eipassoc-0f8d2cbbdd9dff7b4]

  Warning: Argument is deprecated

    with aws_eip.eip,
    on tf2.tf line 15, in resource "aws_eip" "eip":
    15:         vpc = true

  use domain attribute instead


Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-2-227:~/2$
```

i-09dc20fa8e0682f52 (Terraform)

PublicIPs: 18.220.93.211   PrivateIPs: 172.31.2.227

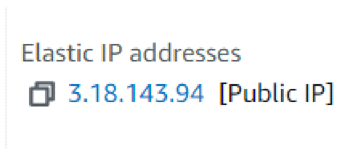
Step 5: Verify the New Deployment

Once the Terraform apply command completes, you can verify the new EC2 instance and its associated Elastic IP by logging into the AWS Management Console and navigating to the EC2 dashboard in the Ohio region.



Instances (3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
	i-0835c03ec204563b6	Running	t2.micro	Initializing	View alarms	us-east-2a	ec2-3-18-143-94.us-eas...	3.18.143.94
Terraform	i-09dc20fa8e0682f52	Running	t2.micro	2/2 checks passec	View alarms	us-east-2a	ec2-18-220-93-211.us-...	18.220.93.2
	i-07bb8472c90139459	Terminated	t2.micro	–	View alarms	us-east-2a	–	–



Elastic IP addresses

3.18.143.94 [Public IP]