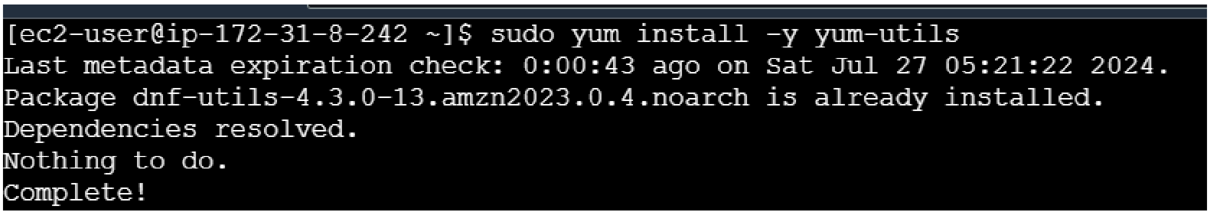
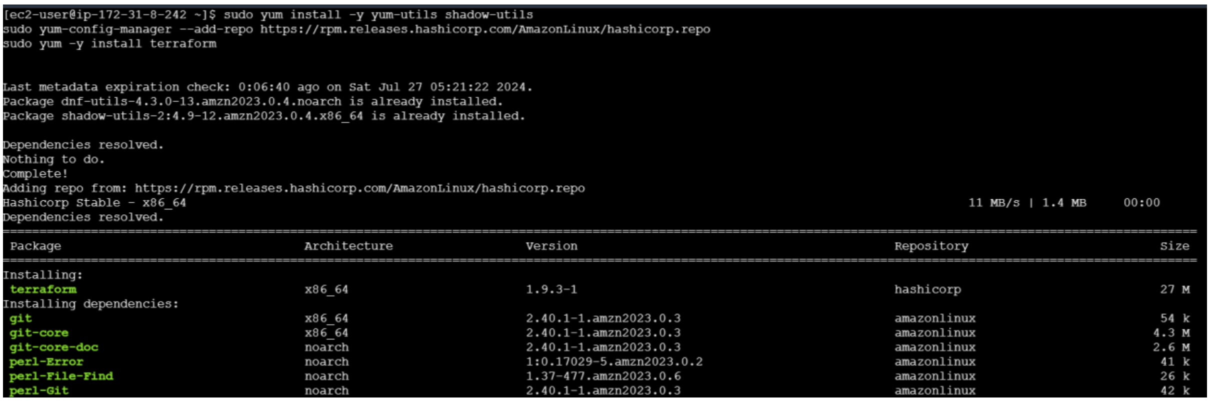
Task: 17 Launch Linux EC2 instances in two regions using a single Terraform file.

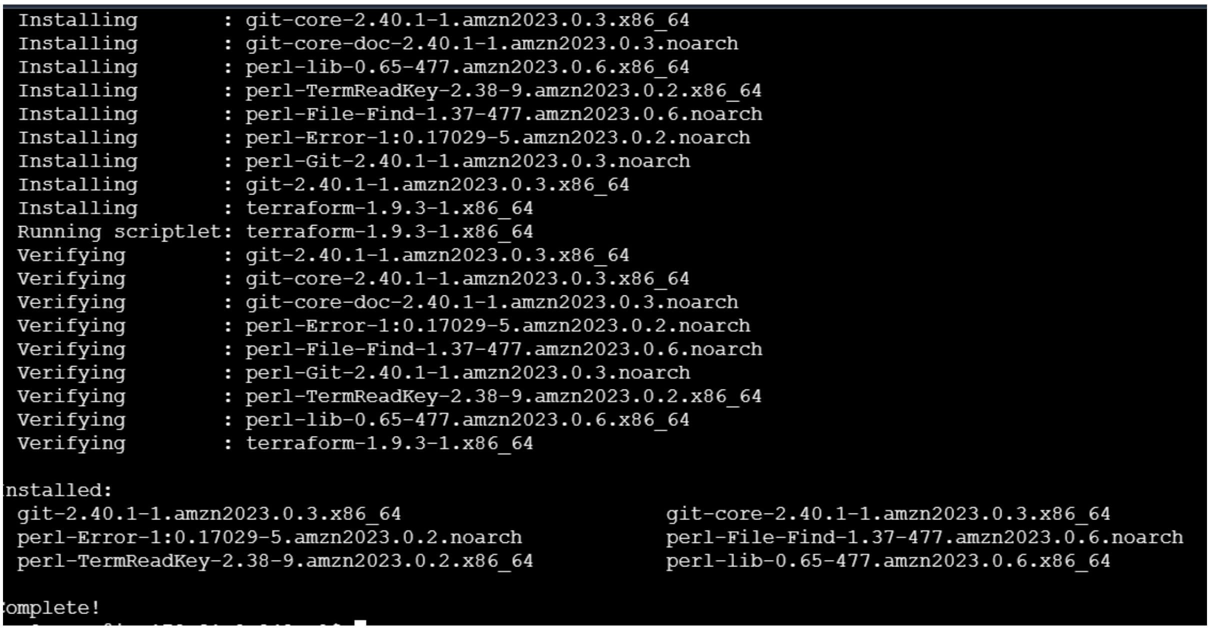
1. Installed terraform:



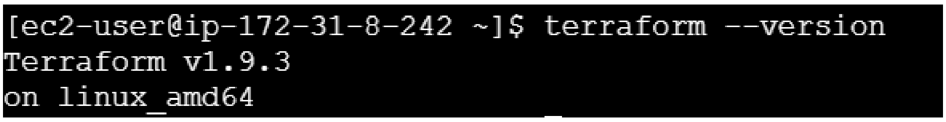
```
[ec2-user@ip-172-31-8-242 ~]$ sudo yum install -y yum-utils
Last metadata expiration check: 0:00:43 ago on Sat Jul 27 05:21:22 2024.
Package dnf-utils-4.3.0-13.amzn2023.0.4.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```



```
[ec2-user@ip-172-31-8-242 ~]$ sudo yum install -y yum-utils shadow-utils
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform


Last metadata expiration check: 0:06:40 ago on Sat Jul 27 05:21:22 2024.
Package dnf-utils-4.3.0-13.amzn2023.0.4.noarch is already installed.
Package shadow-utils-2:4.9-12.amzn2023.0.4.x86_64 is already installed.

Dependencies resolved.
Nothing to do.
Complete!
Adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
Hashicorp Stable - x86_64                                                                  11 MB/s | 1.4 MB     00:00
Dependencies resolved.
========================================================================================================================
 Package                   Architecture          Version                                  Repository              Size
========================================================================================================================
Installing:
 terraform                 x86_64                1.9.3-1                                  hashicorp               27 M
Installing dependencies:
 git                       x86_64                2.40.1-1.amzn2023.0.3                    amazonlinux             54 k
 git-core                  x86_64                2.40.1-1.amzn2023.0.3                    amazonlinux            4.3 M
 git-core-doc              noarch                2.40.1-1.amzn2023.0.3                    amazonlinux            2.6 M
 perl-Error                noarch                1:0.17029-5.amzn2023.0.2                 amazonlinux             41 k
 perl-File-Find            noarch                1.37-477.amzn2023.0.6                    amazonlinux             26 k
 perl-Git                  noarch                2.40.1-1.amzn2023.0.3                    amazonlinux             42 k
```

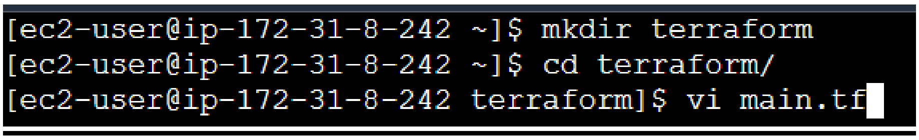


```
 Installing       : git-core-2.40.1-1.amzn2023.0.3.x86_64
 Installing       : git-core-doc-2.40.1-1.amzn2023.0.3.noarch
 Installing       : perl-lib-0.65-477.amzn2023.0.6.x86_64
 Installing       : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64
 Installing       : perl-File-Find-1.37-477.amzn2023.0.6.noarch
 Installing       : perl-Error-1:0.17029-5.amzn2023.0.2.noarch
 Installing       : perl-Git-2.40.1-1.amzn2023.0.3.noarch
 Installing       : git-2.40.1-1.amzn2023.0.3.x86_64
 Installing       : terraform-1.9.3-1.x86_64
 Running scriptlet: terraform-1.9.3-1.x86_64
 Verifying        : git-2.40.1-1.amzn2023.0.3.x86_64
 Verifying        : git-core-2.40.1-1.amzn2023.0.3.x86_64
 Verifying        : git-core-doc-2.40.1-1.amzn2023.0.3.noarch
 Verifying        : perl-Error-1:0.17029-5.amzn2023.0.2.noarch
 Verifying        : perl-File-Find-1.37-477.amzn2023.0.6.noarch
 Verifying        : perl-Git-2.40.1-1.amzn2023.0.3.noarch
 Verifying        : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64
 Verifying        : perl-lib-0.65-477.amzn2023.0.6.x86_64
 Verifying        : terraform-1.9.3-1.x86_64

nstalled:
 git-2.40.1-1.amzn2023.0.3.x86_64                     git-core-2.40.1-1.amzn2023.0.3.x86_64
 perl-Error-1:0.17029-5.amzn2023.0.2.noarch           perl-File-Find-1.37-477.amzn2023.0.6.noarch
 perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64          perl-lib-0.65-477.amzn2023.0.6.x86_64

omplete!
```



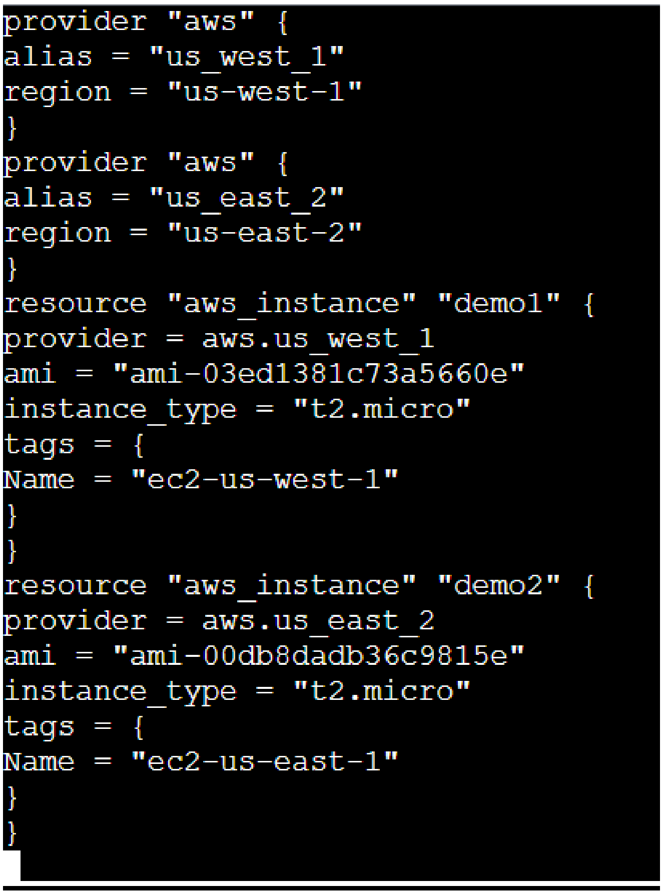
```
[ec2-user@ip-172-31-8-242 ~]$ terraform --version
Terraform v1.9.3
on linux_amd64
```

2. Create a directory for write terraform file:

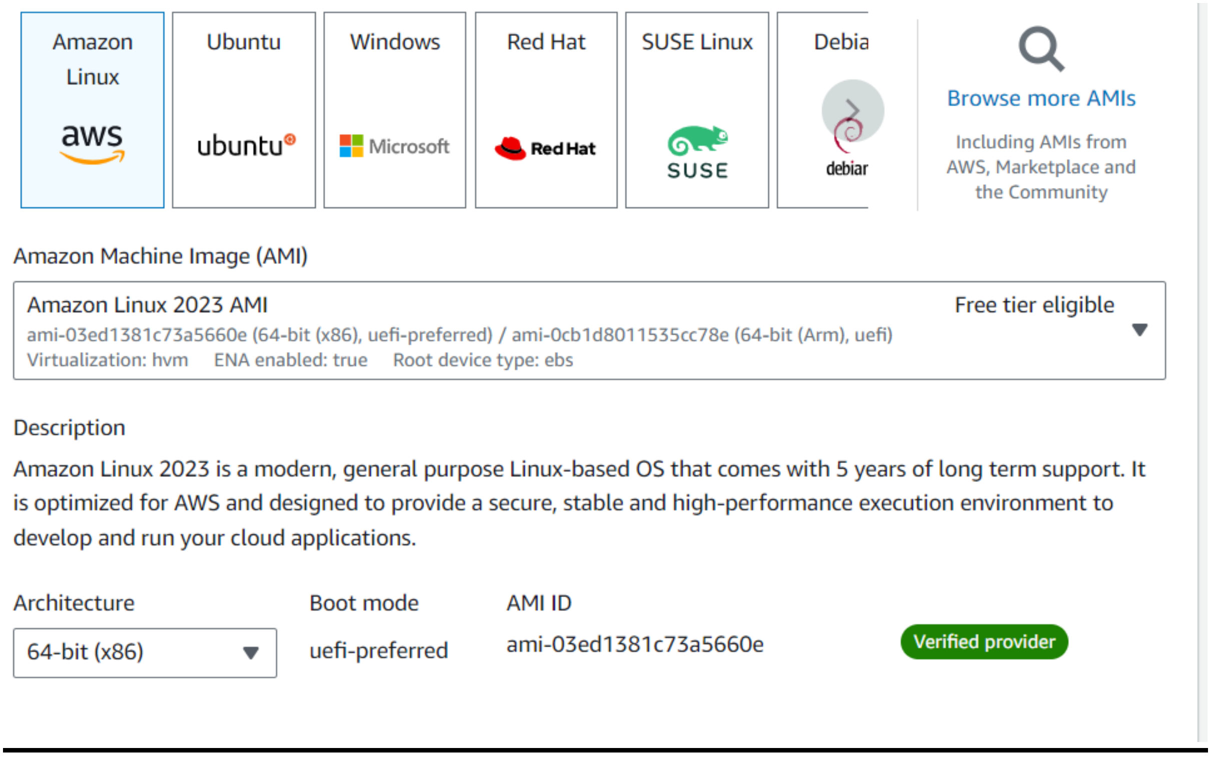


```
[ec2-user@ip-172-31-8-242 ~]$ mkdir terraform
[ec2-user@ip-172-31-8-242 ~]$ cd terraform/
[ec2-user@ip-172-31-8-242 terraform]$ vi main.tf
```

3. Main.tf file:



```
provider "aws" {
alias = "us_west_1"
region = "us-west-1"
}
provider "aws" {
alias = "us_east_2"
region = "us-east-2"
}
resource "aws_instance" "demo1" {
provider = aws.us_west_1
ami = "ami-03ed1381c73a5660e"
instance_type = "t2.micro"
tags = {
Name = "ec2-us-west-1"
}
}
resource "aws_instance" "demo2" {
provider = aws.us_east_2
ami = "ami-00db8dadb36c9815e"
instance_type = "t2.micro"
tags = {
Name = "ec2-us-east-1"
}
}
```



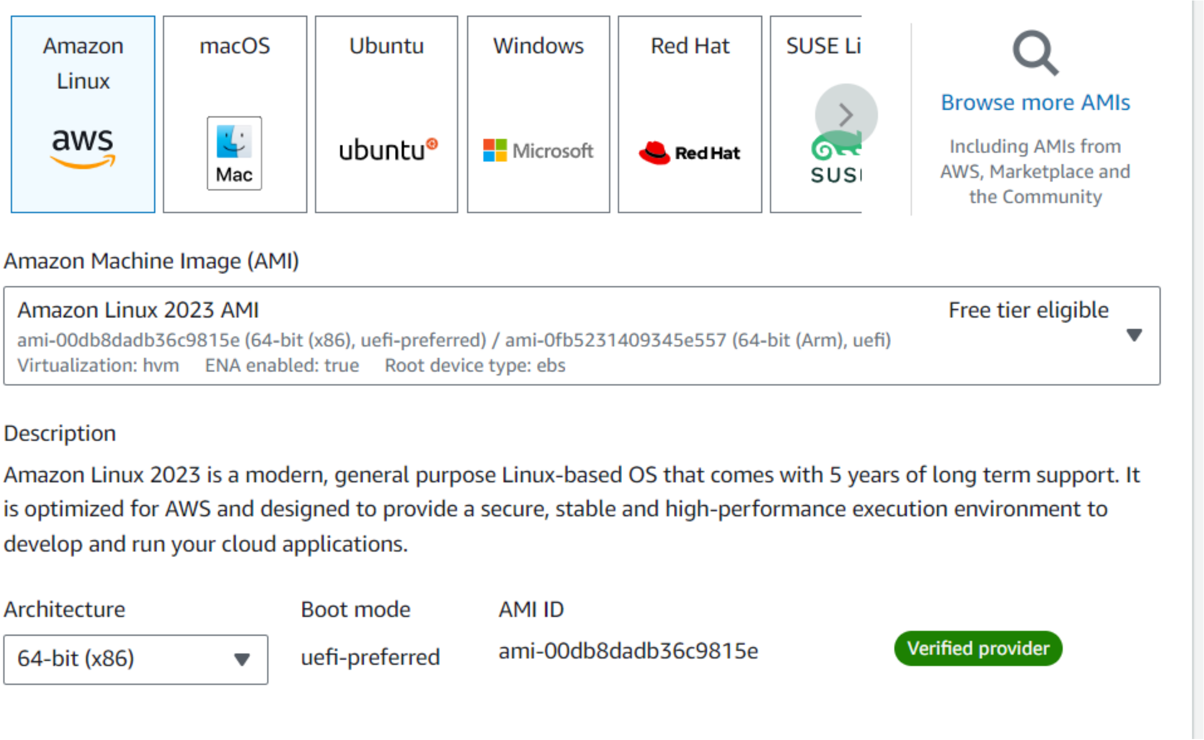
Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible
ami-03ed1381c73a5660e (64-bit (x86), uefi-preferred) / ami-0cb1d8011535cc78e (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture	Boot mode	AMI ID	
64-bit (x86)	uefi-preferred	ami-03ed1381c73a5660e	Verified provider



Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible
ami-00db8dadb36c9815e (64-bit (x86), uefi-preferred) / ami-0fb5231409345e557 (64-bit (Arm), uefi)
Virtualization: hvm   ENA enabled: true   Root device type: ebs

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

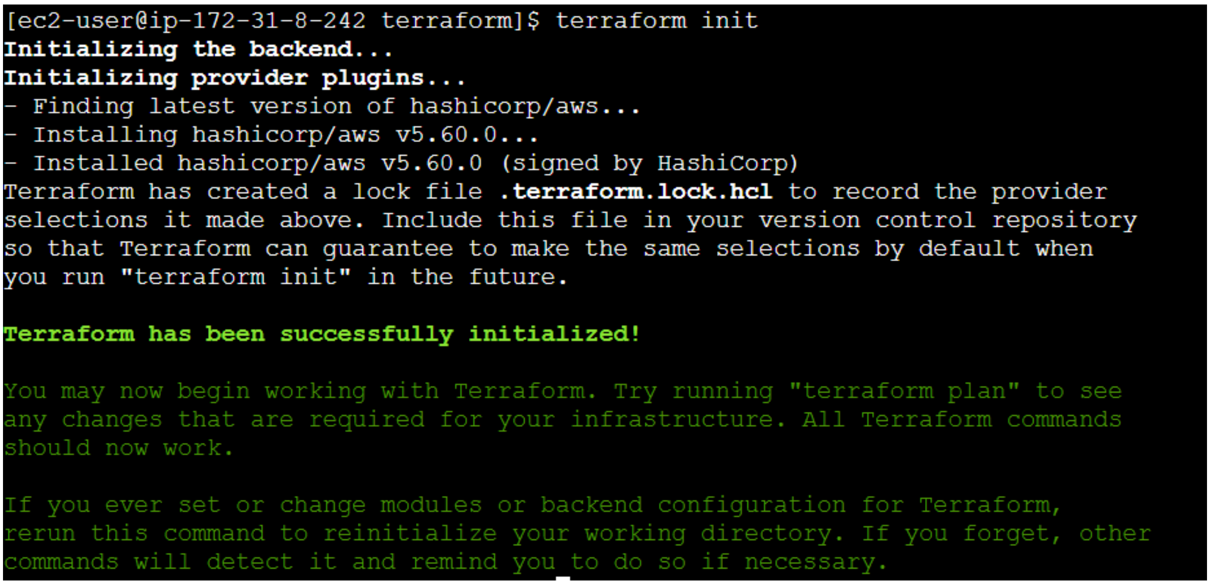
Architecture: 64-bit (x86)

Boot mode: uefi-preferred

AMI ID: ami-00db8dadb36c9815e

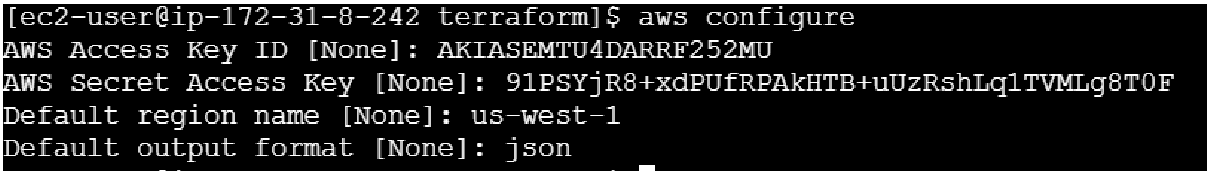
Verified provider

4. Terraform init:



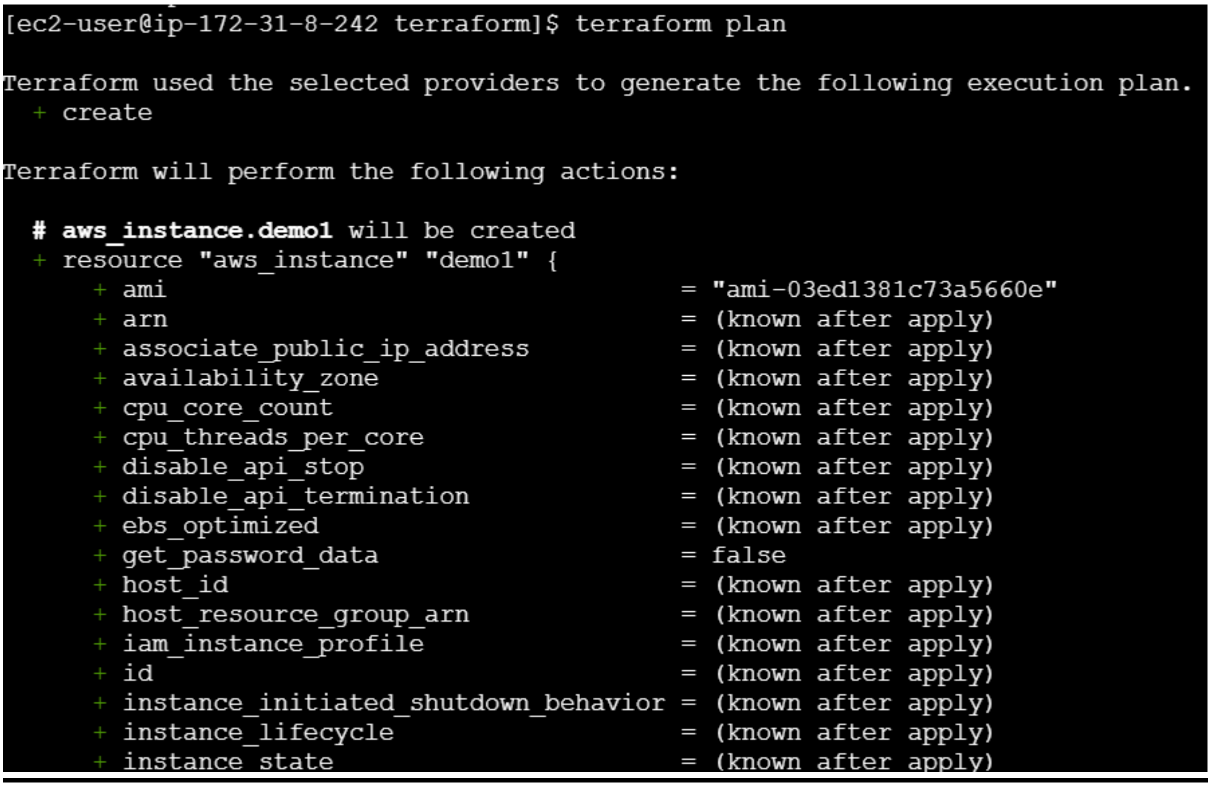
```
[ec2-user@ip-172-31-8-242 terraform]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.60.0...
- Installed hashicorp/aws v5.60.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl 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.
```

5. Aws cli configure:

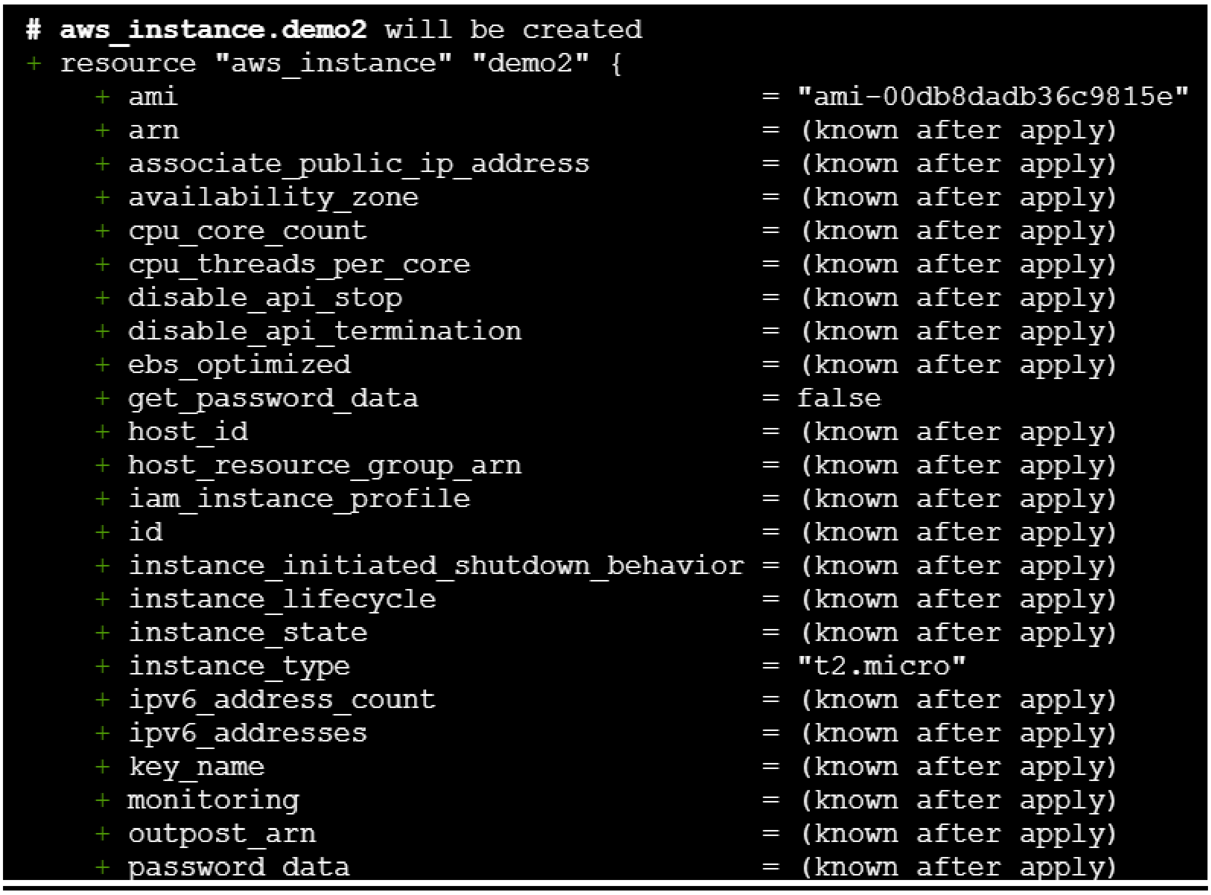


```
[ec2-user@ip-172-31-8-242 terraform]$ aws configure
AWS Access Key ID [None]: AKIASEMTU4DARRF252MU
AWS Secret Access Key [None]: 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
Default region name [None]: us-west-1
Default output format [None]: json
```

6. Terraform paln:

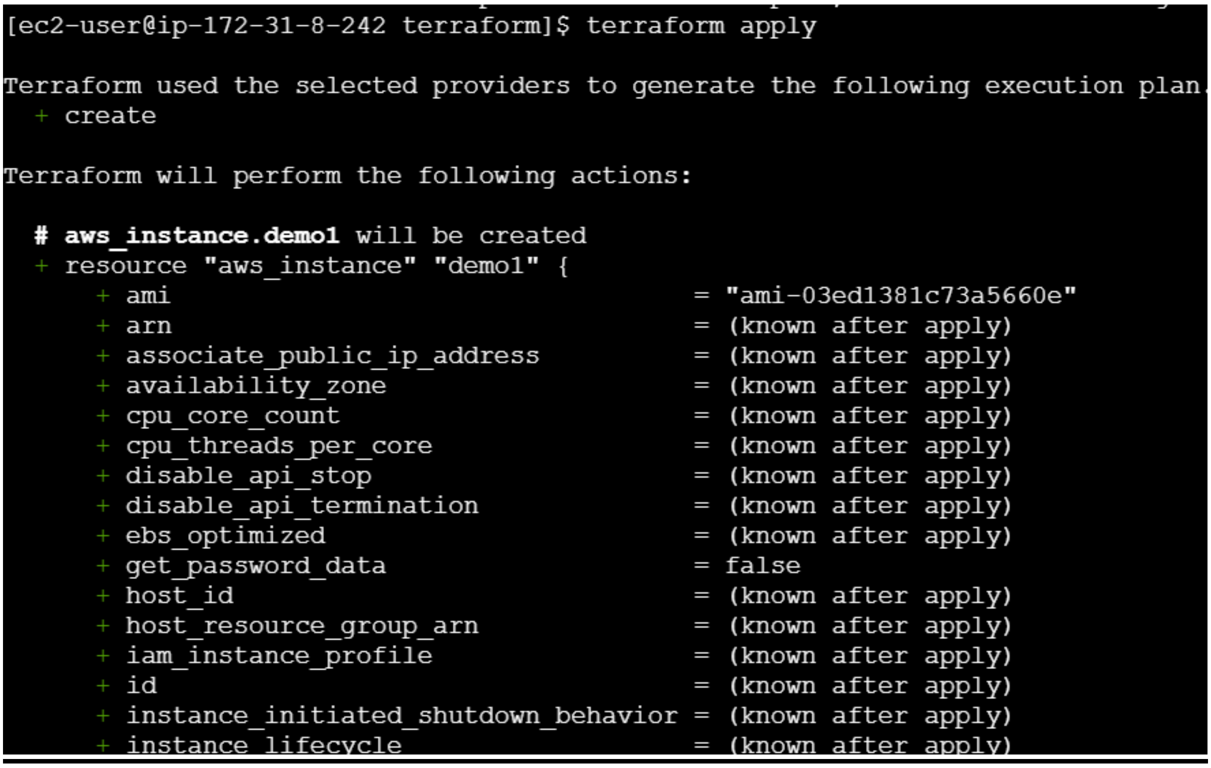


```
[ec2-user@ip-172-31-8-242 terraform]$ terraform plan

Terraform used the selected providers to generate the following execution plan.
  + create

Terraform will perform the following actions:

  # aws_instance.demo1 will be created
  + resource "aws_instance" "demo1" {
      + ami                                  = "ami-03ed1381c73a5660e"
      + 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)
```

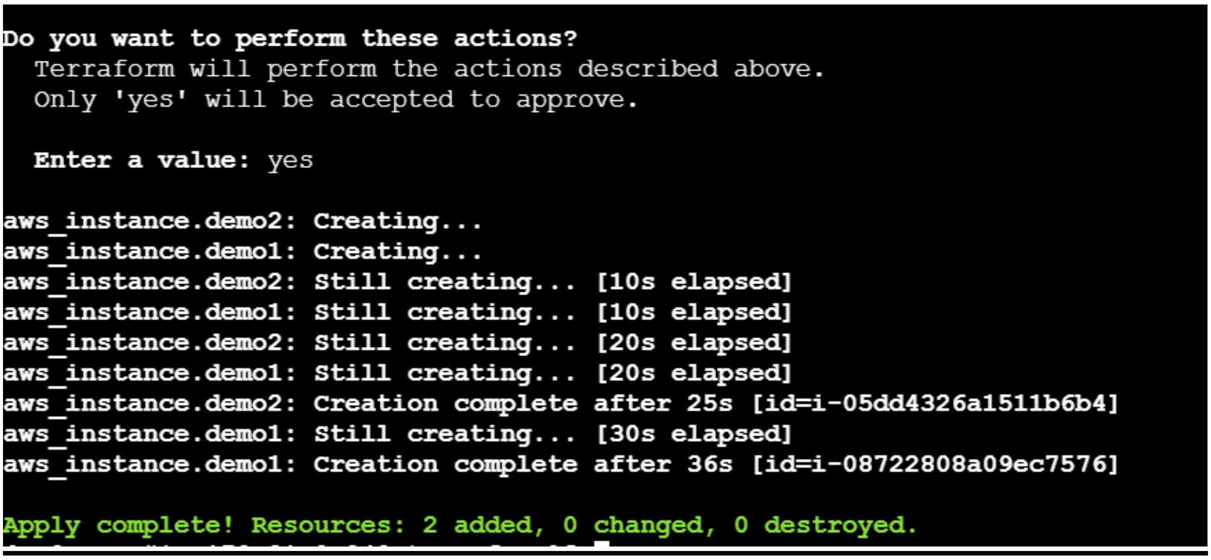


```
  # aws_instance.demo2 will be created
  + resource "aws_instance" "demo2" {
      + ami                                  = "ami-00db8dadb36c9815e"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (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)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
```

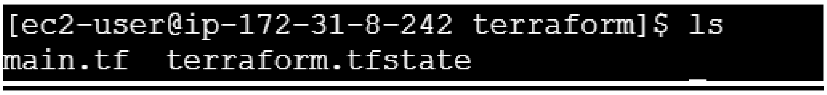
7. Terraform apply:



```
[ec2-user@ip-172-31-8-242 terraform]$ terraform apply

Terraform used the selected providers to generate the following execution plan.
  + create

Terraform will perform the following actions:

  # aws_instance.demo1 will be created
  + resource "aws_instance" "demo1" {
      + ami                                  = "ami-03ed1381c73a5660e"
      + 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)
```

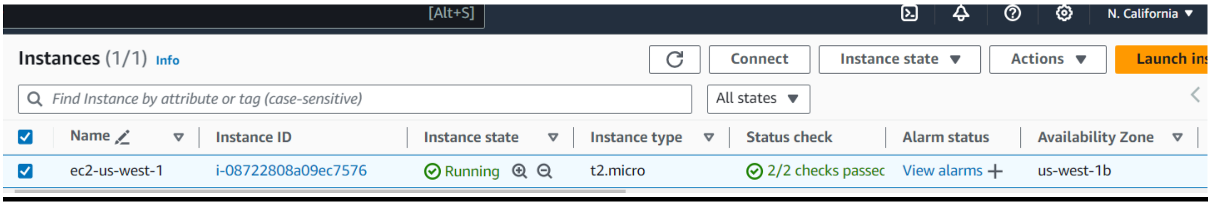


```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.demo2: Creating...
aws_instance.demo1: Creating...
aws_instance.demo2: Still creating... [10s elapsed]
aws_instance.demo1: Still creating... [10s elapsed]
aws_instance.demo2: Still creating... [20s elapsed]
aws_instance.demo1: Still creating... [20s elapsed]
aws_instance.demo2: Creation complete after 25s [id=i-05dd4326a1511b6b4]
aws_instance.demo1: Still creating... [30s elapsed]
aws_instance.demo1: Creation complete after 36s [id=i-08722808a09ec7576]

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

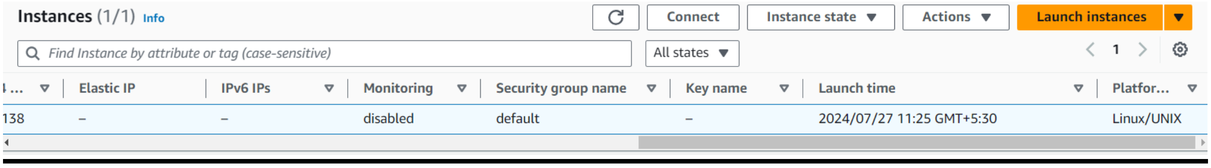


```
[ec2-user@ip-172-31-8-242 terraform]$ ls
main.tf  terraform.tfstate
```

8. Ec2 created in region us-west-1:

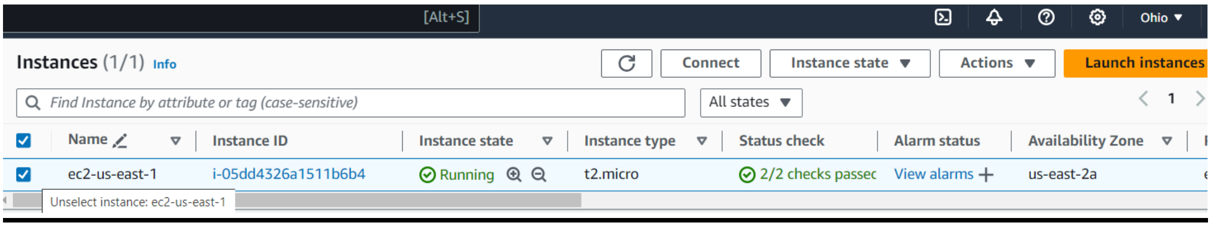


☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☑	ec2-us-west-1	i-08722808a09ec7576	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-west-1b

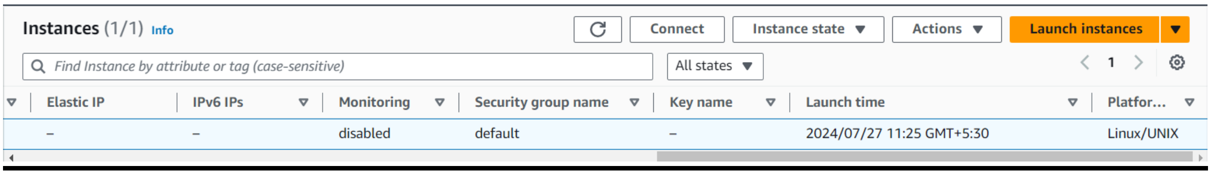


I ...	Elastic IP	IPv6 IPs	Monitoring	Security group name	Key name	Launch time	Platfor...
138	–	–	disabled	default	–	2024/07/27 11:25 GMT+5:30	Linux/UNIX

9. Ec2 created in region us-east-2:



☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☑	ec2-us-east-1	i-05dd4326a1511b6b4	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-2a



	Elastic IP	IPv6 IPs	Monitoring	Security group name	Key name	Launch time	Platfor...
	–	–	disabled	default	–	2024/07/27 11:25 GMT+5:30	Linux/UNIX