Module 8: Terraform Assignment
Terraform Assignment -1

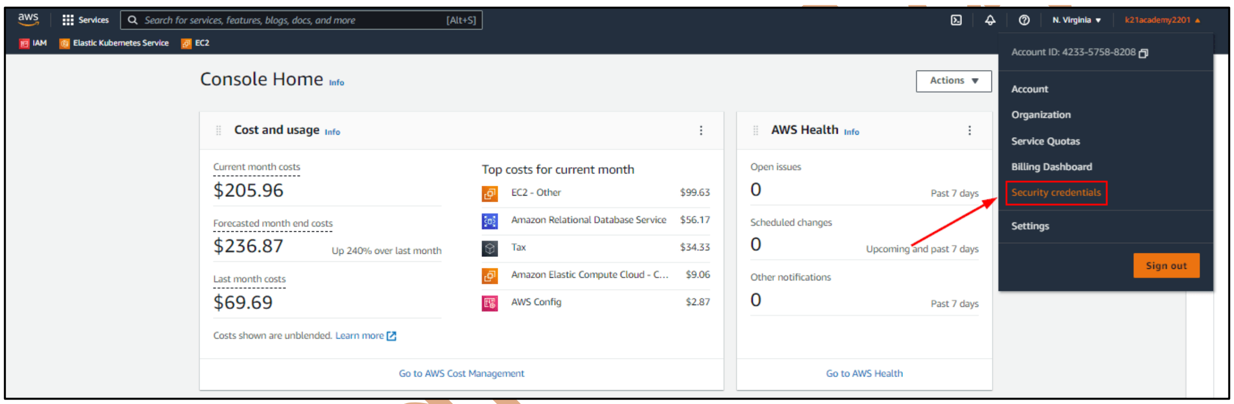
Tasks To Be Performed:

1. Create an EC2 service in the default subnet in the Ohio region

Step 1 – First we need to create access id and security key for perform any terraform task

Steps to create AWS Access key

1) Go to the **AWS management console**, click on your Profile name, and then click on **My Security Credentials**.



Cost and usage Info

Current month costs $205.96

Forecasted month end costs $236.87 Up 240% over last month

Last month costs $69.69

Costs shown are unblended. Learn more

Top costs for current month

EC2 - Other	$99.63
Amazon Relational Database Service	$56.17
Tax	$34.33
Amazon Elastic Compute Cloud - C...	$9.06
AWS Config	$2.87

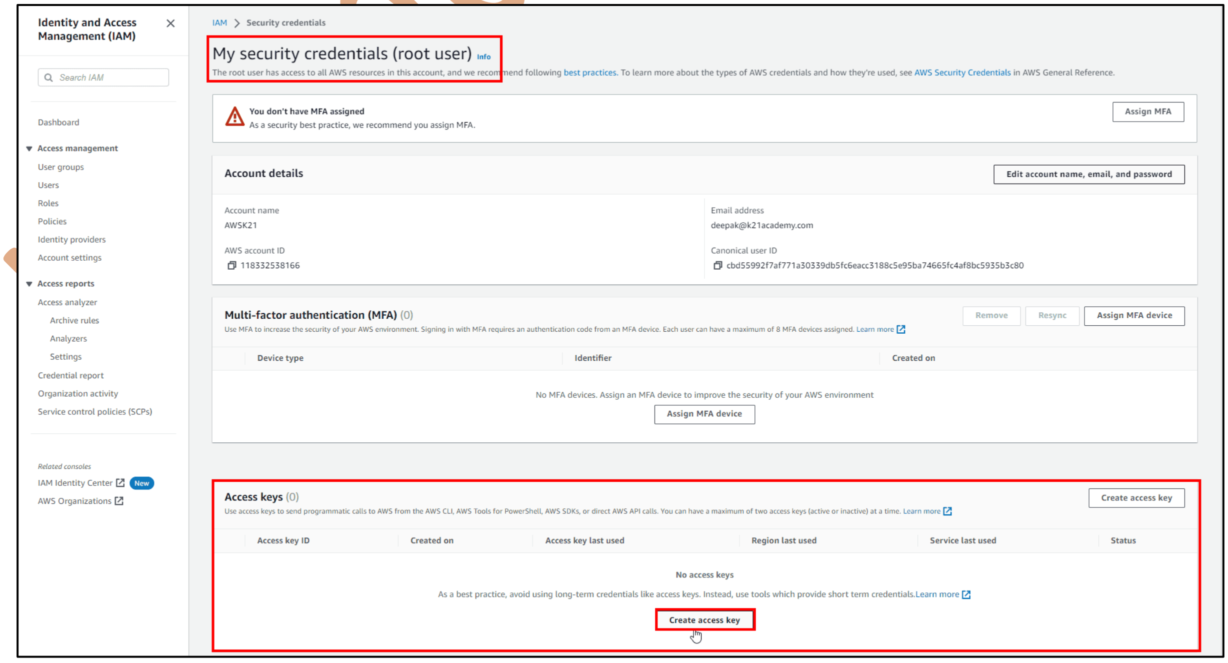
AWS Health Info

Open issues 0 Past 7 days

Scheduled changes 0 Upcoming and past 7 days

Other notifications 0 Past 7 days

2) Go to Access Keys and select Create New Access Key.



My security credentials (root user) Info

The root user has access to all AWS resources in this account, and we recommend following best practices. To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference.

You don't have MFA assigned
As a security best practice, we recommend you assign MFA.

Account details

Account name: AWSK21
AWS account ID: 118332538166
Email address: deepak@k21academy.com
Canonical user ID: cbd55992f7af771a30339db5fc6eacc3188c5e95ba74665fc4af8bc5935b3c80

Multi-factor authentication (MFA) (0)

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. Learn more

Device type	Identifier	Created on

No MFA devices. Assign an MFA device to improve the security of your AWS environment

Access keys (0)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

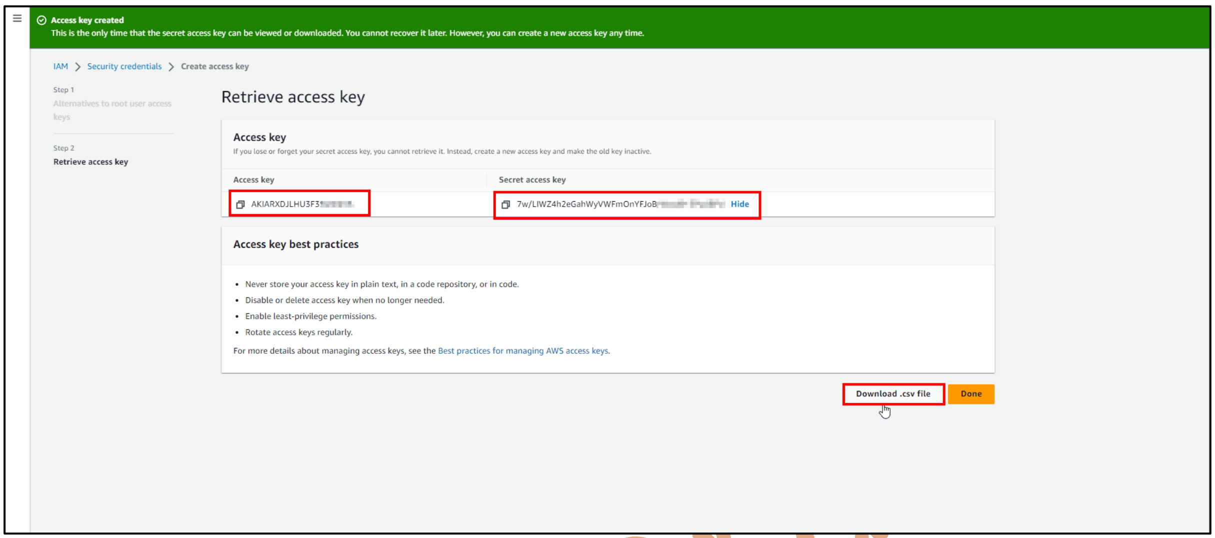
Access key ID	Created on	Access key last used	Region last used	Service last used	Status

No access keys

As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more

Module 8: Terraform Assignment

3) Click on **Show Access Key** and **save/download** the access key and secret access key.



Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

IAM > Security credentials > Create access key

Step 1
Alternatives to root user access keys

Step 2
Retrieve access key

Retrieve access key

Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIARXDJLHU3F3[redacted]	7w/LIWZ4h2eGahWyVWFmOnYFJoB[redacted] Hide

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the Best practices for managing AWS access keys.

Note :-Best Practices For Managing AWS Access Keys

Here are some of the best practices that you must follow while managing AWS access keys:

- **Never generate an account access key.:** One of the best ways to protect your account is to not create access keys for your AWS account root user unless required. Instead, the recommended best practice is to create one or more AWS Identity and Access Management (IAM) users and grant those IAM users the necessary permissions and use them for everyday interaction with AWS.
- **Use temporary security credentials instead of long-term access keys:** Long-term access keys that never expire are not often required. Instead, you can create IAM roles and generate temporary security credentials. Temporary security credentials consist of an access key ID and a secret access key, but they also include a security token that indicates when the credentials expire.
- **Manage IAM user access keys properly:** If you must create access keys for programmatic access to AWS, create them for IAM users, granting the users only the permissions they require.

Module 8: Terraform Assignment

- **Access keys should not be directly embedded in the code:** Put access keys in either the **AWS Credentials file** or **Environment Variables**.

Step 2 After perform this task install terraform in your local machine here we have using windows version of terraform. Please follow the documentation

https://spacelift.io/blog/how-to-install-terraform

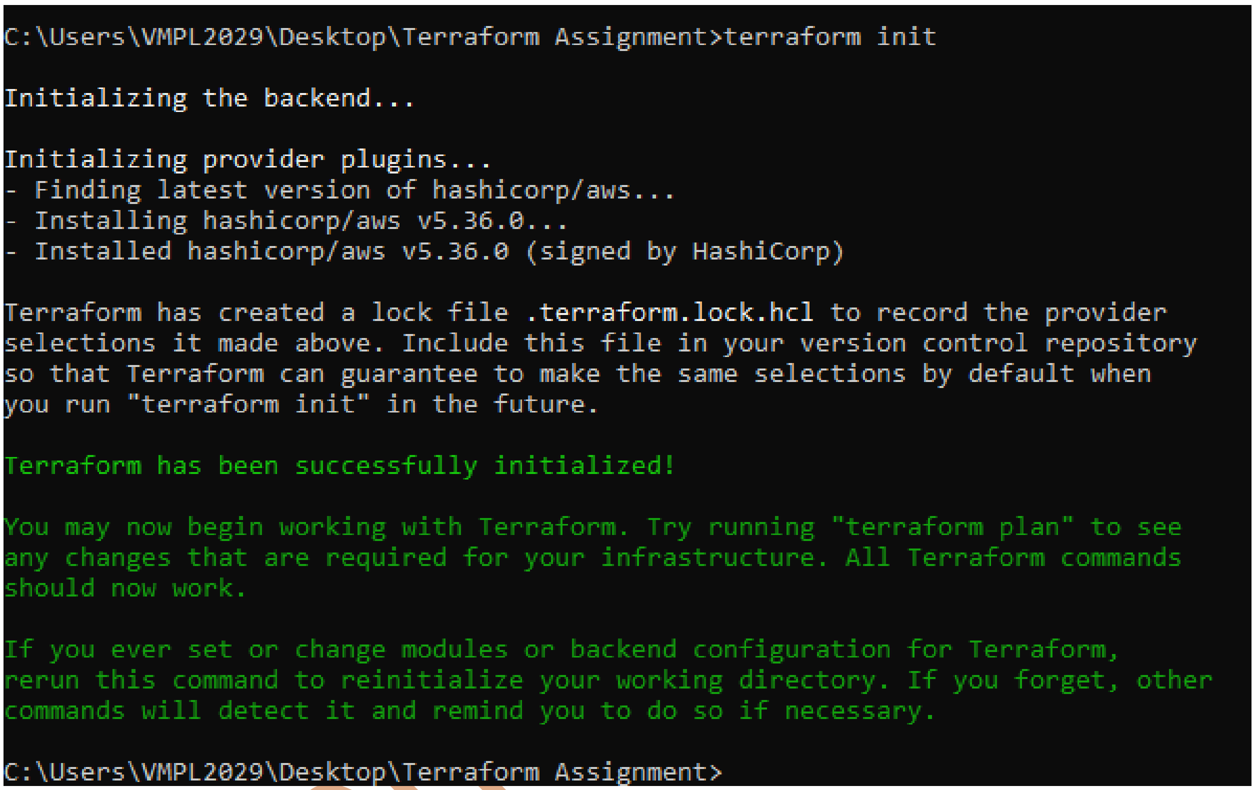
Now we need to our task and the task is Create an EC2 service in the default subnet in the Ohio region.

Before initialize the terraform we need the below configuration file here a sample code

```
## Providers (AWS/Azure/GCP)

# provider block

provider "aws" {
access_key = "access key"
secret_key = "secret key"
region = "ap-south-1"
}

# resource block

resource "aws_instance" "MyWebServer" {
# resource arguments
ami = "ami-0f5ee92e2d63afc18" # replace with ami-
id in your account
instance_type = "t2.micro"
}
```

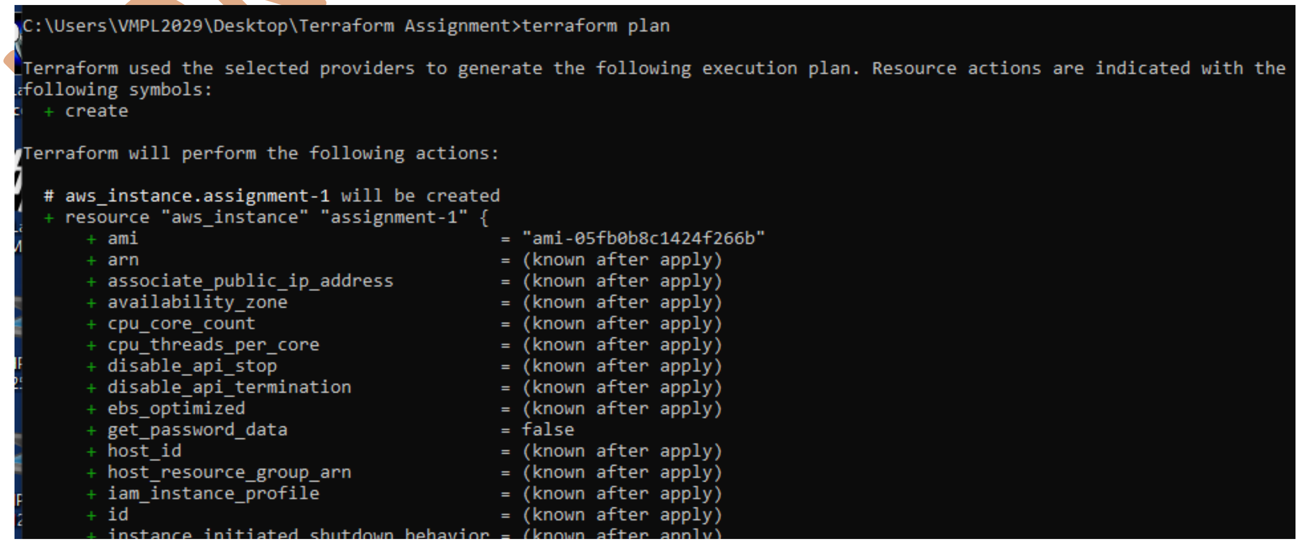
Module 8: Terraform Assignment

Where we have created configuration .tf file Initialize terraform



```
C:\Users\VMPL2029\Desktop\Terraform Assignment>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.36.0...
- Installed hashicorp/aws v5.36.0 (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.

C:\Users\VMPL2029\Desktop\Terraform Assignment>
```

After terraform init just check the what plan of the terraform using $ terraform plan



```
C:\Users\VMPL2029\Desktop\Terraform Assignment>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:

  # aws_instance.assignment-1 will be created
  + resource "aws_instance" "assignment-1" {
      + ami                                  = "ami-05fb0b8c1424f266b"
      + 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)
      + 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)
      + instance_initiated_shutdown_behavior = (known after apply)
```

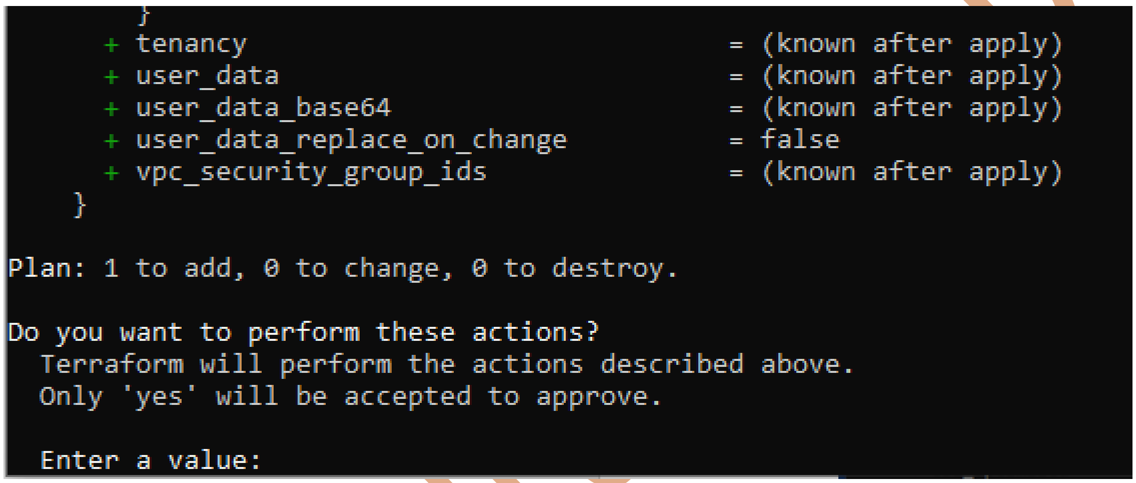
And here everything looking good so

Module 8: Terraform Assignment

Now apply your code.

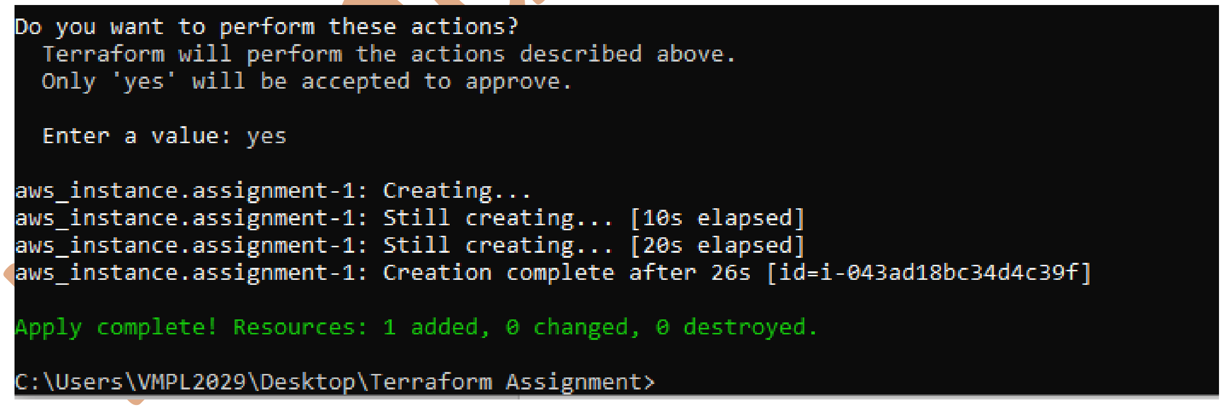
$ terraform apply

Here ask for the final permission just type yes and hit enter



```
          }
      + 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?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

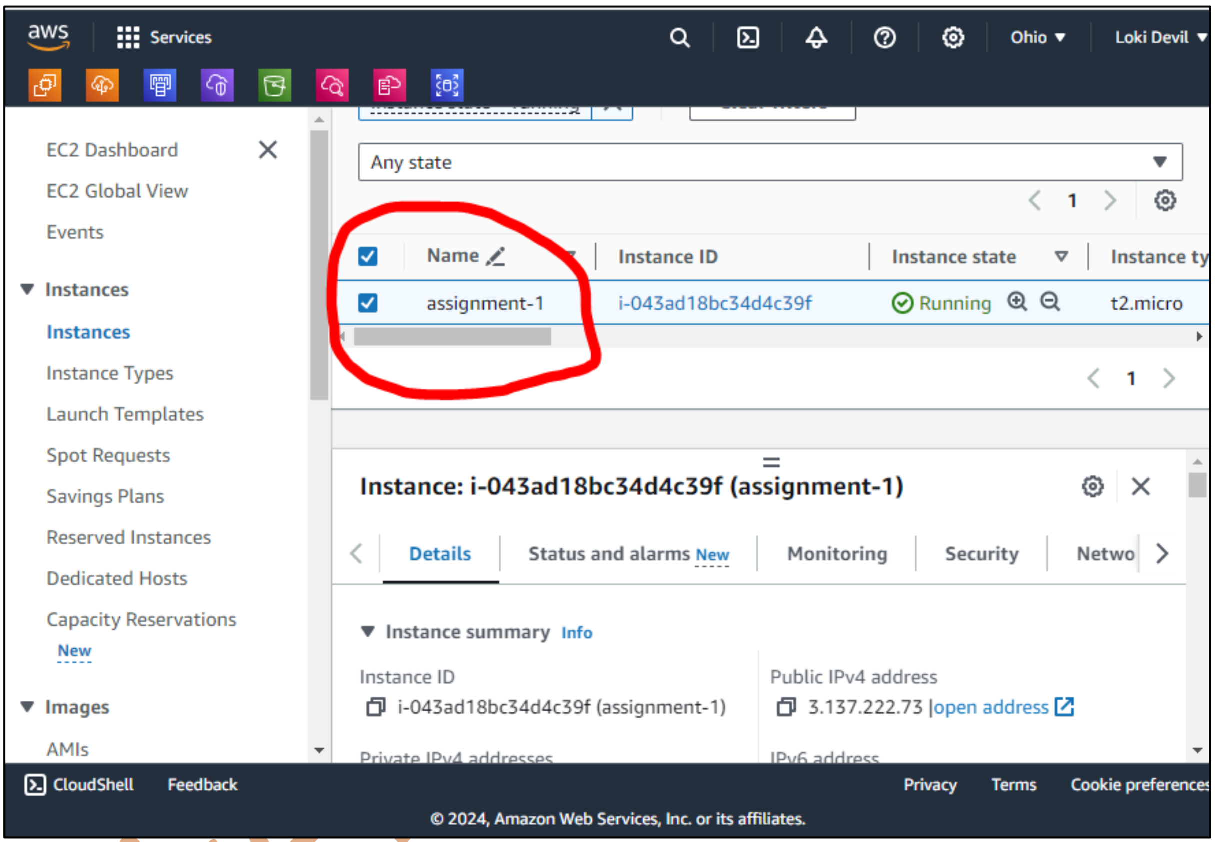
There we go we have complete our first assignment



```
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-1: Creating...
aws_instance.assignment-1: Still creating... [10s elapsed]
aws_instance.assignment-1: Still creating... [20s elapsed]
aws_instance.assignment-1: Creation complete after 26s [id=i-043ad18bc34d4c39f]

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

C:\Users\VMPL2029\Desktop\Terraform Assignment>
```

Module 8: Terraform Assignment

Lets verify in our AWS console



Any state

Name	Instance ID	Instance state	Instance ty
assignment-1	i-043ad18bc34d4c39f	Running	t2.micro

Instance: i-043ad18bc34d4c39f (assignment-1)

Details | Status and alarms New | Monitoring | Security | Netwo

Instance summary Info

Instance ID
i-043ad18bc34d4c39f (assignment-1)

Public IPv4 address
3.137.222.73 |open address

Private IPv4 addresses

IPv6 address

© 2024, Amazon Web Services, Inc. or its affiliates.

We have completed assignment 1 successfully.

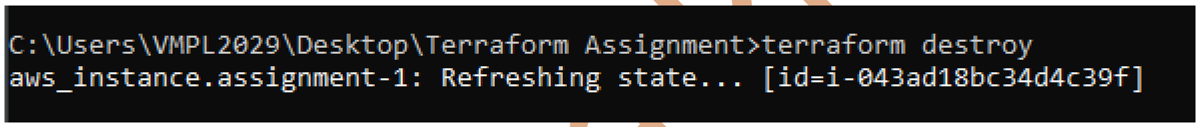
Module 8: Terraform Assignment

Terraform Assignment -2

Tasks To Be Performed:

1. **Destroy the previous deployment**

To destroy the previous deployment in Terraform, you can use the `terraform destroy` command. Navigate to the directory containing your Terraform configuration files and execute the following command:



```
C:\Users\VMPL2029\Desktop\Terraform Assignment>terraform destroy
aws_instance.assignment-1: Refreshing state... [id=i-043ad18bc34d4c39f]
```

2. **Create a new EC2 instance with an Elastic IP**

Step 1 :- To create a new EC2 instance with an Elastic IP using Terraform, you can use the following configuration:

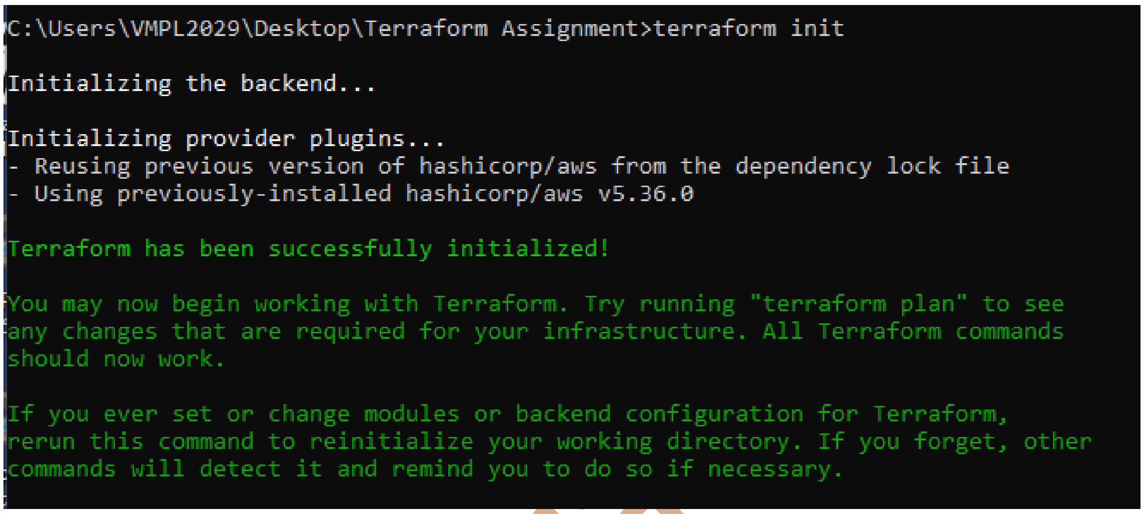
```
resource "aws_instance" "assignment-2" {
    ami = "ami-05fb0b8c1424f266b"
    instance_type = "t2.micro"
    key_name = "terraform-kp"
    tags = {
    Name = "assignment-2"
    }
}

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
```

Module 8: Terraform Assignment

```
}
```

Make sure to replace `"ami-12345678"` with your desired AMI ID and modify other parameters as per your requirements. This configuration will create an EC2 instance and associate an Elastic IP address with it.

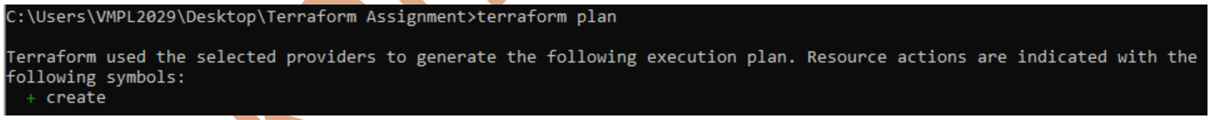
Step 2:- As the previous initialised the terraform using the command $ terraform init



```
C:\Users\VMPL2029\Desktop\Terraform Assignment>terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.36.0

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.
```

After applying this configuration, you can use the `public_ip` output to get the Elastic IP address assigned to the EC2 instance.

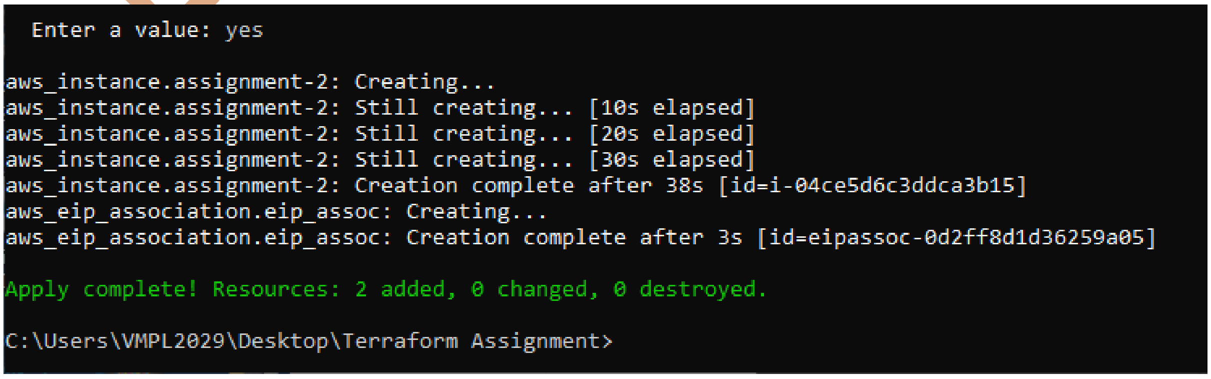
And then $ terraform plan



```
C:\Users\VMPL2029\Desktop\Terraform Assignment>terraform plan

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

Finally apply the configuration

Using $ terraform apply



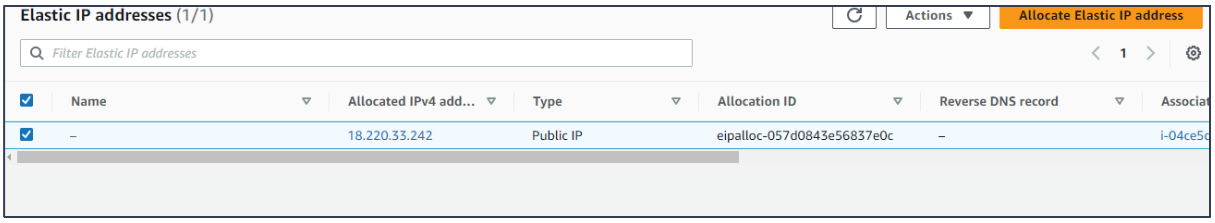
```
  Enter a value: yes

aws_instance.assignment-2: Creating...
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 38s [id=i-04ce5d6c3ddca3b15]
aws_eip_association.eip_assoc: Creating...
aws_eip_association.eip_assoc: Creation complete after 3s [id=eipassoc-0d2ff8d1d36259a05]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

C:\Users\VMPL2029\Desktop\Terraform Assignment>
```

Finnaly we have complte our 2nd assgnment.

Module 8: Terraform Assignment

Lets verify

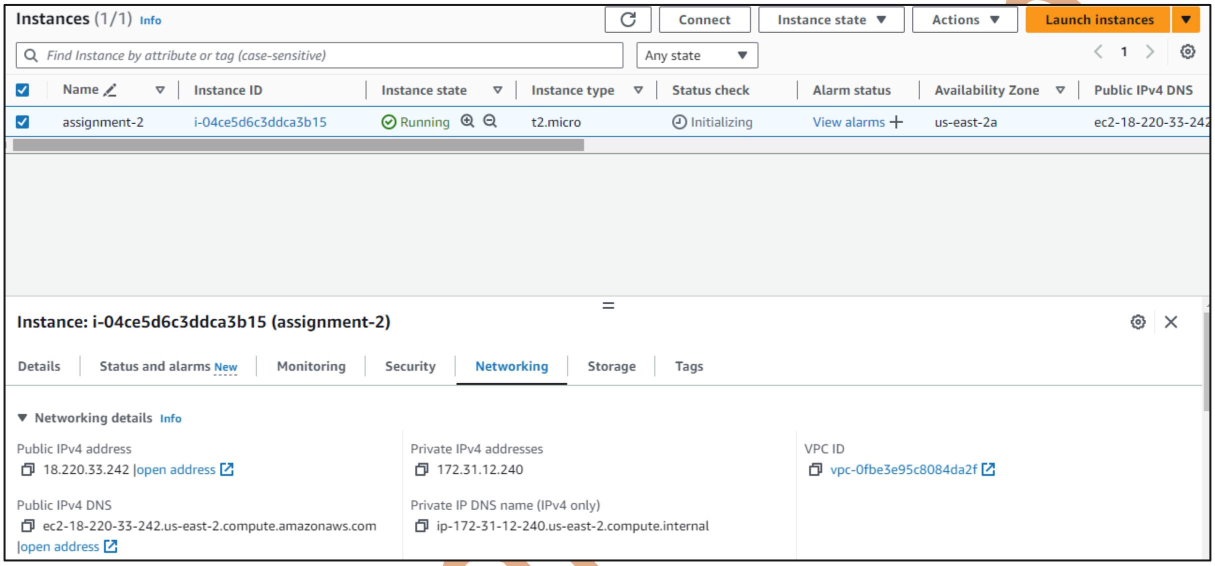
This is the EIP



Elastic IP addresses (1/1)

Name	Allocated IPv4 add...	Type	Allocation ID	Reverse DNS record	Associat
–	18.220.33.242	Public IP	eipalloc-057d0843e56837e0c	–	i-04ce5c

And this is the EC2 associate with the EIP



Instances (1/1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
assignment-2	i-04ce5d6c3ddca3b15	Running	t2.micro	Initializing	View alarms	us-east-2a	ec2-18-220-33-242

Instance: i-04ce5d6c3ddca3b15 (assignment-2)

Networking details

Public IPv4 address: 18.220.33.242 |open address

Private IPv4 addresses: 172.31.12.240

VPC ID: vpc-0fbe3e95c8084da2f

Public IPv4 DNS: ec2-18-220-33-242.us-east-2.compute.amazonaws.com |open address

Private IP DNS name (IPv4 only): ip-172-31-12-240.us-east-2.compute.internal

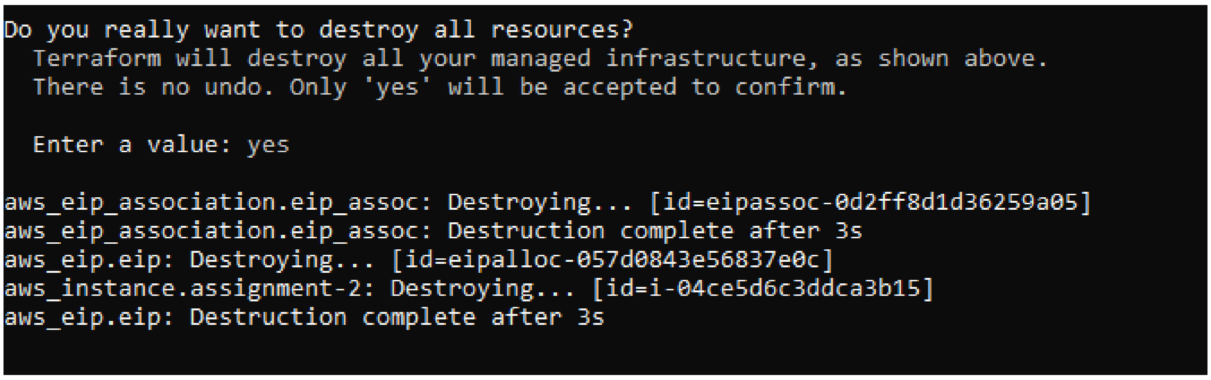
Terraform Assignment -3

Tasks To Be Performed:

1. Destroy the previous deployment

To destroy the previous deployment in Terraform, you can use the **`terraform destroy`** command. Navigate to the directory containing your Terraform configuration files and execute the following command:

Module 8: Terraform Assignment



```
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

aws_eip_association.eip_assoc: Destroying... [id=eipassoc-0d2ff8d1d36259a05]
aws_eip_association.eip_assoc: Destruction complete after 3s
aws_eip.eip: Destroying... [id=eipalloc-057d0843e56837e0c]
aws_instance.assignment-2: Destroying... [id=i-04ce5d6c3ddca3b15]
aws_eip.eip: Destruction complete after 3s
```

2. Create 2 EC2 instances in Ohio and N.Virginia respectively
3. 3. Rename Ohio's instance to 'hello-ohio' and Virginia's instance to 'hello-virginia

Here we taking N.Virginia EC2 name ansible and Ohio named as

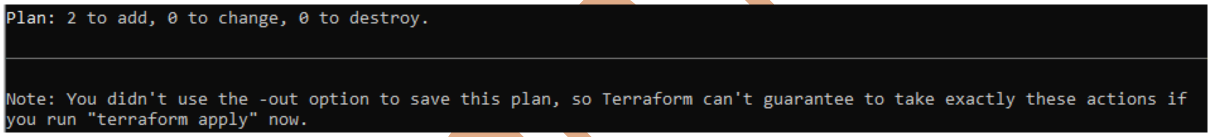
Step 1:-Below is a Terraform configuration to create two EC2 instances, each in a different region (Ohio and N. Virginia):

Step 2:- To rename the instances to 'hello-ohio' and 'hello-virginia' respectively, you can use the `tags` parameter in Terraform. Below is the updated Terraform configuration:

In this configuration, I've added tags with the key Name for each instance resource. This will rename the instances accordingly. Ensure you replace **"ami-12345678"** with your desired AMI ID for each region and modify other parameters as needed.

```
provider "aws" {
    alias = "Ohio"
    region = "us-east-2"
    access_key = "AKIA5XB3OVIEMFVRVNJ6"
    secret_key =
"eMFMn/3t8ipDKwWV0WEKLXOWa9Q8E2dBDz5dqHfr"
}
resource "aws_instance" "assignment-3-1" {
    provider = aws.NV
    ami = "ami-0c7217cdde317cfec"
    instance_type = "t2.micro"
    key_name = "ansiblekp"
    tags = {
    Name = "hello-virginia"
    }
}
resource "aws_instance" "assignment-3-2" {
    provider = aws.Ohio
    ami = "ami-05fb0b8c1424f266b"
    instance_type = "t2.micro"
    key_name = "terraform-kp"
    tags = {
    Name = "hello-ohio"
    }
}
```

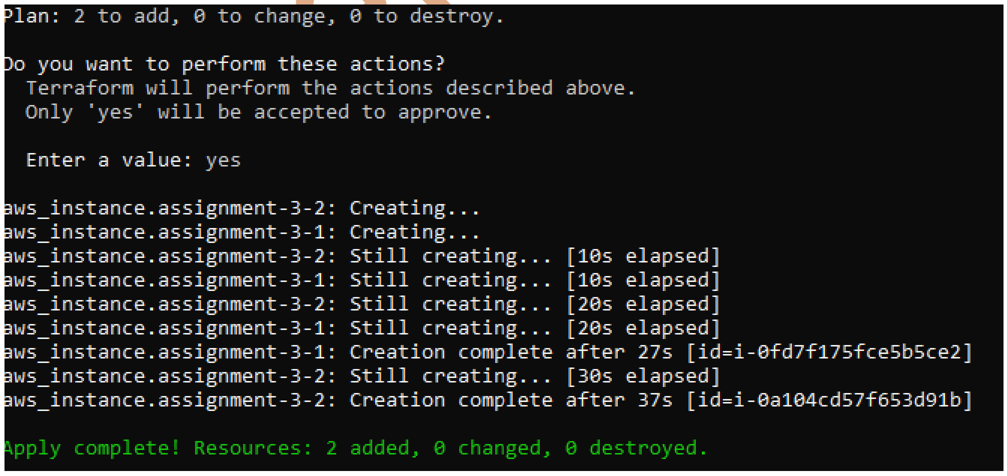
And first run $ terraform plan



```
Plan: 2 to add, 0 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.
```

And here everything looks good

Now apply these configuration using the command $ terraform apply

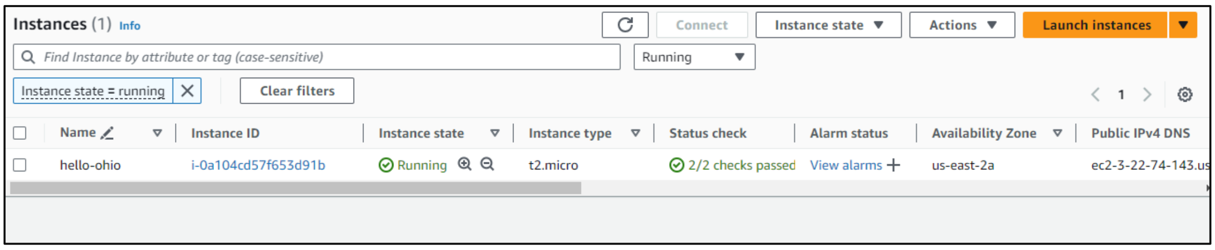


```
Plan: 2 to add, 0 to change, 0 to destroy.

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

aws_instance.assignment-3-2: Creating...
aws_instance.assignment-3-1: Creating...
aws_instance.assignment-3-2: Still creating... [10s elapsed]
aws_instance.assignment-3-1: Still creating... [10s elapsed]
aws_instance.assignment-3-2: Still creating... [20s elapsed]
aws_instance.assignment-3-1: Still creating... [20s elapsed]
aws_instance.assignment-3-1: Creation complete after 27s [id=i-0fd7f175fce5b5ce2]
aws_instance.assignment-3-2: Still creating... [30s elapsed]
aws_instance.assignment-3-2: Creation complete after 37s [id=i-0a104cd57f653d91b]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Lets verify the result that we have applied

Here in the ohio region 1 EC2

Module 8: Terraform Assignment



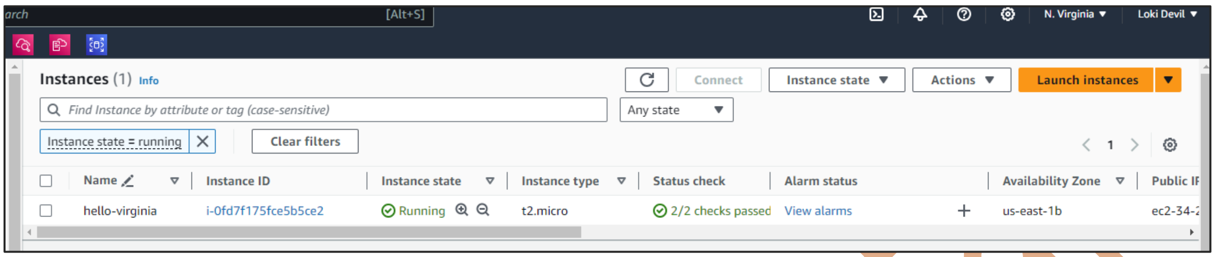
Instances (1) Info

Find Instance by attribute or tag (case-sensitive) | Running

Instance state = running | Clear filters

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
	hello-ohio	i-0a104cd57f653d91b	Running	t2.micro	2/2 checks passed	View alarms +	us-east-2a	ec2-3-22-74-143.us

And in the N. Virginia Region



Instances (1) Info

Find Instance by attribute or tag (case-sensitive) | Any state

Instance state = running | Clear filters

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IF
	hello-virginia	i-0fd7f175fce5b5ce2	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-34-2

Now we have successfully completed our assignment 3.

Terraform Assignment -4

Tasks To Be Performed:

1. Destroy the previous deployments

2. Create a VPC with the required components using Terraform

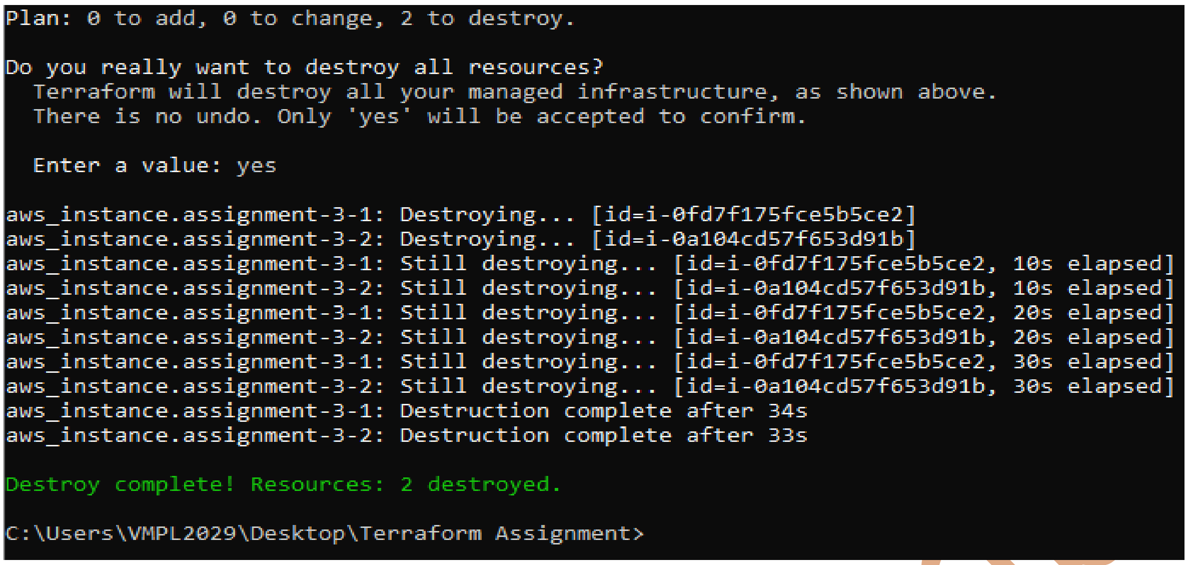
3. Deploy an EC2 instance inside the VPC

Steps:- To accomplish the tasks you've outlined, we'll break it down into three parts:

Task 1: Destroy Previous Deployments

To destroy previous deployments, you can run the **$ terraform destroy** command in the directory where your Terraform configuration files are located.

Module 8: Terraform Assignment



```
Plan: 0 to add, 0 to change, 2 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

aws_instance.assignment-3-1: Destroying... [id=i-0fd7f175fce5b5ce2]
aws_instance.assignment-3-2: Destroying... [id=i-0a104cd57f653d91b]
aws_instance.assignment-3-1: Still destroying... [id=i-0fd7f175fce5b5ce2, 10s elapsed]
aws_instance.assignment-3-2: Still destroying... [id=i-0a104cd57f653d91b, 10s elapsed]
aws_instance.assignment-3-1: Still destroying... [id=i-0fd7f175fce5b5ce2, 20s elapsed]
aws_instance.assignment-3-2: Still destroying... [id=i-0a104cd57f653d91b, 20s elapsed]
aws_instance.assignment-3-1: Still destroying... [id=i-0fd7f175fce5b5ce2, 30s elapsed]
aws_instance.assignment-3-2: Still destroying... [id=i-0a104cd57f653d91b, 30s elapsed]
aws_instance.assignment-3-1: Destruction complete after 34s
aws_instance.assignment-3-2: Destruction complete after 33s

Destroy complete! Resources: 2 destroyed.

C:\Users\VMPL2029\Desktop\Terraform Assignment>
```

Task 2: Create a VPC with Required Components

Below is a Terraform configuration to create a VPC with required components:

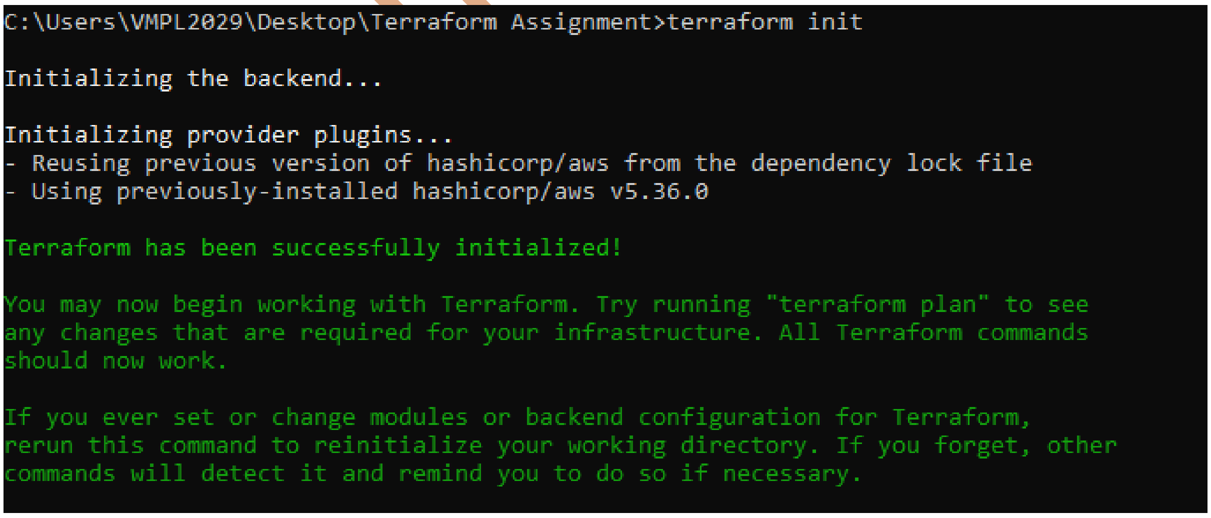
```
resource "aws_instance" "assignment-4" {
    ami = "ami-05fb0b8c1424f266b"
    instance_type = "t2.micro"
    subnet_id = aws_subnet.assignment-4-subnet.id
    key_name = "terraform-kp"
    tags = {
    Name = "assignment-4"
    }
}

resource "aws_vpc" "assignment-4-vpc" {
    cidr_block = "10.0.0.0/16"
    tags = {
    Name = "assignment-4-vpc"
    }
}
```

Task 3: Deploy an EC2 Instance Inside the VPC

Below is a Terraform configuration to deploy an EC2 instance inside the VPC:

```
 resource "aws_subnet" "assignment-4-subnet" {
    vpc_id = aws_vpc.assignment-4-vpc.id
    cidr_block = "10.0.1.0/24"
    availability_zone = "us-east-2a"
    tags = {
    Name = "assignment-4-subnet"
    }
}
```

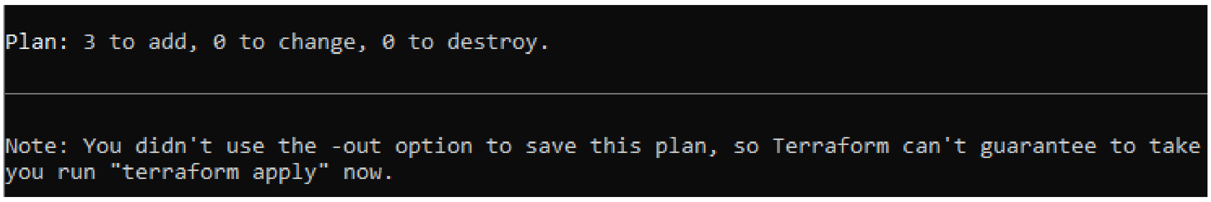
After creating these Terraform configurations, you can initialize Terraform in your directory (`terraform init`), apply the changes (`terraform apply`), and then destroy them when needed (`terraform destroy`). Make sure to review and adapt the configurations to your specific requirements before applying them.



```
C:\Users\VMPL2029\Desktop\Terraform Assignment>terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.36.0

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.
```

After that check the plan of this configuration

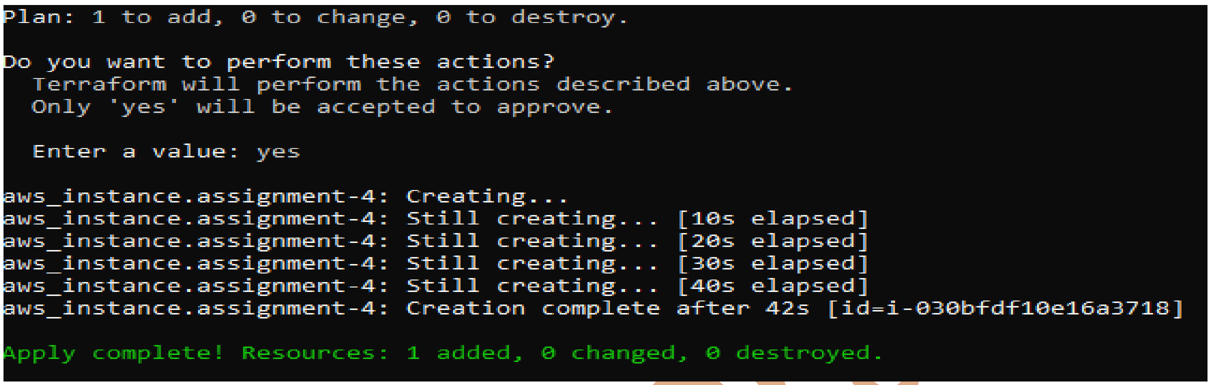
Module 8: Terraform Assignment



```
Plan: 3 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take
you run "terraform apply" now.
```

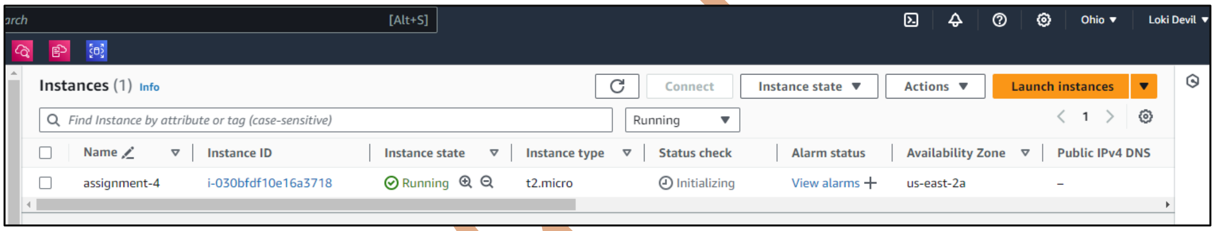
Looks everything good

Apply using the command $ terraform 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

aws_instance.assignment-4: Creating...
aws_instance.assignment-4: Still creating... [10s elapsed]
aws_instance.assignment-4: Still creating... [20s elapsed]
aws_instance.assignment-4: Still creating... [30s elapsed]
aws_instance.assignment-4: Still creating... [40s elapsed]
aws_instance.assignment-4: Creation complete after 42s [id=i-030bfdf10e16a3718]

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

Lets verify



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
	assignment-4	i-030bfdf10e16a3718	Running	t2.micro	Initializing	View alarms +	us-east-2a	–

Now we have performed assignment 4 successfully.

Terraform Assignment – 5

You have been asked to:

1. Destroy the previous deployments
2. Create a script to install apache2
3. Run this script on a newly created EC2 instance
4. Print the IP address of the instance in a file on the local, once deployed

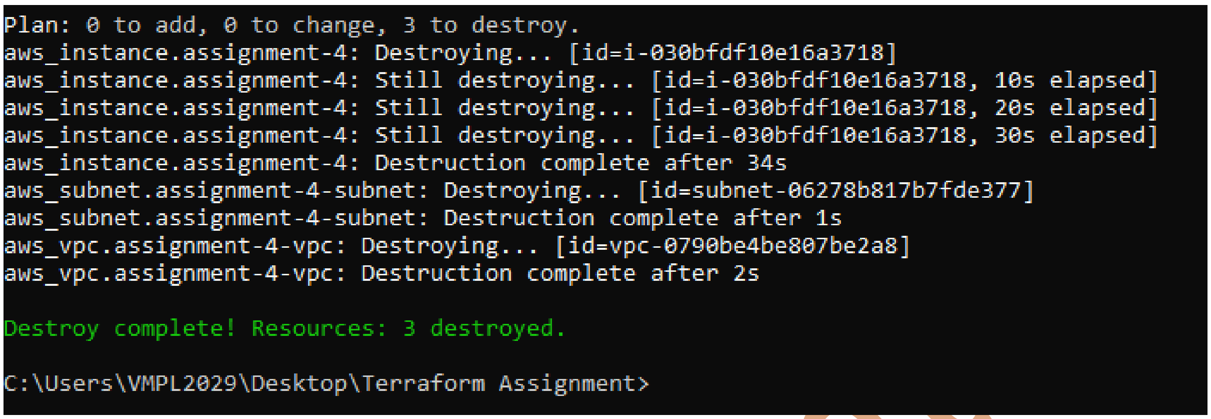
To full fill the tasks you've outlined, we'll break it down into steps:

Step 1: Destroy Previous Deployments

Module 8: Terraform Assignment

Ensure you are in the directory containing your Terraform configuration files and execute the following command to destroy previous deployments:

$ terraform destroy -auto-approve



```
Plan: 0 to add, 0 to change, 3 to destroy.
aws_instance.assignment-4: Destroying... [id=i-030bfdf10e16a3718]
aws_instance.assignment-4: Still destroying... [id=i-030bfdf10e16a3718, 10s elapsed]
aws_instance.assignment-4: Still destroying... [id=i-030bfdf10e16a3718, 20s elapsed]
aws_instance.assignment-4: Still destroying... [id=i-030bfdf10e16a3718, 30s elapsed]
aws_instance.assignment-4: Destruction complete after 34s
aws_subnet.assignment-4-subnet: Destroying... [id=subnet-06278b817b7fde377]
aws_subnet.assignment-4-subnet: Destruction complete after 1s
aws_vpc.assignment-4-vpc: Destroying... [id=vpc-0790be4be807be2a8]
aws_vpc.assignment-4-vpc: Destruction complete after 2s

Destroy complete! Resources: 3 destroyed.

C:\Users\VMPL2029\Desktop\Terraform Assignment>
```

Step 2: Create a Script to Install Apache2

Create a bash script named install_apache.sh with the following content:

```
#!/bin/bash
sudo apt update -y
sudo apt install apache2 -y
sudo su
echo "assignmnet 5 is done" > /var/www/html/index.html
```

Step 3: Run the Script on a Newly Created EC2 Instance

Modify your Terraform configuration to include user data that runs the script on the EC2 instance:

Step 4: Print the IP Address of the Instance to a File Locally

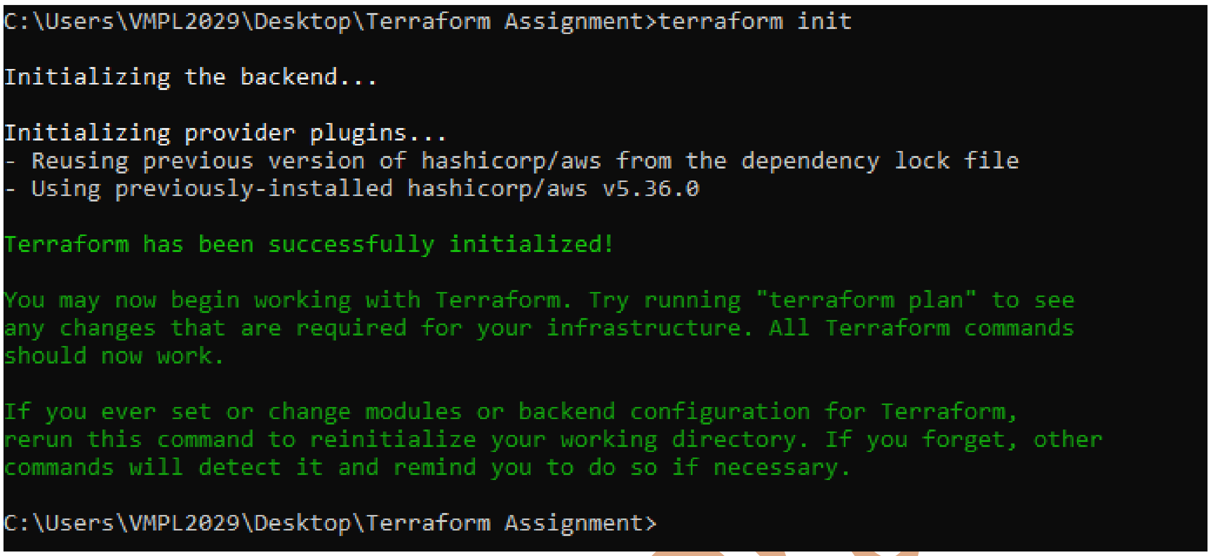
After applying the Terraform configuration, you can use Terraform output to print the IP address of the instance to a file locally. Add an output block to your Terraform configuration:

```
output "IPv4" {
    value = aws_instance.a5-instance.public_ip
```

Module 8: Terraform Assignment

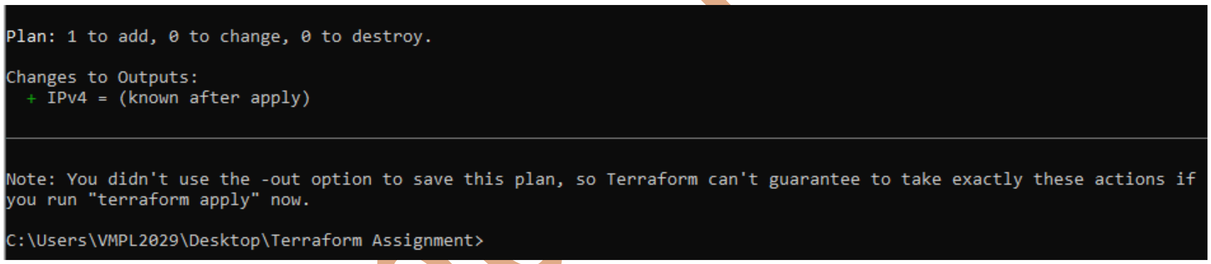
```
}
```

And initialize the configuration using $ terraform init command



```
C:\Users\VMPL2029\Desktop\Terraform Assignment>terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.36.0

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.

C:\Users\VMPL2029\Desktop\Terraform Assignment>
```

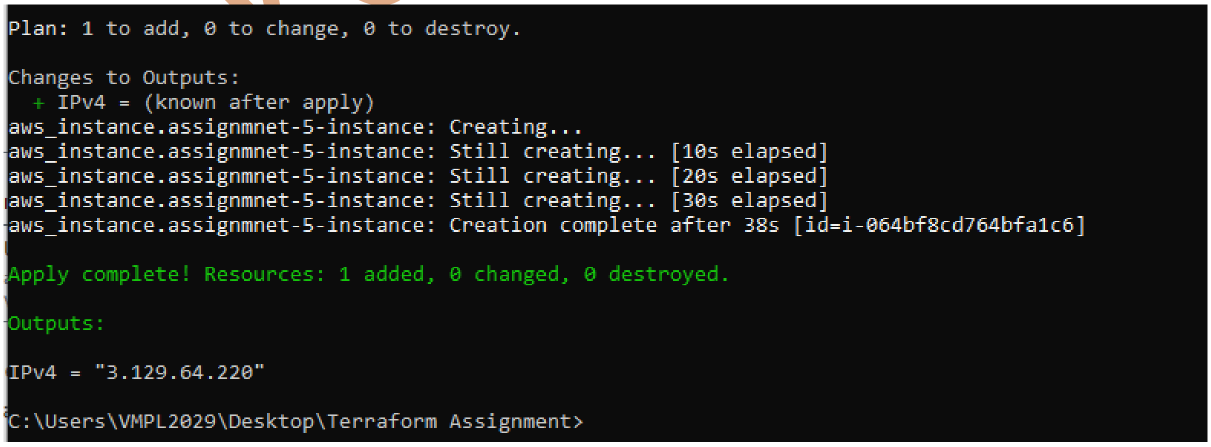
Check terraform plan



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

Changes to Outputs:
  + IPv4 = (known after apply)

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.

C:\Users\VMPL2029\Desktop\Terraform Assignment>
```

Everything looks good in the our configuration lets apply it.

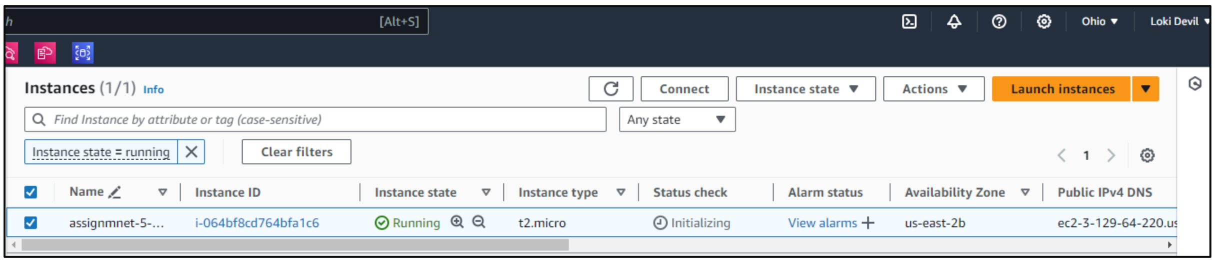
Using command $ terraform apply -auto-approve



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

Changes to Outputs:
  + IPv4 = (known after apply)
aws_instance.assignmnet-5-instance: Creating...
aws_instance.assignmnet-5-instance: Still creating... [10s elapsed]
aws_instance.assignmnet-5-instance: Still creating... [20s elapsed]
aws_instance.assignmnet-5-instance: Still creating... [30s elapsed]
aws_instance.assignmnet-5-instance: Creation complete after 38s [id=i-064bf8cd764bfa1c6]

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

Outputs:

IPv4 = "3.129.64.220"

C:\Users\VMPL2029\Desktop\Terraform Assignment>
```

Lets verify it

Module 8: Terraform Assignment

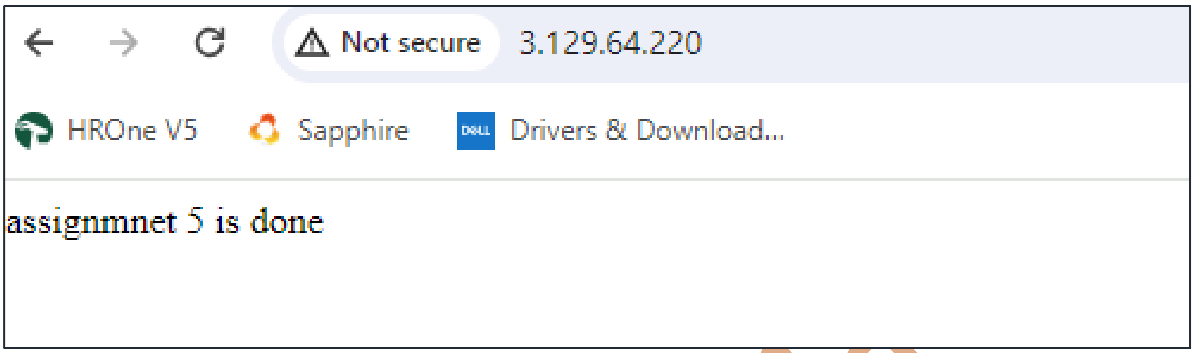


Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
assignmnet-5-...	i-064bf8cd764bfa1c6	Running	t2.micro	Initializing	View alarms +	us-east-2b	ec2-3-129-64-220.us



assignmnet 5 is done

Now we have successfully performed our assignment 5

Summary:

Destroy the previous deployments with terraform destroy -auto-approve.

Create the install_apache.sh script to install Apache2.

Modify your Terraform configuration to include user data to run the script on the EC2 instance.

Add an output block to print the IP address of the instance locally.

Run terraform apply to create the instance.

Capture the IP address with terraform output instance_ip > instance_ip.txt.