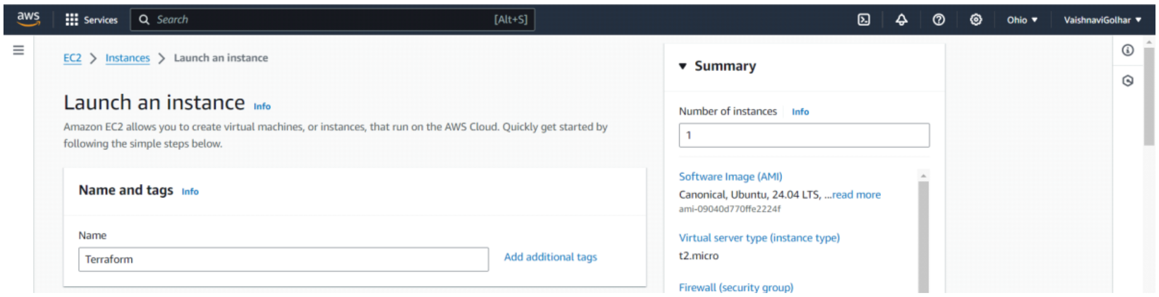
# Module 8: Terraform Assignment-1

## Tasks To Be Performed:

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

## Solution:

## Step 1: Install Terraform on AWS ec2 instance



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: Terraform

Add additional tags

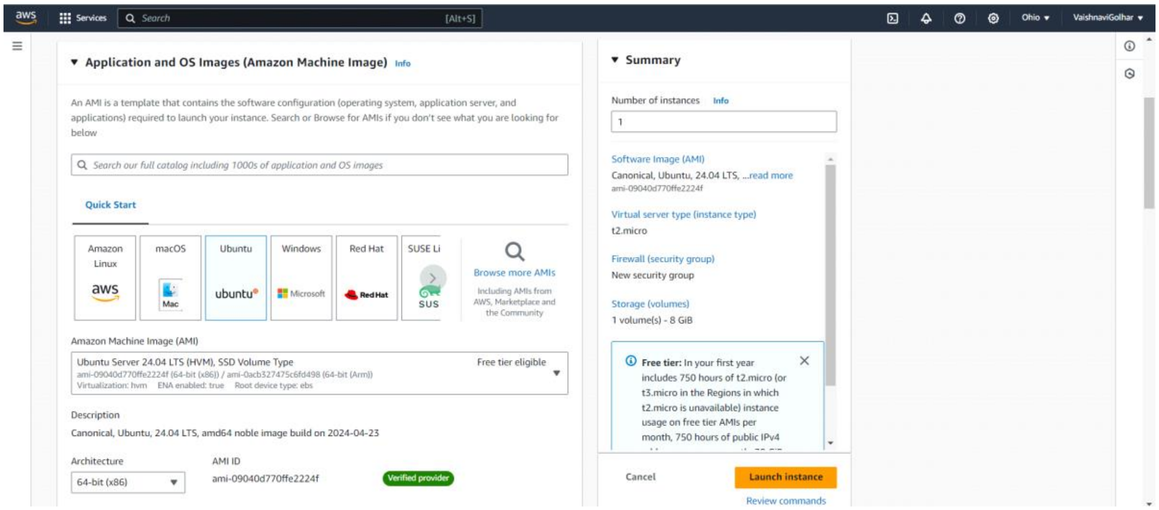
▾ Summary

Number of instances Info: 1

Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...read more
ami-09040d770ffe2224f

Virtual server type (instance type)
t2.micro

Firewall (security group)



▾ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)
Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-09040d770ffe2224f (64-bit (x86)) / ami-0acb327475c6fd498 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Canonical, Ubuntu, 24.04 LTS, amd64 noble image build on 2024-04-23

Architecture: 64-bit (x86)

AMI ID: ami-09040d770ffe2224f — Verified provider

▾ Summary

Number of instances Info: 1

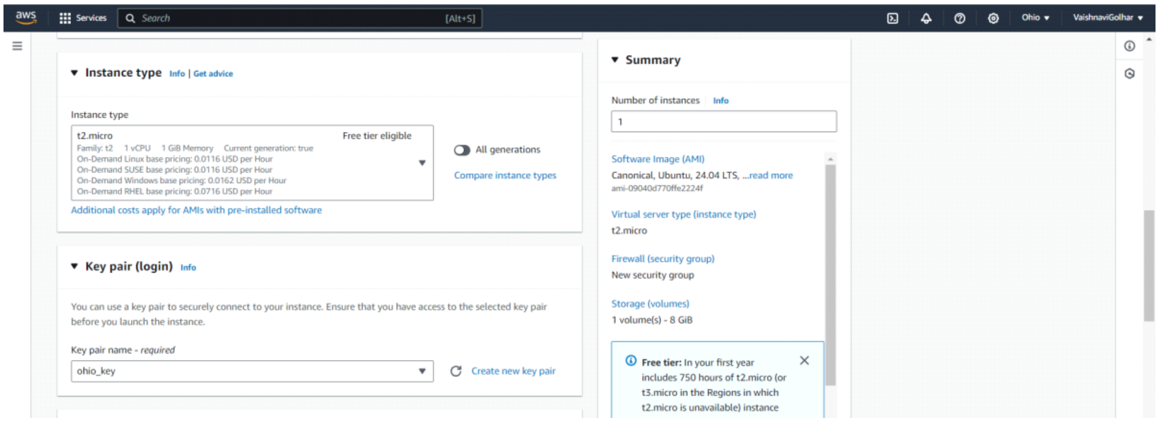
Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...read more
ami-09040d770ffe2224f

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4



▾ **Instance type** Info | Get advice

Instance type

t2.micro Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0116 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand RHEL base pricing: 0.0716 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▾ **Key pair (login)** Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

ohio_key ▾

Create new key pair

▾ **Summary**

Number of instances Info

1

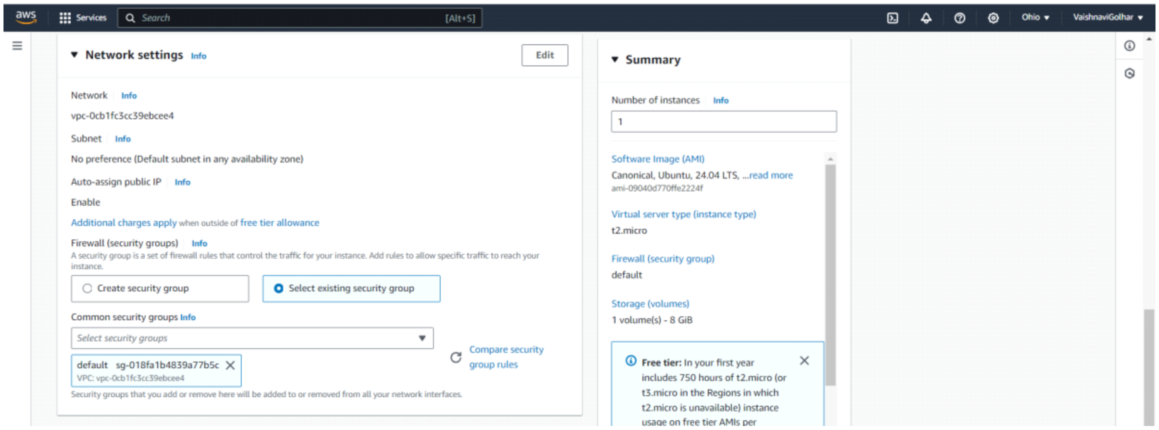
Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...read more
ami-09040d770ffe2224f

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

**Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance



▾ **Network settings** Info

Network Info
vpc-0cb1fc3cc39ebcee4

Subnet Info
No preference (Default subnet in any availability zone)

Auto-assign public IP Info
Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

○ Create security group ● Select existing security group

Common security groups Info

*Select security groups* ▾

default sg-018fa1b4839a77b5c
VPC: vpc-0cb1fc3cc39ebcee4

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

▾ **Summary**

Number of instances Info

1

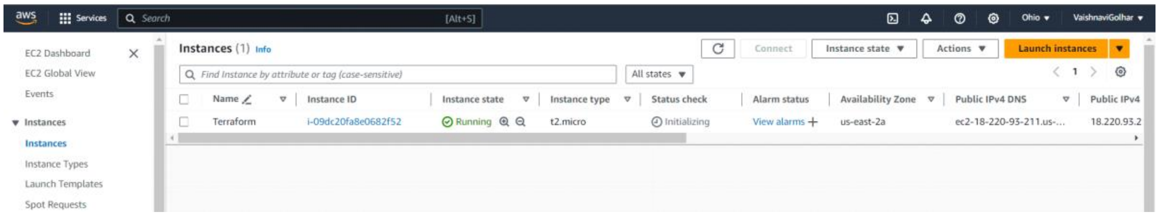
Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...read more
ami-09040d770ffe2224f

Virtual server type (instance type)
t2.micro

Firewall (security group)
default

Storage (volumes)
1 volume(s) - 8 GiB

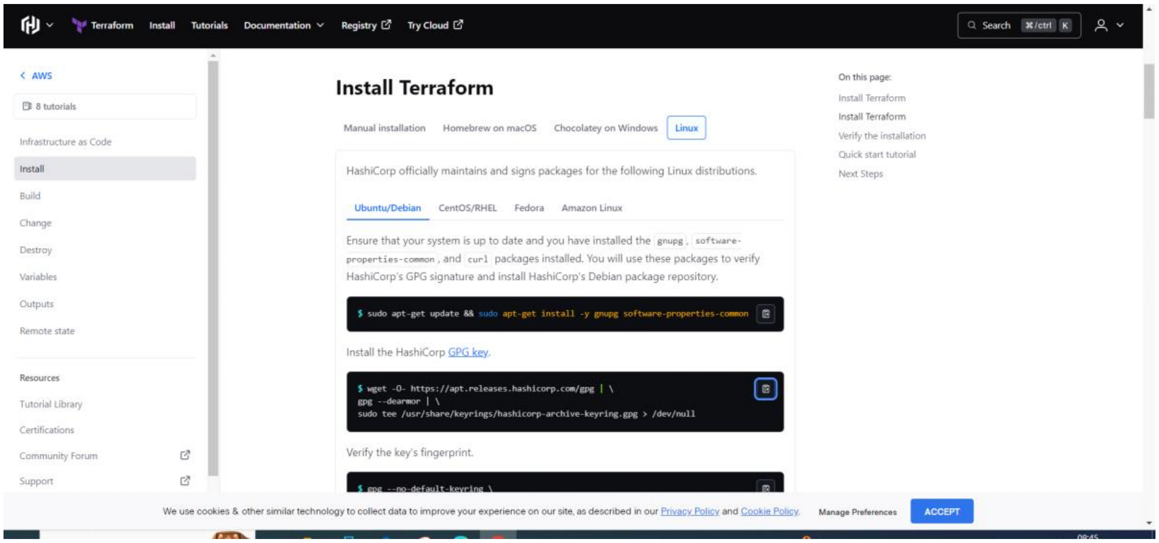
**Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per



**Instances** (1) Info

*Find Instance by attribute or tag (case-sensitive)*

| | Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS | Public IPv4 |
|---|---|---|---|---|---|---|---|---|---|
| □ | Terraform | i-09dc20fa8e0682f52 | Running | t2.micro | Initializing | View alarms + | us-east-2a | ec2-18-220-93-211.us-... | 18.220.93.2 |



## Install Terraform

Manual installation | Homebrew on macOS | Chocolatey on Windows | Linux

HashiCorp officially maintains and signs packages for the following Linux distributions.

Ubuntu/Debian | CentOS/RHEL | Fedora | Amazon Linux

Ensure that your system is up to date and you have installed the `gnupg`, `software-properties-common`, and `curl` packages installed. You will use these packages to verify HashiCorp's GPG signature and install HashiCorp's Debian package repository.

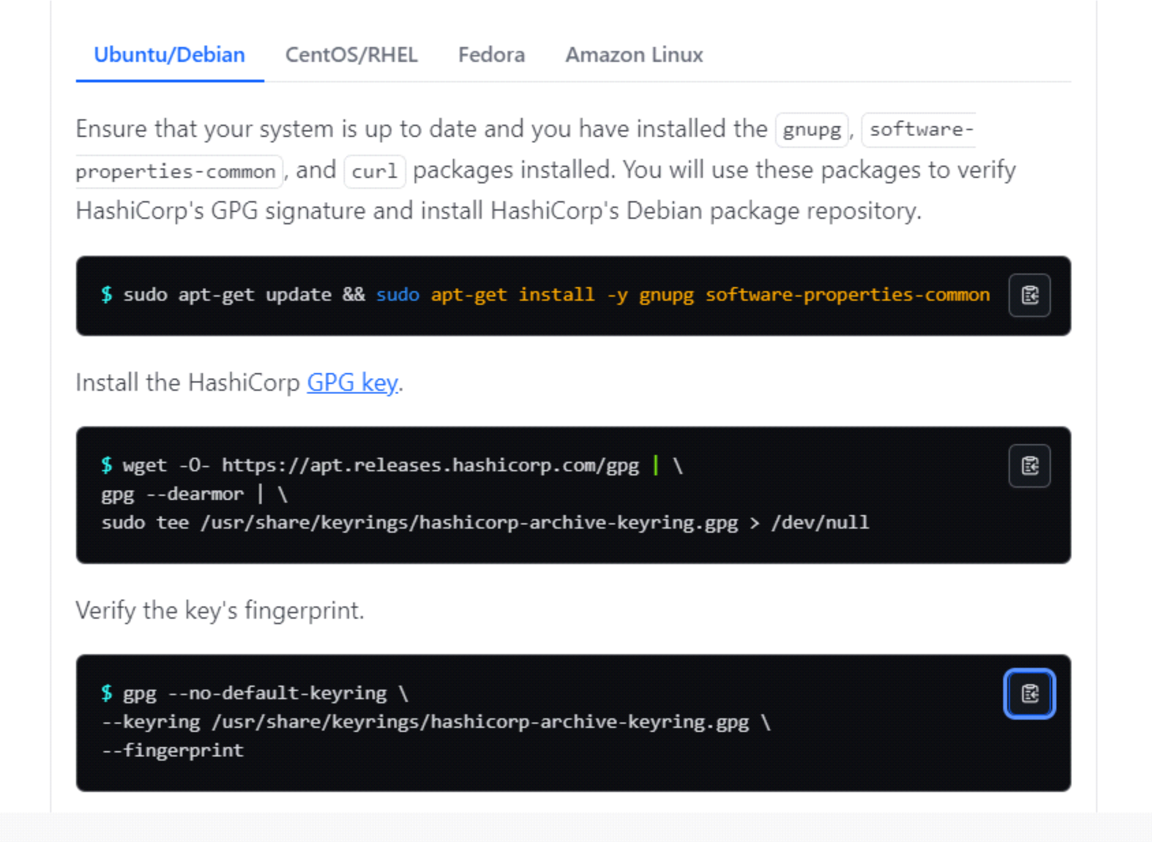
```
$ sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
```

Install the HashiCorp GPG key.

```
$ wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null
```

Verify the key's fingerprint.

We use cookies & other similar technology to collect data to improve your experience on our site, as described in our Privacy Policy and Cookie Policy.



Ubuntu/Debian | CentOS/RHEL | Fedora | Amazon Linux

Ensure that your system is up to date and you have installed the `gnupg`, `software-properties-common`, and `curl` packages installed. You will use these packages to verify HashiCorp's GPG signature and install HashiCorp's Debian package repository.

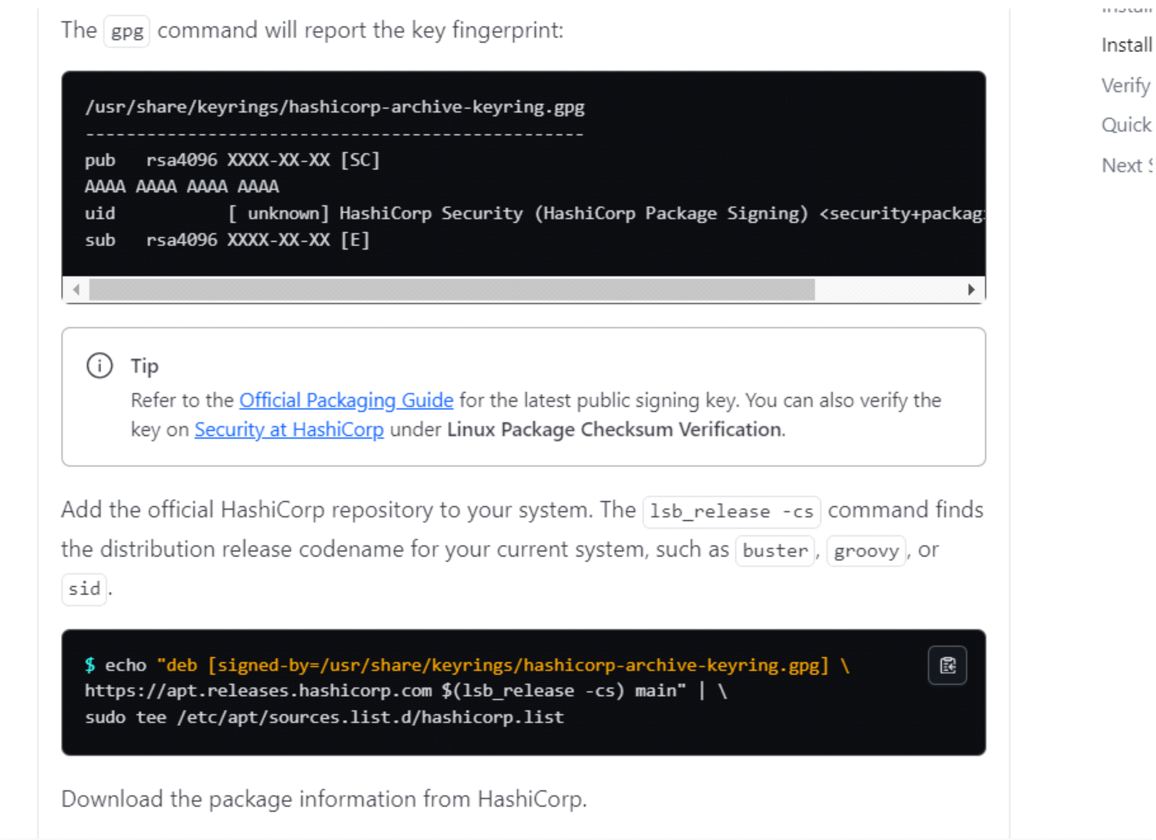
```
$ sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
```

Install the HashiCorp GPG key.

```
$ wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null
```

Verify the key's fingerprint.

```
$ gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
```



The `gpg` command will report the key fingerprint:

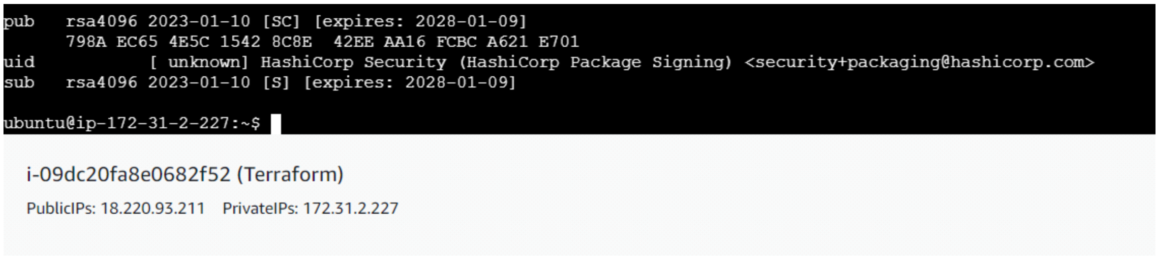
```
/usr/share/keyrings/hashicorp-archive-keyring.gpg
-------------------------------------------------
pub   rsa4096 XXXX-XX-XX [SC]
AAAA AAAA AAAA AAAA
uid           [ unknown] HashiCorp Security (HashiCorp Package Signing) <security+packag
sub   rsa4096 XXXX-XX-XX [E]
```

Tip
Refer to the Official Packaging Guide for the latest public signing key. You can also verify the key on Security at HashiCorp under **Linux Package Checksum Verification**.

Add the official HashiCorp repository to your system. The `lsb_release -cs` command finds the distribution release codename for your current system, such as `buster`, `groovy`, or `sid`.

```
$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
```

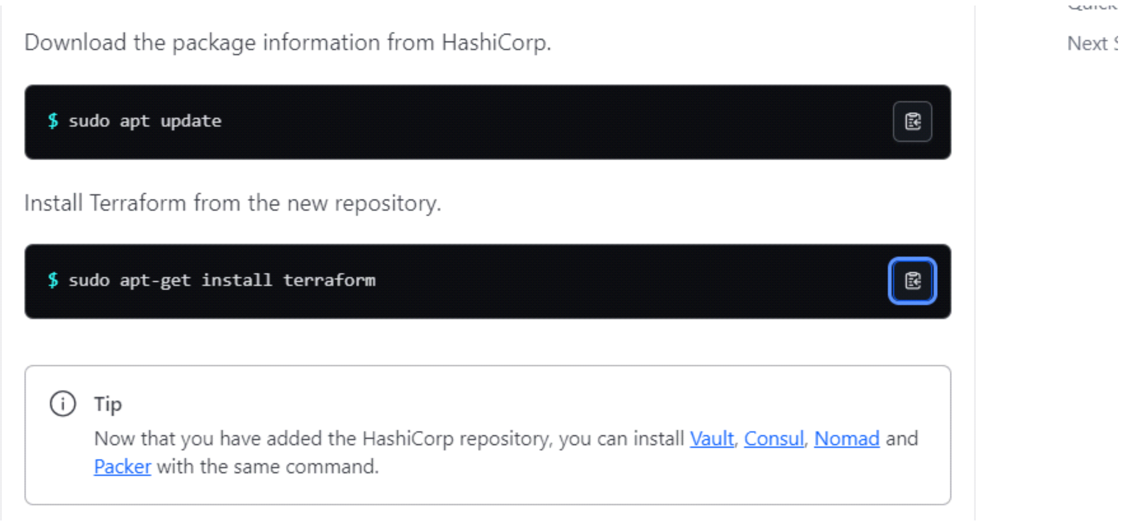
Download the package information from HashiCorp.



```
pub   rsa4096 2023-01-10 [SC] [expires: 2028-01-09]
      798A EC65 4E5C 1542 8C8E  42EE AA16 FCBC A621 E701
uid           [ unknown] HashiCorp Security (HashiCorp Package Signing) <security+packaging@hashicorp.com>
sub   rsa4096 2023-01-10 [S] [expires: 2028-01-09]

ubuntu@ip-172-31-2-227:~$
```

i-09dc20fa8e0682f52 (Terraform)

PublicIPs: 18.220.93.211   PrivateIPs: 172.31.2.227



Download the package information from HashiCorp.

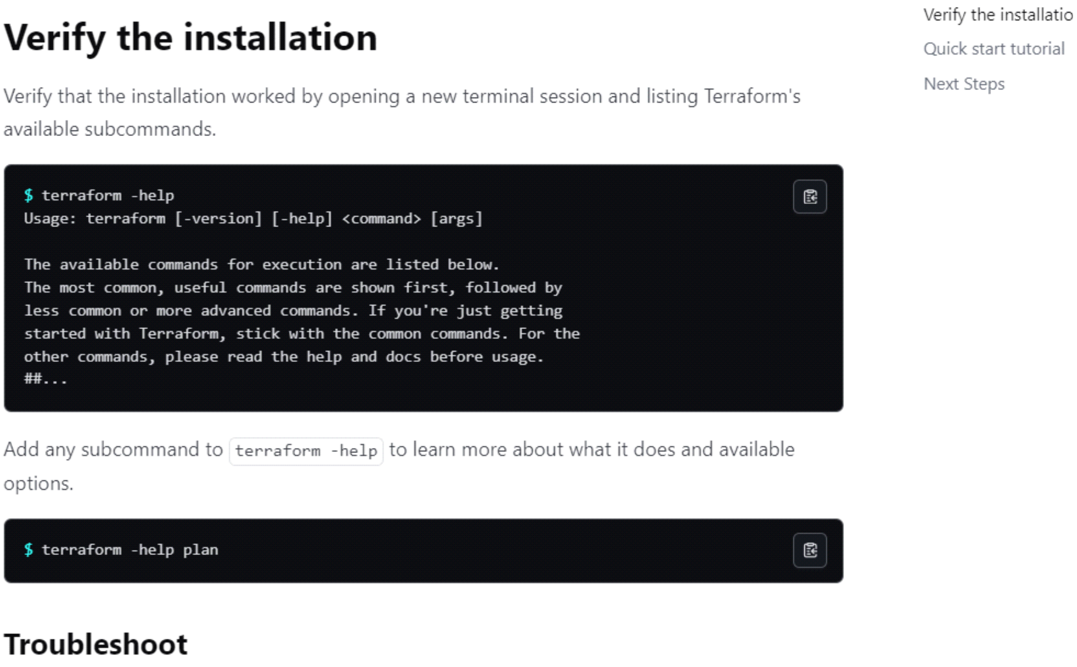
```
$ sudo apt update
```

Install Terraform from the new repository.

```
$ sudo apt-get install terraform
```

Tip

Now that you have added the HashiCorp repository, you can install Vault, Consul, Nomad and Packer with the same command.



## Verify the installation

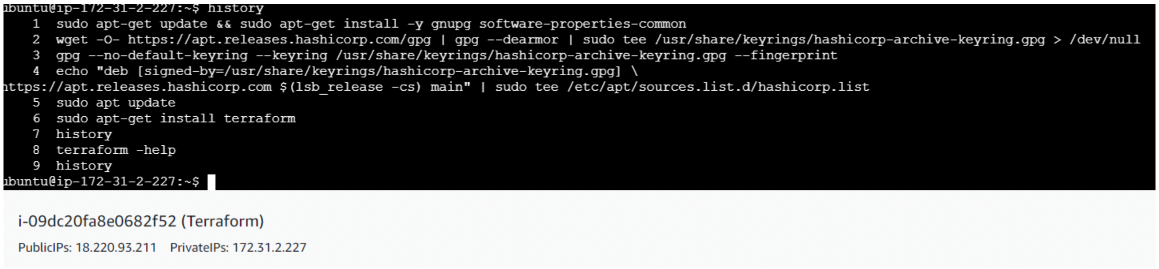
Verify that the installation worked by opening a new terminal session and listing Terraform's available subcommands.

```
$ terraform -help
Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.
##...
```

Add any subcommand to `terraform -help` to learn more about what it does and available options.

```
$ terraform -help plan
```

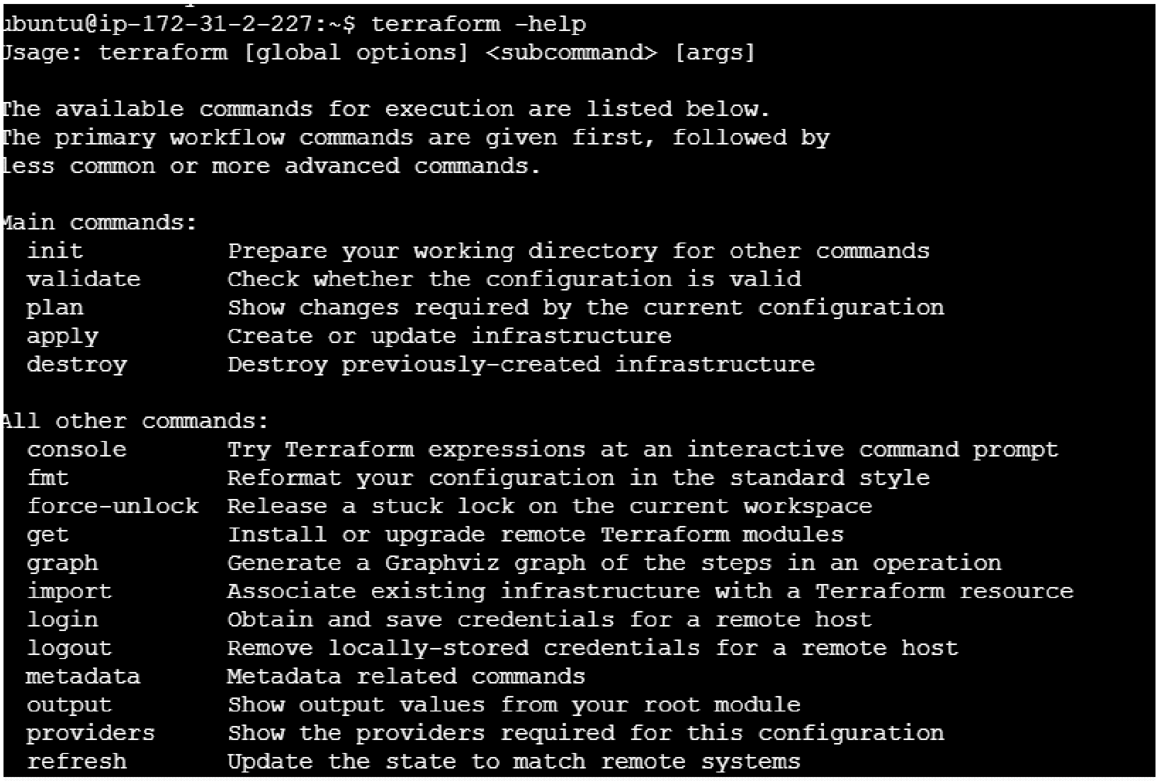
### Troubleshoot



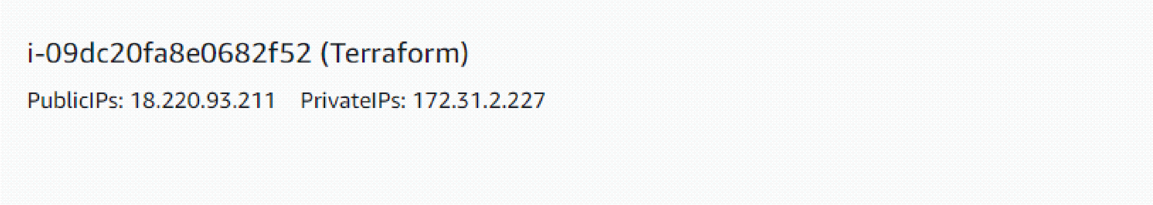
```
ubuntu@ip-172-31-2-227:~$ history
    1  sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
    2  wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null
    3  gpg --no-default-keyring --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg --fingerprint
    4  echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
    5  sudo apt update
    6  sudo apt-get install terraform
    7  history
    8  terraform -help
    9  history
ubuntu@ip-172-31-2-227:~$
```

i-09dc20fa8e0682f52 (Terraform)

PublicIPs: 18.220.93.211    PrivateIPs: 172.31.2.227



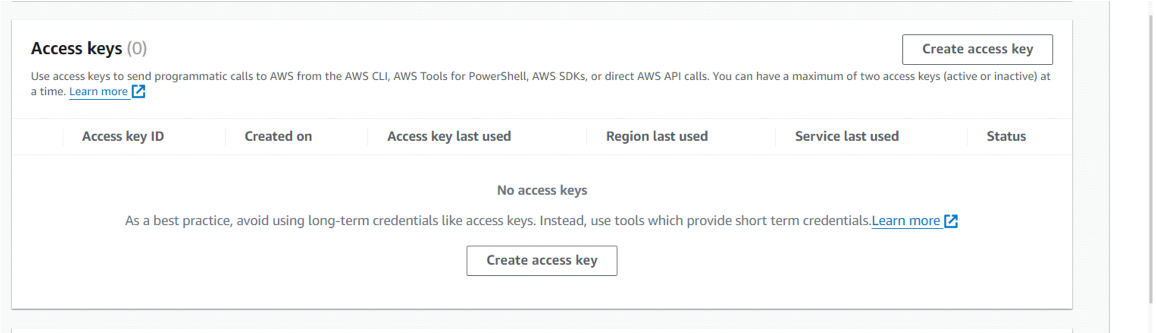
```
ubuntu@ip-172-31-2-227:~$ terraform -help
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  output        Show output values from your root module
  providers     Show the providers required for this configuration
  refresh       Update the state to match remote systems
```



i-09dc20fa8e0682f52 (Terraform)

PublicIPs: 18.220.93.211    PrivateIPs: 172.31.2.227

## Step 2: Create Access and Secret key



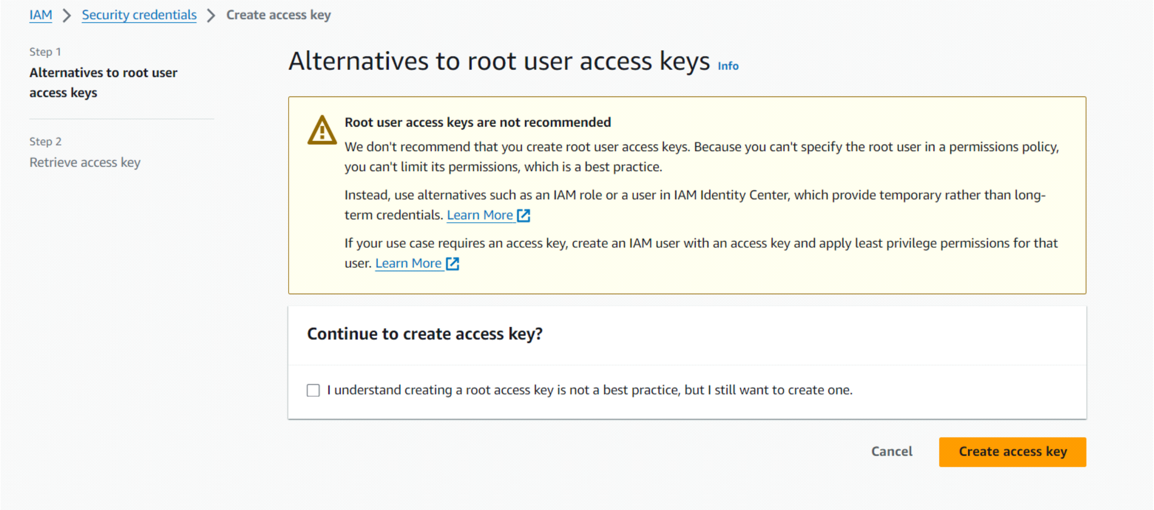
Access keys (0)

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



IAM > Security credentials > Create access key

Step 1
Alternatives to root user access keys

Step 2
Retrieve access key

Alternatives to root user access keys Info

**Root user access keys are not recommended**

We don't recommend that you create root user access keys. Because you can't specify the root user in a permissions policy, you can't limit its permissions, which is a best practice.

Instead, use alternatives such as an IAM role or a user in IAM Identity Center, which provide temporary rather than long-term credentials. Learn More

If your use case requires an access key, create an IAM user with an access key and apply least privilege permissions for that user. Learn More

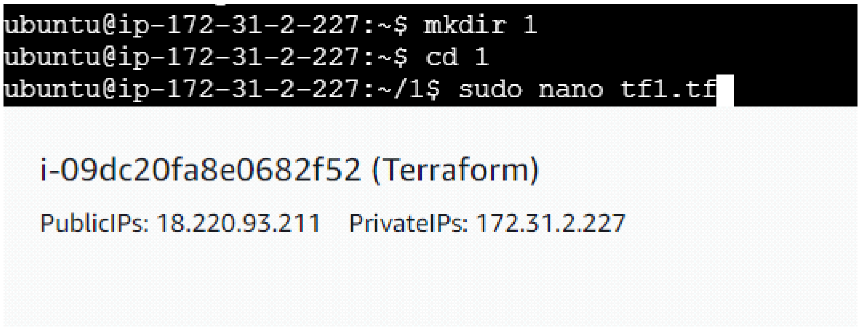
**Continue to create access key?**

□ I understand creating a root access key is not a best practice, but I still want to create one.

Cancel | Create access key

## Step 3: Write the Terraform Configuration

Create a directory for your Terraform configuration files. Inside this directory, create a file named tf1.tf and add the following configuration:



```
ubuntu@ip-172-31-2-227:~$ mkdir 1
ubuntu@ip-172-31-2-227:~$ cd 1
ubuntu@ip-172-31-2-227:~/1$ sudo nano tf1.tf
```

i-09dc20fa8e0682f52 (Terraform)

PublicIPs: 18.220.93.211    PrivateIPs: 172.31.2.227

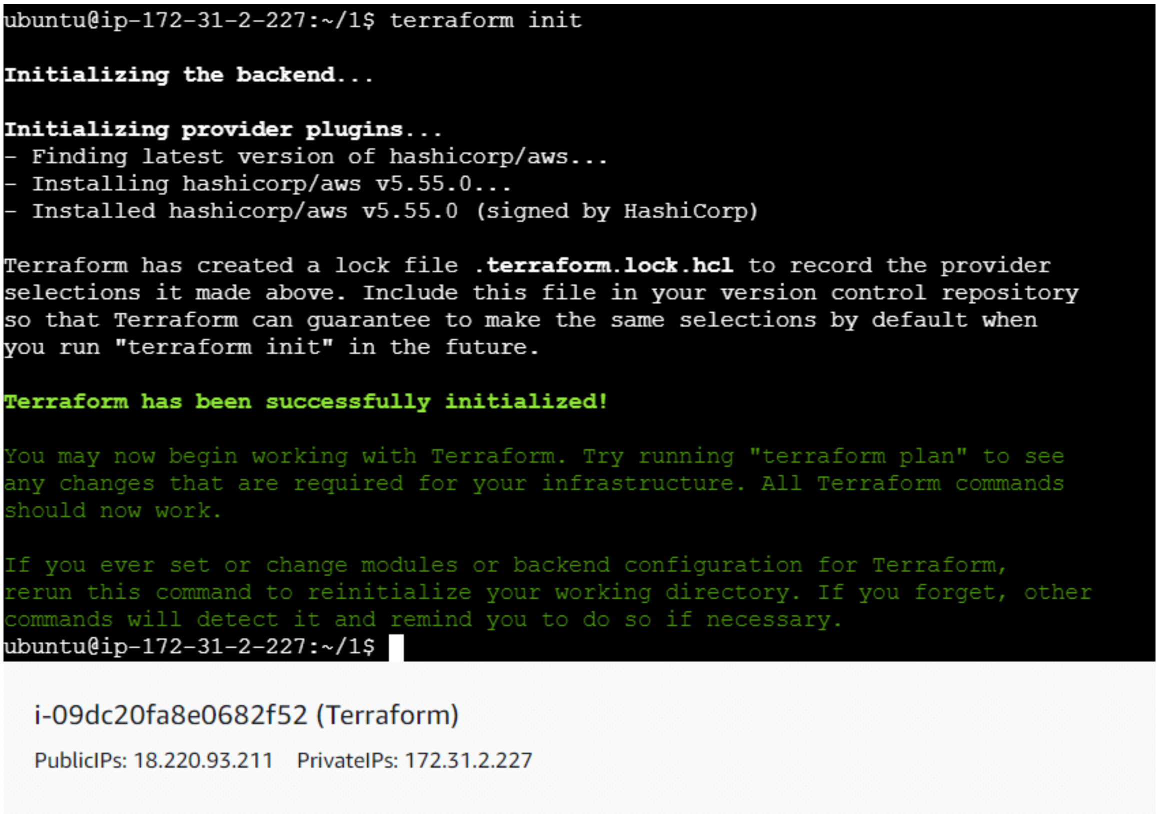
```
provider "aws" {
        region = "us-east-2"
        access_key = " "
        secret_key = " "
}
resource "aws_instance" "assignment-1" {
        ami = "ami-09040d770ffe2224f"
```

```
    key_name = "ohio_key"

    instance_type = "t2.micro"

    tags = {

    name = "assignment-1"

    }

}
```

**Step 4: Initialize Terraform**

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

terraform init



```
ubuntu@ip-172-31-2-227:~/1$ 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:~/1$
```

i-09dc20fa8e0682f52 (Terraform)

PublicIPs: 18.220.93.211    PrivateIPs: 172.31.2.227

**Step 5: Plan the Terraform Deployment**

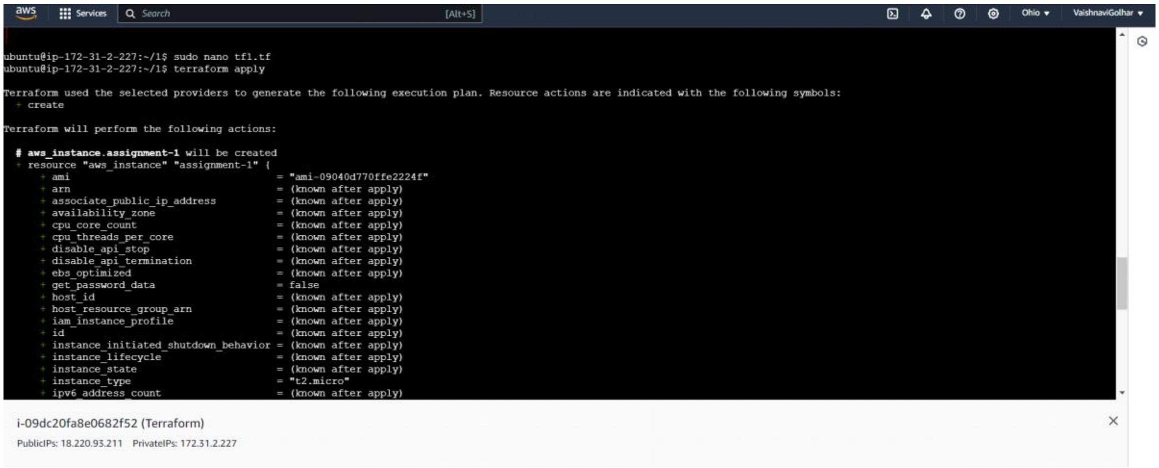
Run the following command to see what resources Terraform will create:

terraform plan

## Step 6: Apply the Terraform Configuration

Run the following command to create the EC2 instance:

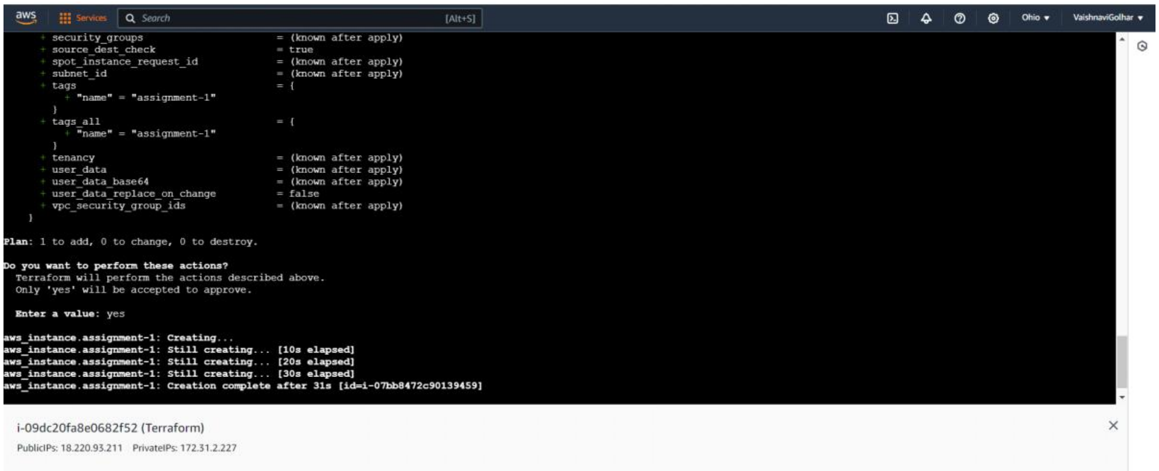
terraform apply



```
ubuntu@ip-172-31-2-227:~/1$ sudo nano tf1.tf
ubuntu@ip-172-31-2-227:~/1$ terraform 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.assignment-1 will be created
  + resource "aws_instance" "assignment-1" {
      + ami                                  = "ami-09040d770ffe2224f"
      + 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)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
```

i-09dc20fa8e0682f52 (Terraform)

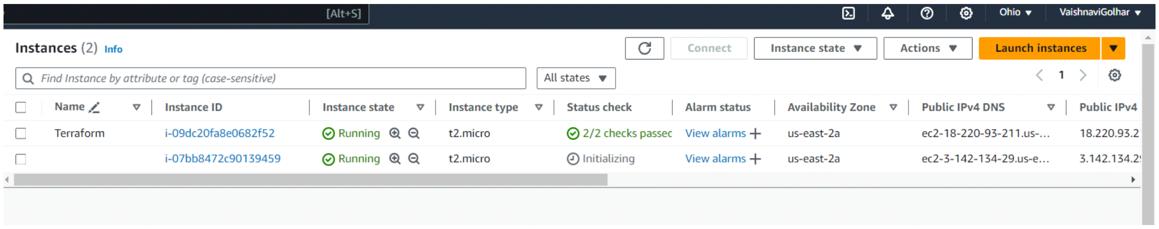
PublicIPs: 18.220.93.211 PrivateIPs: 172.31.2.227



```
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags                                 = {
          + "name" = "assignment-1"
        }
      + tags_all                             = {
          + "name" = "assignment-1"
        }
      + 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: 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: Still creating... [30s elapsed]
aws_instance.assignment-1: Creation complete after 31s [id=i-07bb8472c90139459]
```

i-09dc20fa8e0682f52 (Terraform)

PublicIPs: 18.220.93.211 PrivateIPs: 172.31.2.227



Instances (2) Info

Find Instance by attribute or tag (case-sensitive)

| ☐ | 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 | ✓ Running | t2.micro | Initializing | View alarms + | us-east-2a | ec2-3-142-134-29.us-e... | 3.142.134.2 |