Mayuri Yerunde
D15B , 70

EXPERIMENT - 6

AIM: To Build , change and destroy AWS | GCP | Azure | Digital Ocean Infrastructure using terraform.

THEORY:

Terraform core concepts:-

1. Variables: Also used as Input-variable, It is key-value pair used by terraform modules to allow customization.

2. Provider: It is a plugin to interact with APIs of service and access its related resources.

3. module: It is a folder with terraform templates where all configurations are defined.

4. State: It consists of cached information about infrastructure managed by terraform and its related configuration.

5. Resources: It refers to a block of one or more infrastructure objects which are used in configuring and managing the infrastructure.

6. Data Source: It is implemented by providers to return information on external object to terraform

7. output values: These are return values of terraform module that can be used by other configuration

(8) Plan: It is one of the stages where it determines what needs to be treated, updated or destroyed to move from real state of infrastructure to desired state.

(9) Apply: It is one of the stages where it applies changes in real/current state of infrastructure in order to move to desired state.

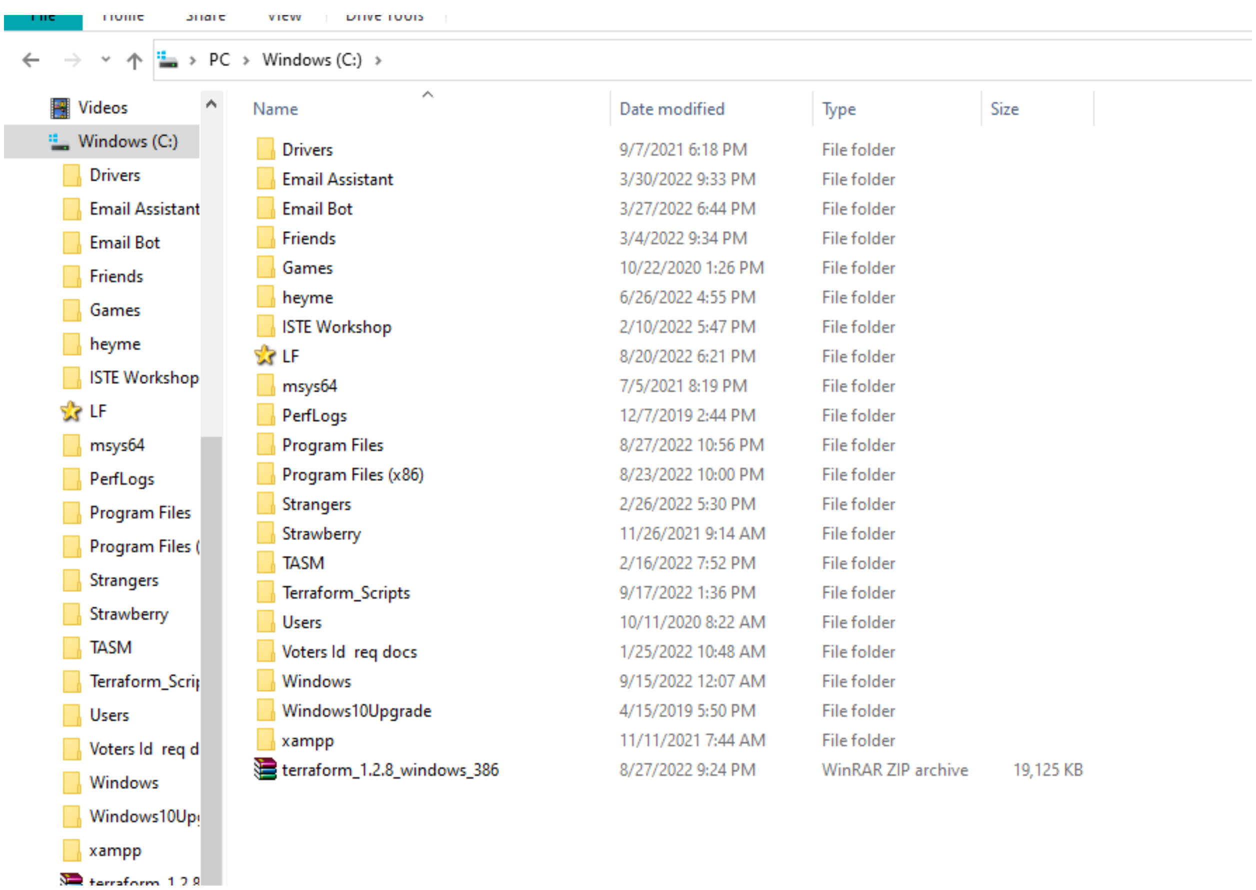
Terraform providers:- Responsible for understanding API interaction and exposing resources. It is an executable plug-in that contains code necessary to interact with API of source.

Terraform configuration files:-
configuration files are setup of files used to describe infrastructure in terraform files. Configuration files let you write a configuration that declares your desired state.

Conclusion:-
using terraform, on AWS EC2 instance, a S3 bucket, and docker container was created and destroyed successfully.

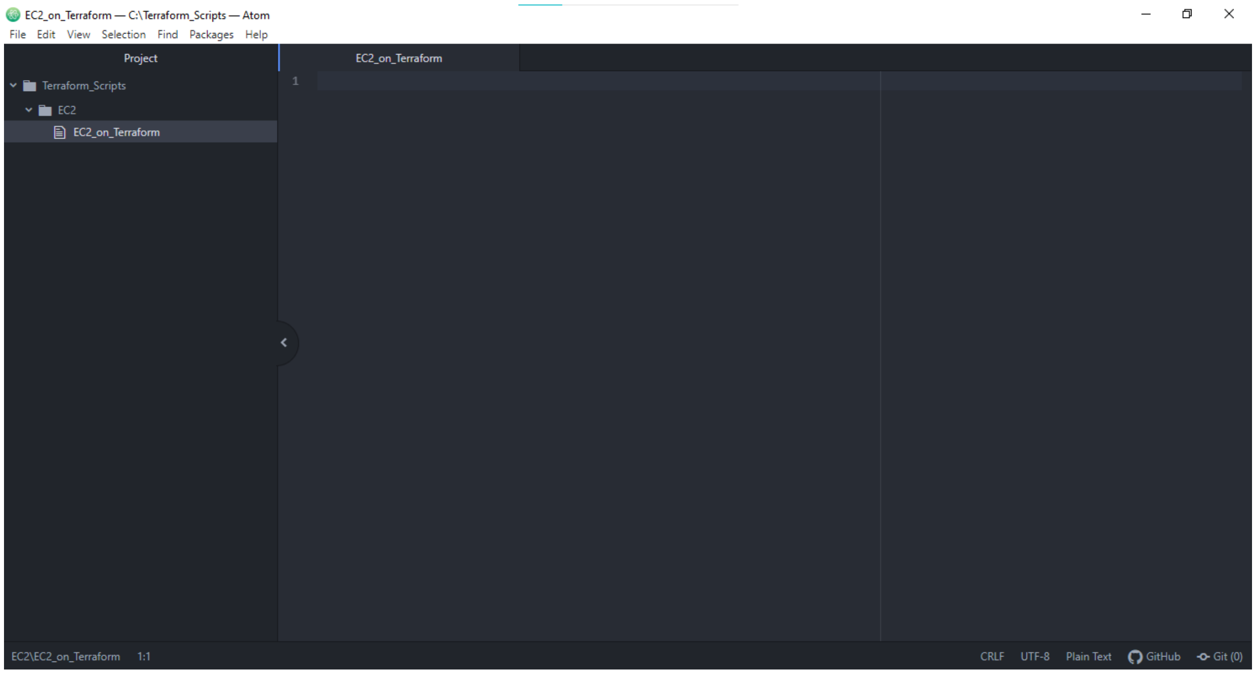
EC2 Instance creation.

1. Create a folder in any drive named ‘Terraform_Scripts’. Inside that folder create a folder for EC2.

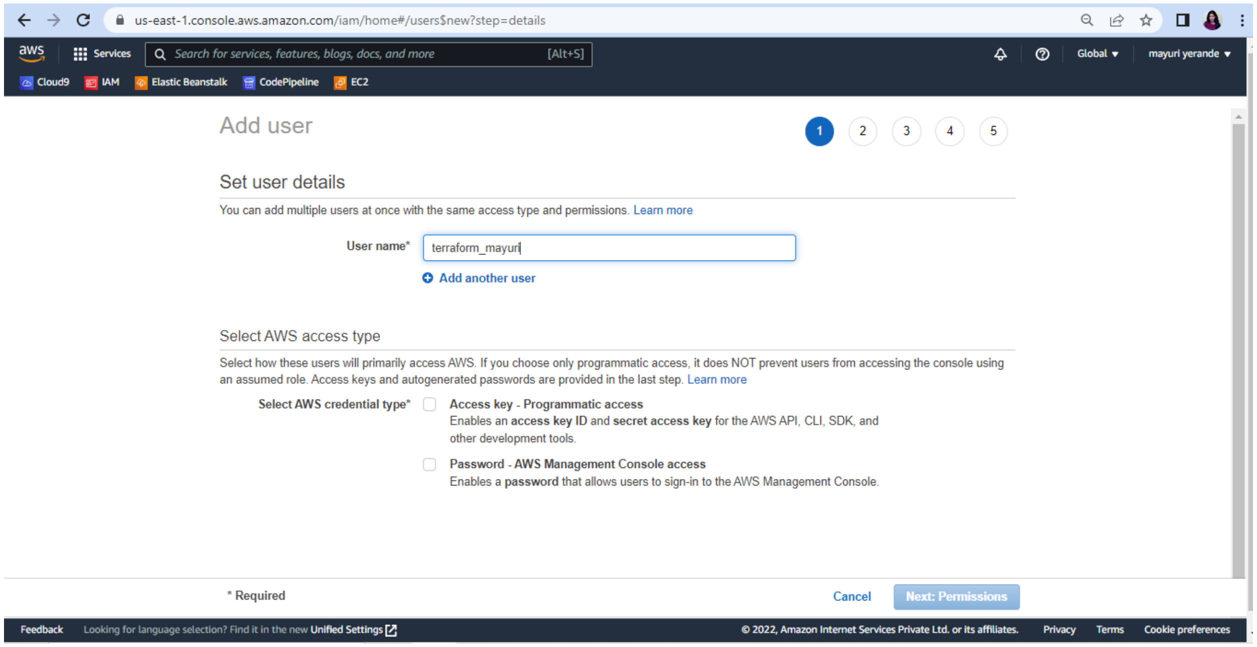


Name	Date modified	Type	Size
Drivers	9/7/2021 6:18 PM	File folder	
Email Assistant	3/30/2022 9:33 PM	File folder	
Email Bot	3/27/2022 6:44 PM	File folder	
Friends	3/4/2022 9:34 PM	File folder	
Games	10/22/2020 1:26 PM	File folder	
heyme	6/26/2022 4:55 PM	File folder	
ISTE Workshop	2/10/2022 5:47 PM	File folder	
LF	8/20/2022 6:21 PM	File folder	
msys64	7/5/2021 8:19 PM	File folder	
PerfLogs	12/7/2019 2:44 PM	File folder	
Program Files	8/27/2022 10:56 PM	File folder	
Program Files (x86)	8/23/2022 10:00 PM	File folder	
Strangers	2/26/2022 5:30 PM	File folder	
Strawberry	11/26/2021 9:14 AM	File folder	
TASM	2/16/2022 7:52 PM	File folder	
Terraform_Scripts	9/17/2022 1:36 PM	File folder	
Users	10/11/2020 8:22 AM	File folder	
Voters Id req docs	1/25/2022 10:48 AM	File folder	
Windows	9/15/2022 12:07 AM	File folder	
Windows10Upgrade	4/15/2019 5:50 PM	File folder	
xampp	11/11/2021 7:44 AM	File folder	
terraform_1.2.8_windows_386	8/27/2022 9:24 PM	WinRAR ZIP archive	19,125 KB

2. Open a any editor inside the EC2 folder and create a text file and start editing the file


EC2_on_Terraform — C:\Terraform_Scripts — Atom
File Edit View Selection Find Packages Help
Project
Terraform_Scripts
EC2
EC2_on_Terraform
EC2_on_Terraform
EC2\EC2_on_Terraform 1:1
CRLF UTF-8 Plain Text GitHub Git (0)

3. Open AWS Console and Search for IAM to get Access Key ID and Secrete Key. To get those, add a new user.


us-east-1.console.aws.amazon.com/iam/home#/users$new?step=details

Add user

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* terraform_mayuri

Add another user

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. Learn more

Select AWS credential type*

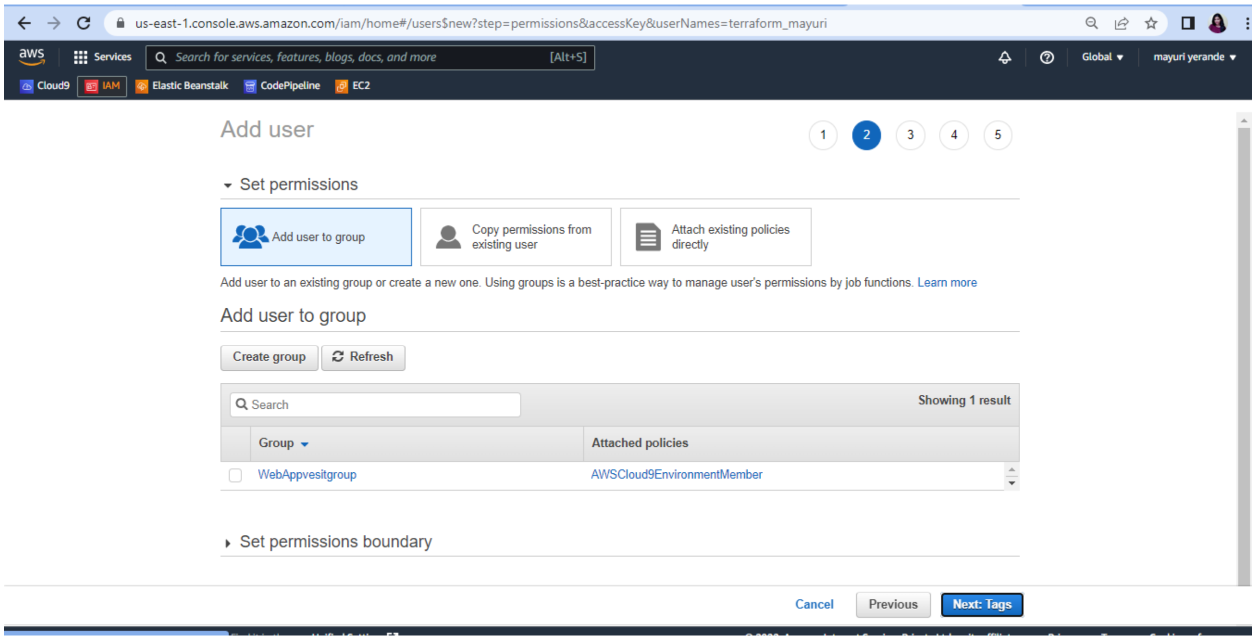
Access key - Programmatic access
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

Password - AWS Management Console access
Enables a **password** that allows users to sign-in to the AWS Management Console.

* Required

Services
Search for services, features, blogs, docs, and more [Alt+S]
Global
mayuri yerande
Cloud9 IAM Elastic Beanstalk CodePipeline EC2
1 2 3 4 5
Cancel
Next: Permissions
Feedback Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

4. Add the user to a new group. For that you'll need to create a new group.


us-east-1.console.aws.amazon.com/iam/home#/users$new?step=permissions&accessKey&userNames=terraform_mayuri

Add user

Set permissions

Add user to group | Copy permissions from existing user | Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Add user to group

Create group | Refresh

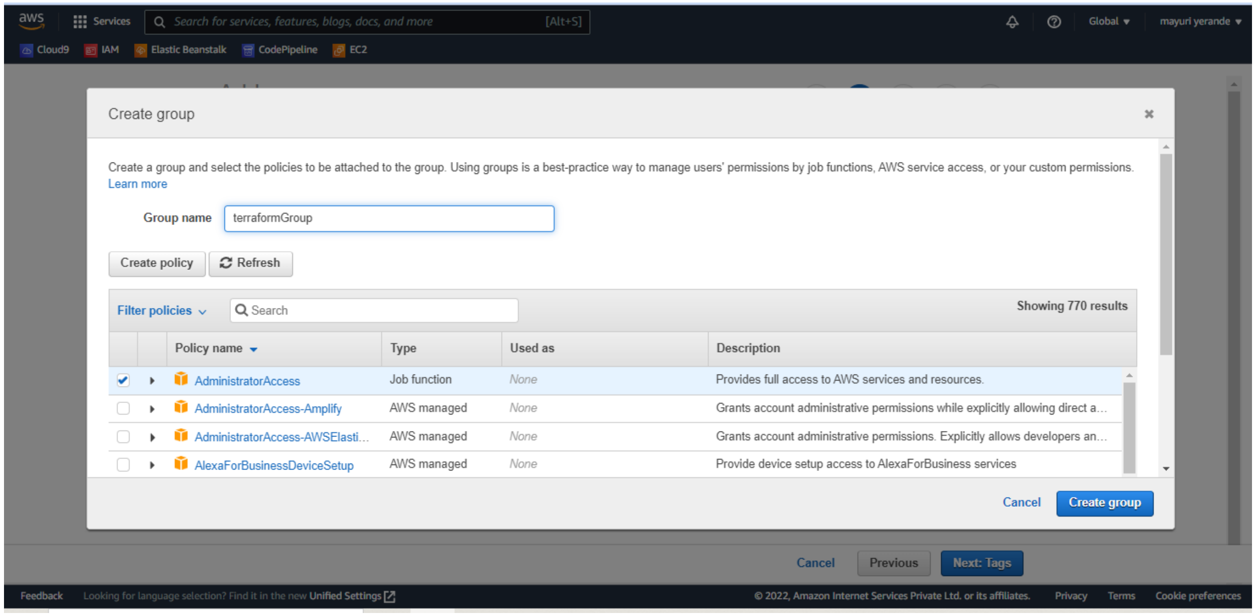
Showing 1 result

Group	Attached policies
WebAppvesitgroup	AWSCloud9EnvironmentMember

Set permissions boundary

Cancel | Previous | Next: Tags

5. Create a new group named 'TerraformGroup'. Attach 'AdministratorAccess' policy to that group.


Create group

Create a group and select the policies to be attached to the group. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. Learn more

Group name: terraformGroup

Create policy | Refresh

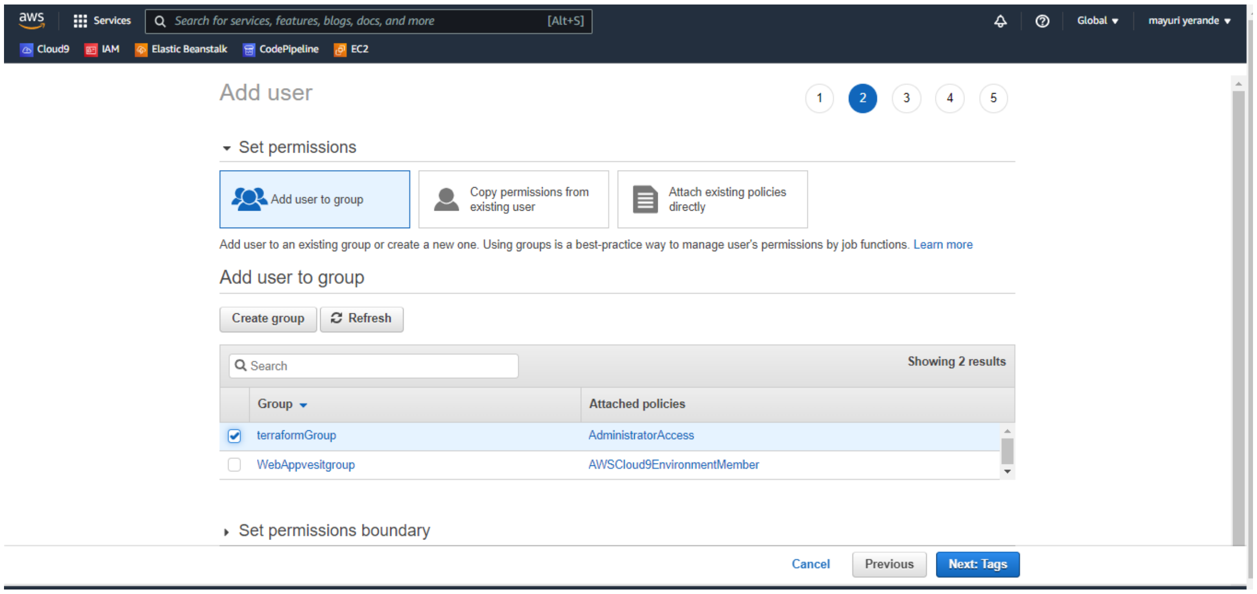
Filter policies | Search | Showing 770 results

	Policy name	Type	Used as	Description
✓	AdministratorAccess	Job function	None	Provides full access to AWS services and resources.
	AdministratorAccess-Amplify	AWS managed	None	Grants account administrative permissions while explicitly allowing direct a...
	AdministratorAccess-AWSElasti...	AWS managed	None	Grants account administrative permissions. Explicitly allows developers an...
	AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaForBusiness services

Cancel | Create group

© 2022, Amazon Internet Services Private Ltd. or its affiliates.

6. Add the user to that group and create a user



Add user

1 2 3 4 5

Set permissions

Add user to group | Copy permissions from existing user | Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Add user to group

Create group | Refresh

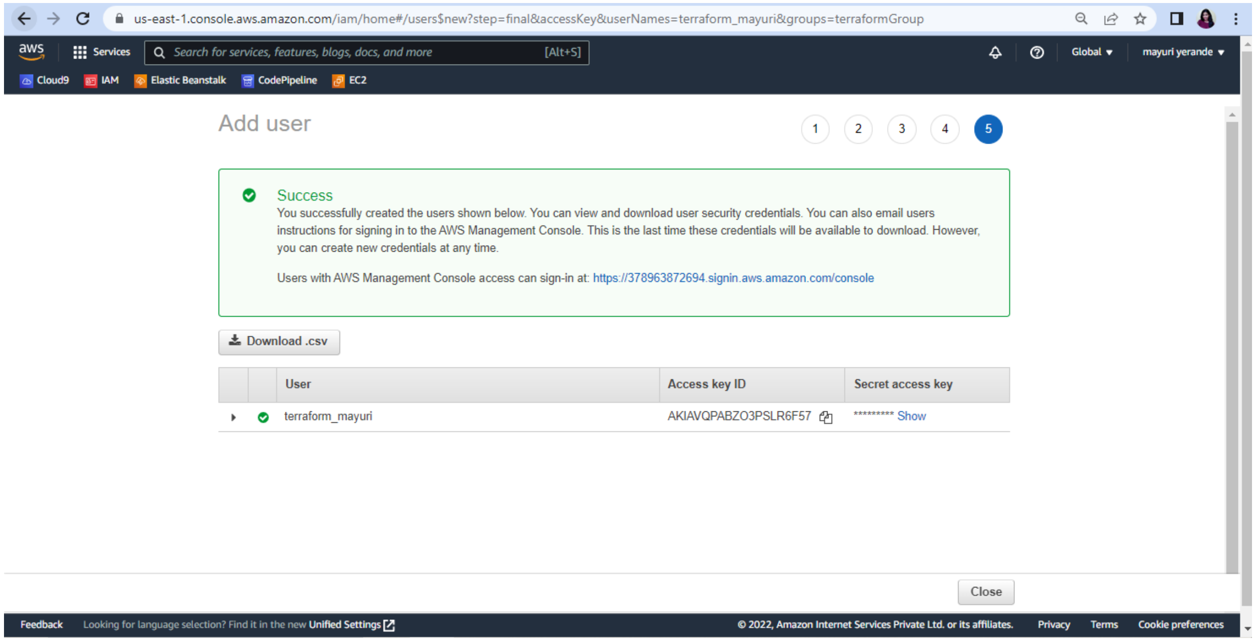
Search | Showing 2 results

	Group	Attached policies
☑	terraformGroup	AdministratorAccess
☐	WebAppvesitgroup	AWSCloud9EnvironmentMember

Set permissions boundary

Cancel | Previous | Next: Tags

7. After successful creation of the user, you can see the ‘Access Key’ and ‘Secret Key’ next to it. Copy it and store as it will be required for further use.


us-east-1.console.aws.amazon.com/iam/home#/users$new?step=final&accessKey&userNames=terraform_mayuri&groups=terraformGroup
aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
Global
mayuri yerande
Cloud9
IAM
Elastic Beanstalk
CodePipeline
EC2

Add user

1 2 3 4 5

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://378963872694.signin.aws.amazon.com/console

Download .csv

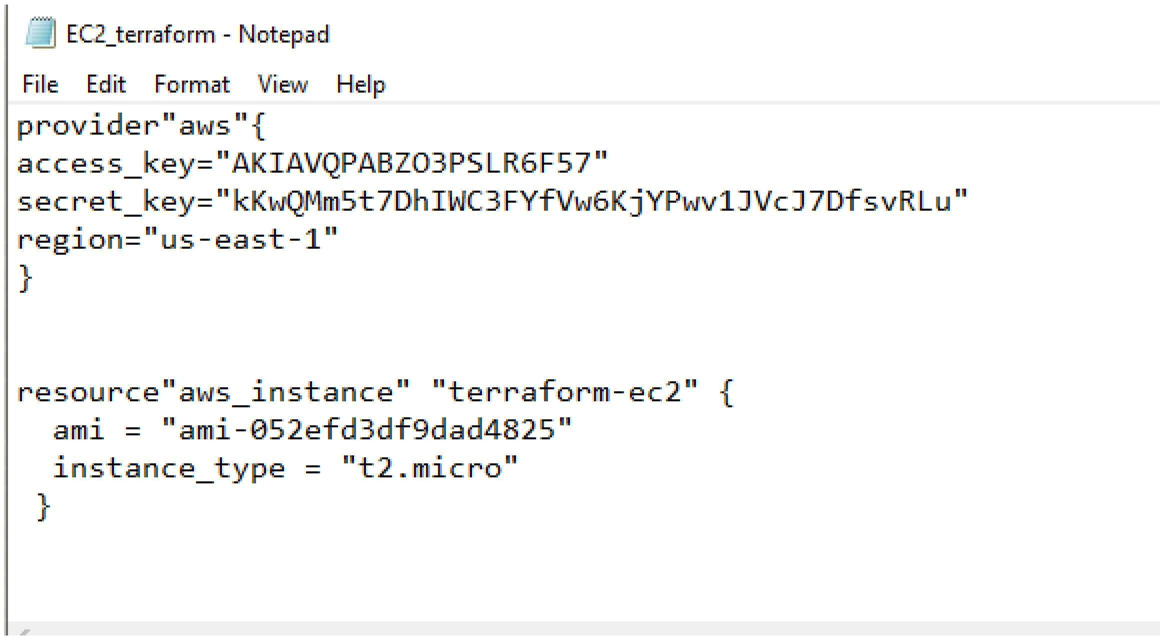
	User	Access key ID	Secret access key
✓	terraform_mayuri	AKIAVQPABZO3PSLR6F57	********* Show

Close

Feedback Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

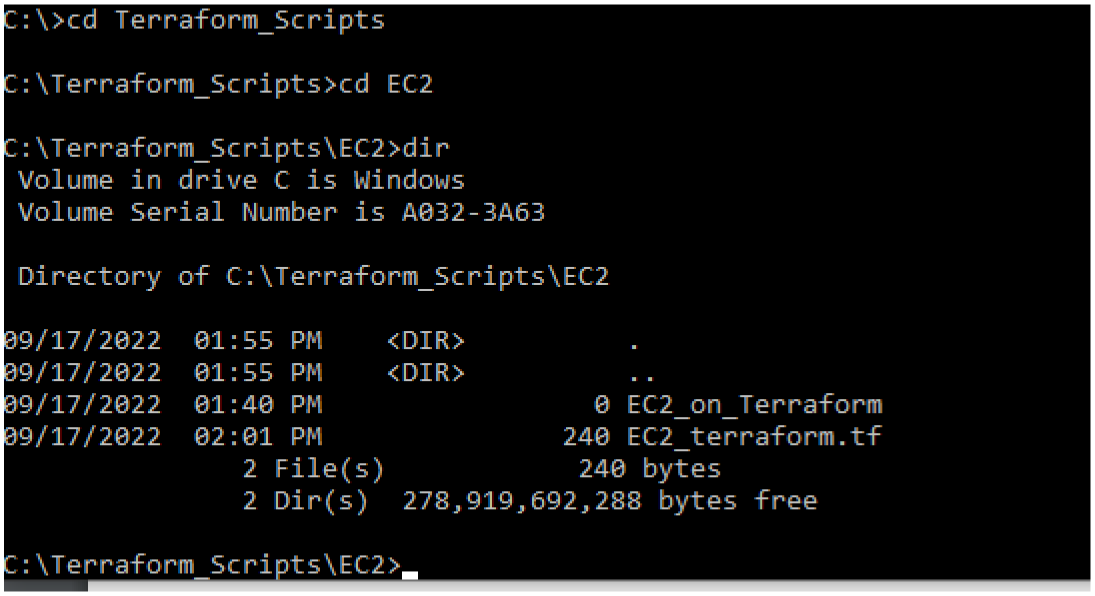
8. Write Terraform Script for creating a EC2 instance using an automated Script and save the file using .tf extension.



```
provider"aws"{
access_key="AKIAVQPABZO3PSLR6F57"
secret_key="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"
region="us-east-1"
}


resource"aws_instance" "terraform-ec2" {
  ami = "ami-052efd3df9dad4825"
  instance_type = "t2.micro"
 }
```

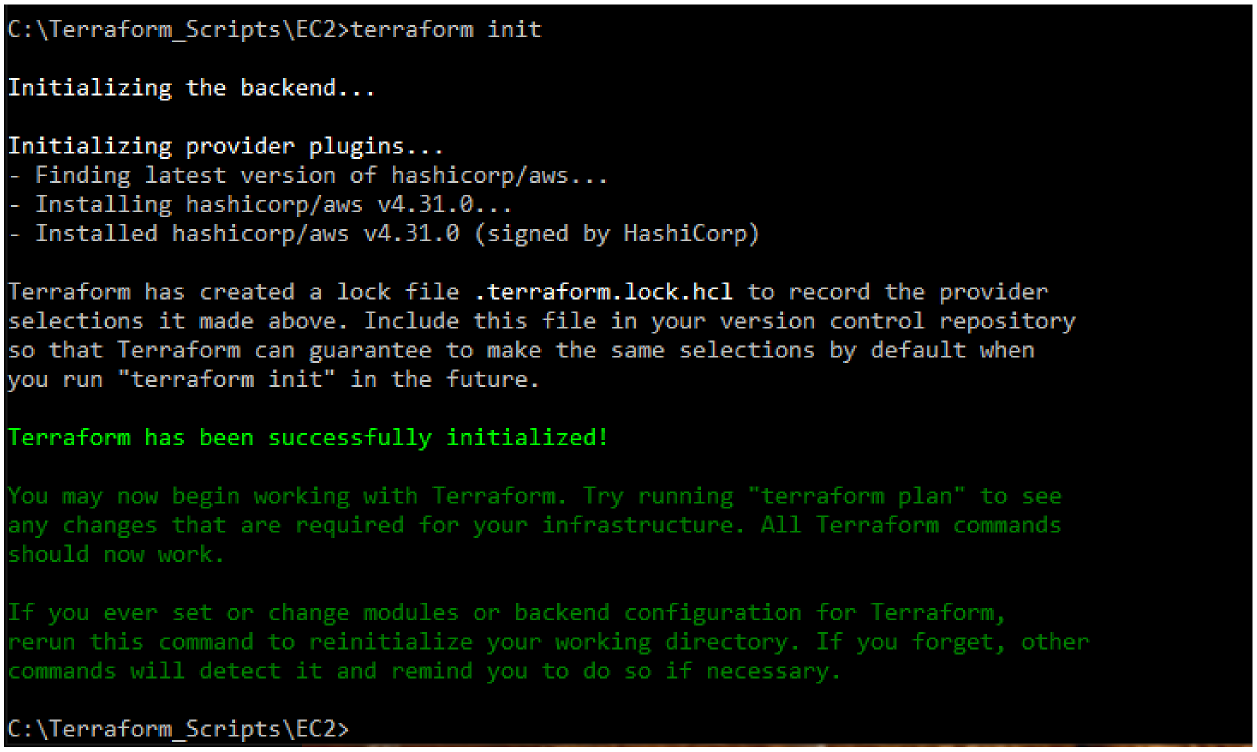
9. AMI stands for Amazon Machine Image which is the id of EC2 Virtual machine instance which can be copied from AWS EC2 service ami = " " To get AMI, First open AWS console and open EC2 service. Click on Launch instance, which will show you list of Operating systems for which EC2 instance to be created. Copy the AMI id of an image for which instance to be created and paste it into our terraform Script. [Note: Ami changes region to region, so see the region before copying AMI which is mentioned in the script, in our example it is us-east-1]

10. Open Command Prompt and go to Terraform_Script directory where our .tf files are stored



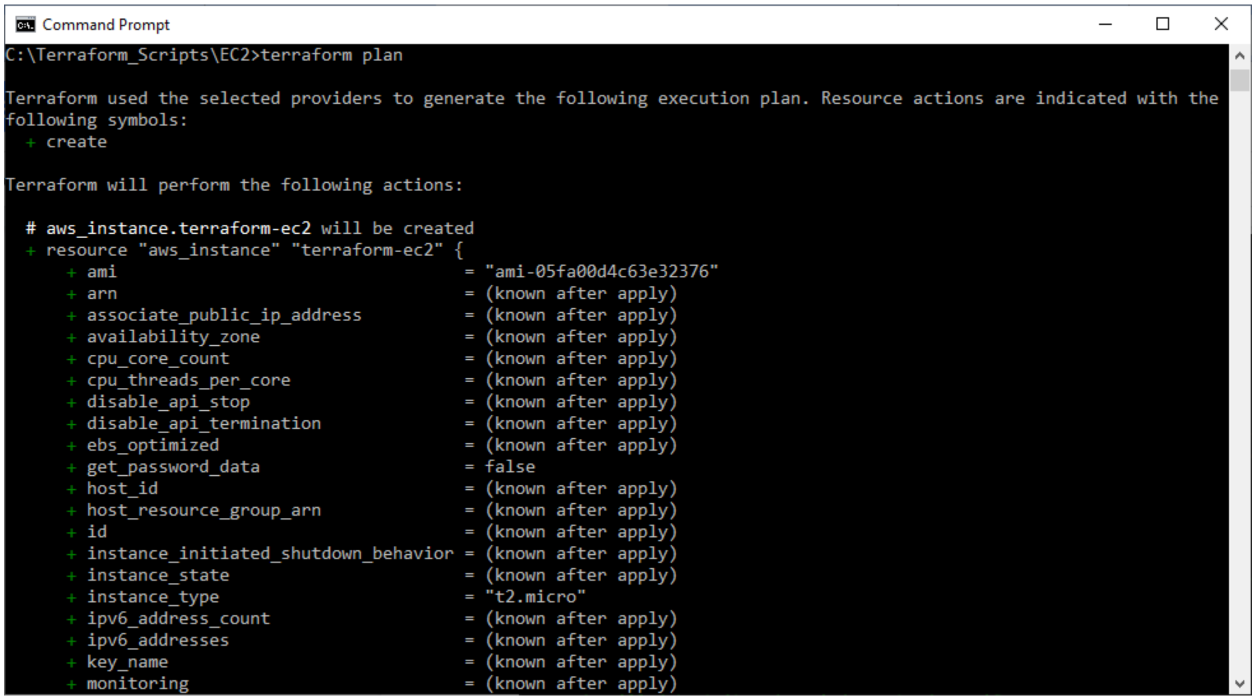
```
C:\>cd Terraform_Scripts

C:\Terraform_Scripts>cd EC2

C:\Terraform_Scripts\EC2>dir
 Volume in drive C is Windows
 Volume Serial Number is A032-3A63

 Directory of C:\Terraform_Scripts\EC2

09/17/2022  01:55 PM    <DIR>          .
09/17/2022  01:55 PM    <DIR>          ..
09/17/2022  01:40 PM                 0 EC2_on_Terraform
09/17/2022  02:01 PM               240 EC2_terraform.tf
               2 File(s)            240 bytes
               2 Dir(s)  278,919,692,288 bytes free

C:\Terraform_Scripts\EC2>
```

11. Execute Terraform Init command to initialize the resources



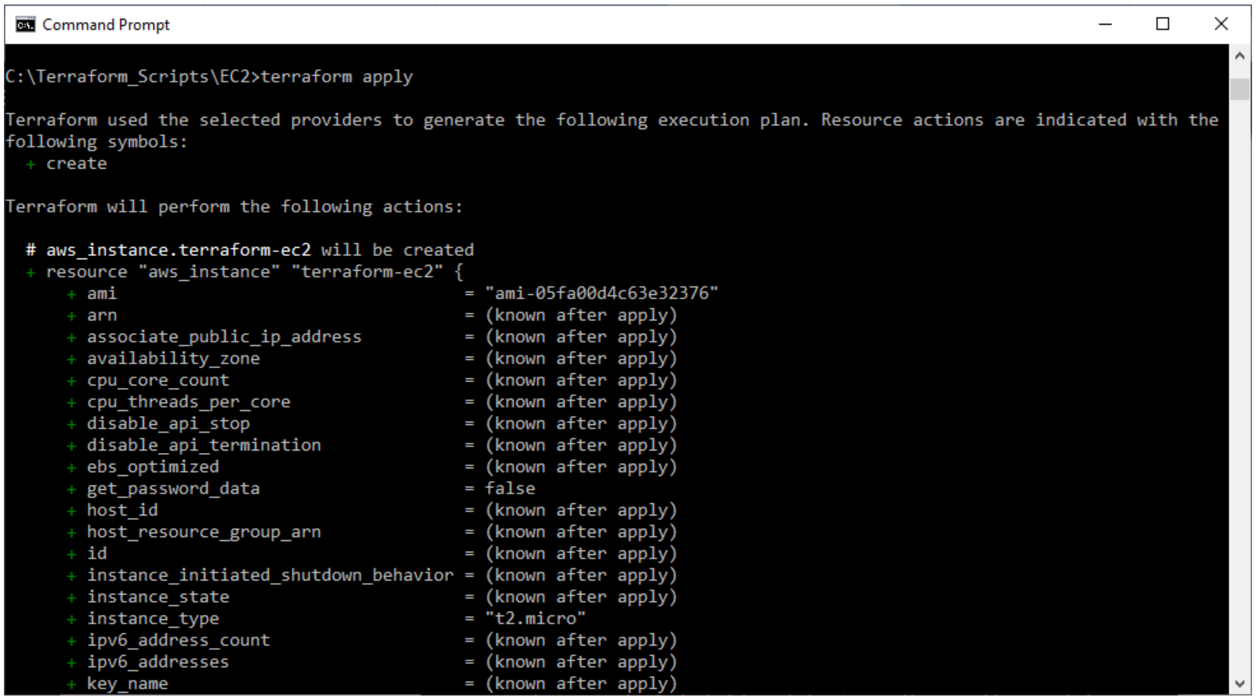
```
C:\Terraform_Scripts\EC2>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.31.0...
- Installed hashicorp/aws v4.31.0 (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:\Terraform_Scripts\EC2>
```

12. Execute Terraform plan to see the available resources

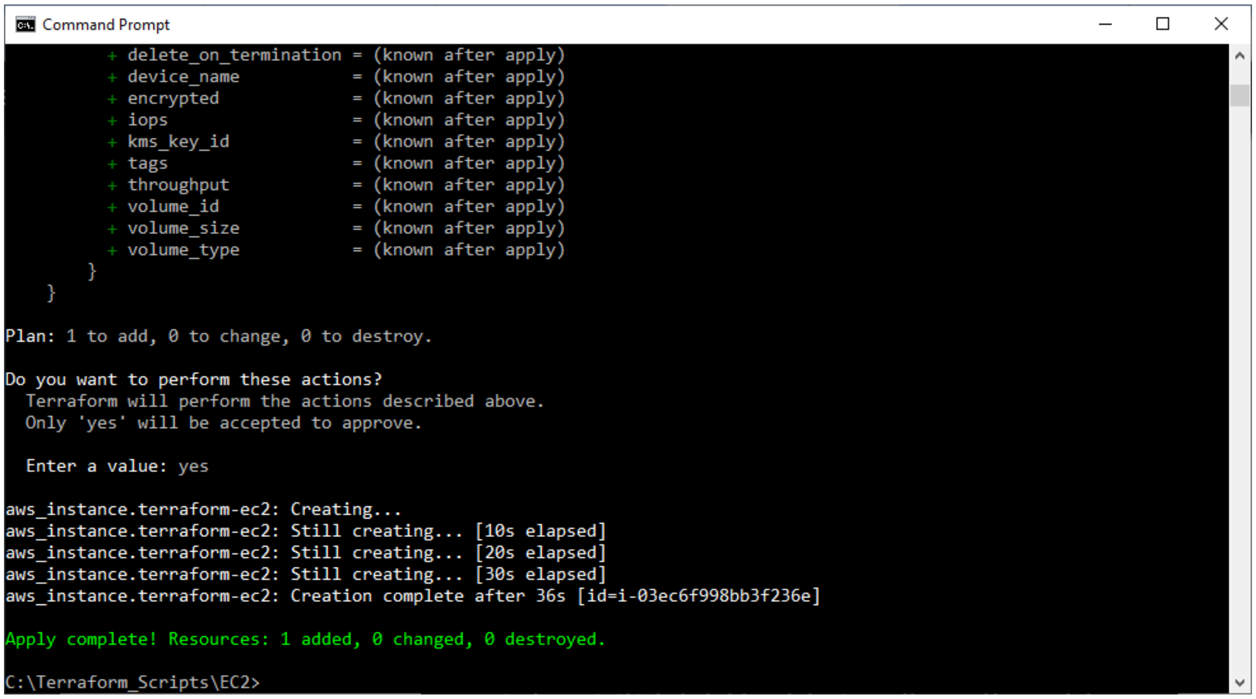


```
C:\Terraform_Scripts\EC2>terraform plan

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.terraform-ec2 will be created
  + resource "aws_instance" "terraform-ec2" {
      + ami                                  = "ami-05fa00d4c63e32376"
      + 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)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (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)
```

13. Execute ‘Terraform apply’ to apply the configuration, which will automatically create an EC2 instance based on our configuration.

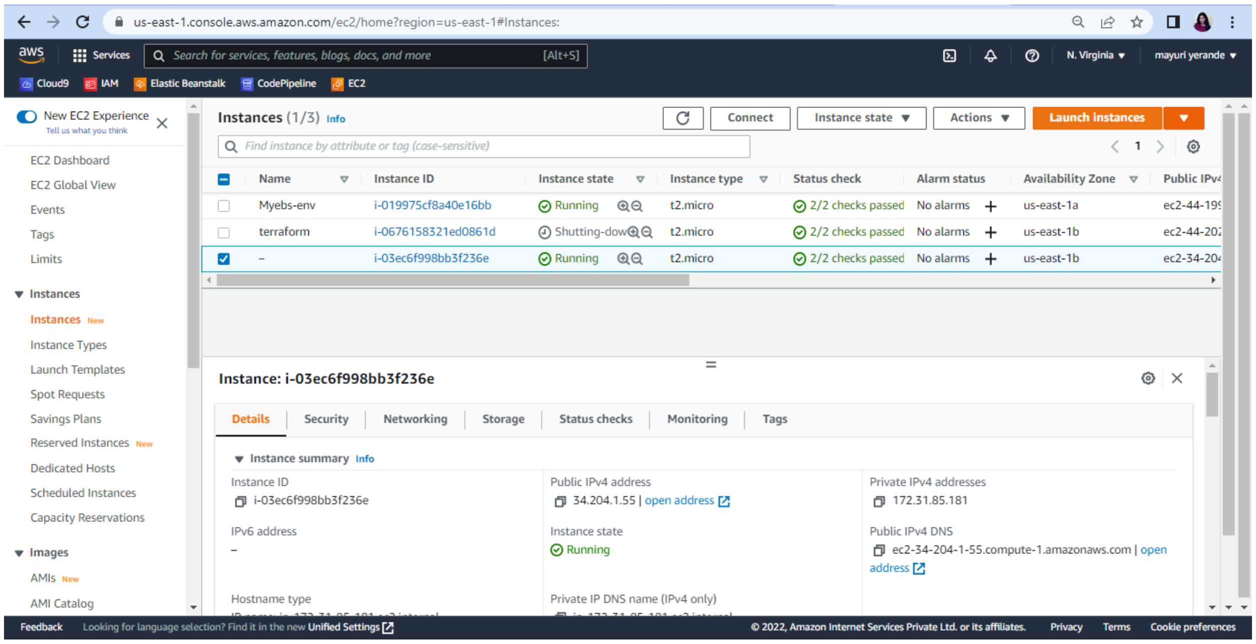


```
C:\Terraform_Scripts\EC2>terraform apply

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.terraform-ec2 will be created
  + resource "aws_instance" "terraform-ec2" {
      + ami                                  = "ami-05fa00d4c63e32376"
      + 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)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (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)
```



```
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (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.terraform-ec2: Creating...
aws_instance.terraform-ec2: Still creating... [10s elapsed]
aws_instance.terraform-ec2: Still creating... [20s elapsed]
aws_instance.terraform-ec2: Still creating... [30s elapsed]
aws_instance.terraform-ec2: Creation complete after 36s [id=i-03ec6f998bb3f236e]

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

C:\Terraform_Scripts\EC2>
```

14. After Creation of instance using Terraform.



Instances (1/3) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
☐	Myebs-env	i-019975cf8a40e16bb	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-44-19
☐	terraform	i-0676158321ed0861d	Shutting-dow	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-44-20
✓	–	i-03ec6f998bb3f236e	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-34-20

Instance: i-03ec6f998bb3f236e

Instance summary Info

Instance ID: i-03ec6f998bb3f236e

Public IPv4 address: 34.204.1.55 | open address

Private IPv4 addresses: 172.31.85.181

IPv6 address: –

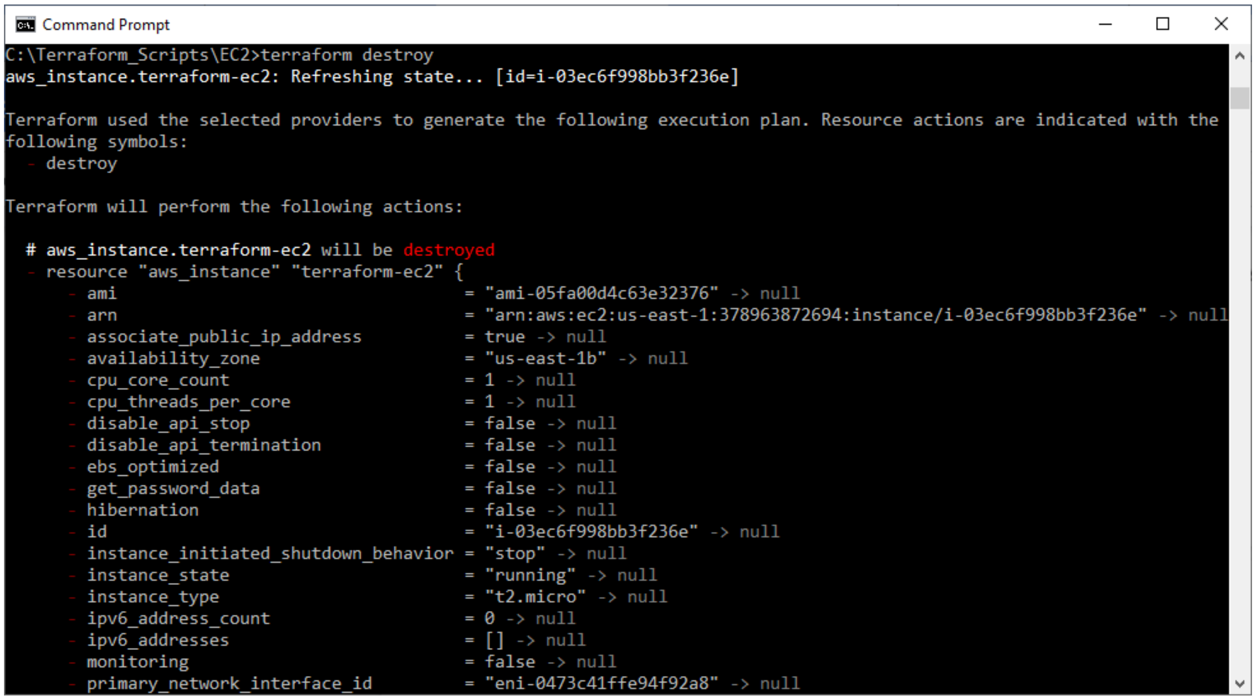
Instance state: Running

Public IPv4 DNS: ec2-34-204-1-55.compute-1.amazonaws.com | open address

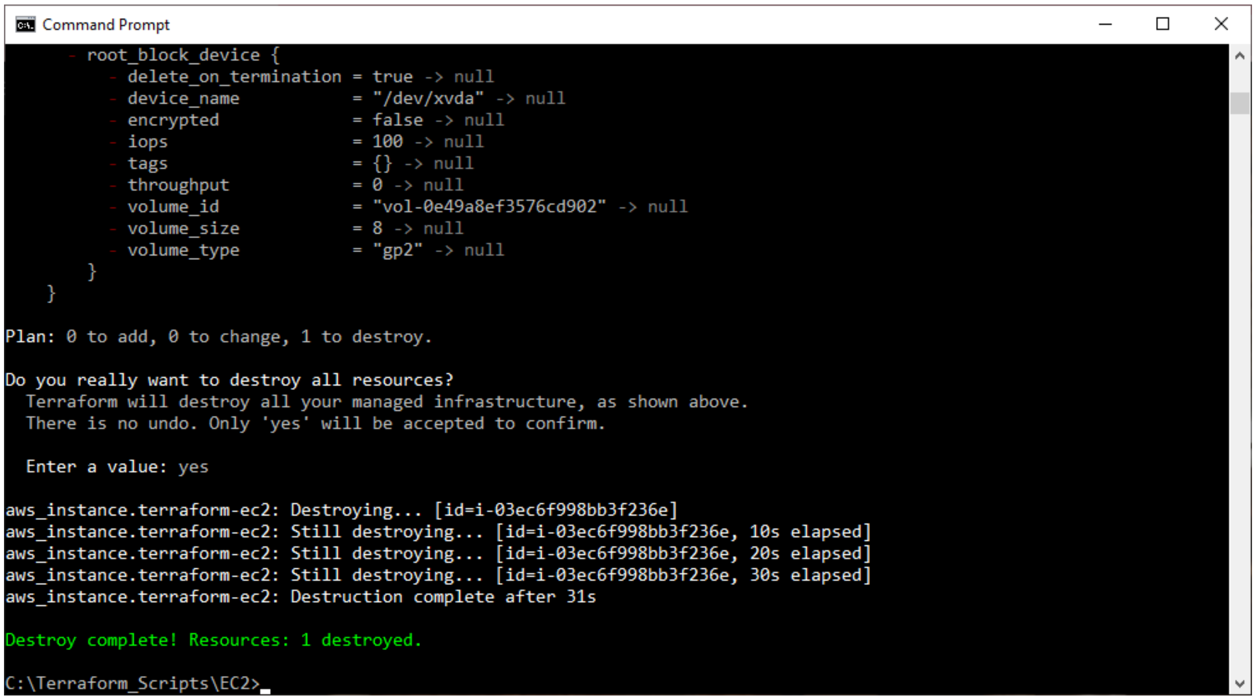
Hostname type

Private IP DNS name (IPv4 only)

15. Execute Terraform destroy to delete the configuration, which will automatically delete an EC2 instance



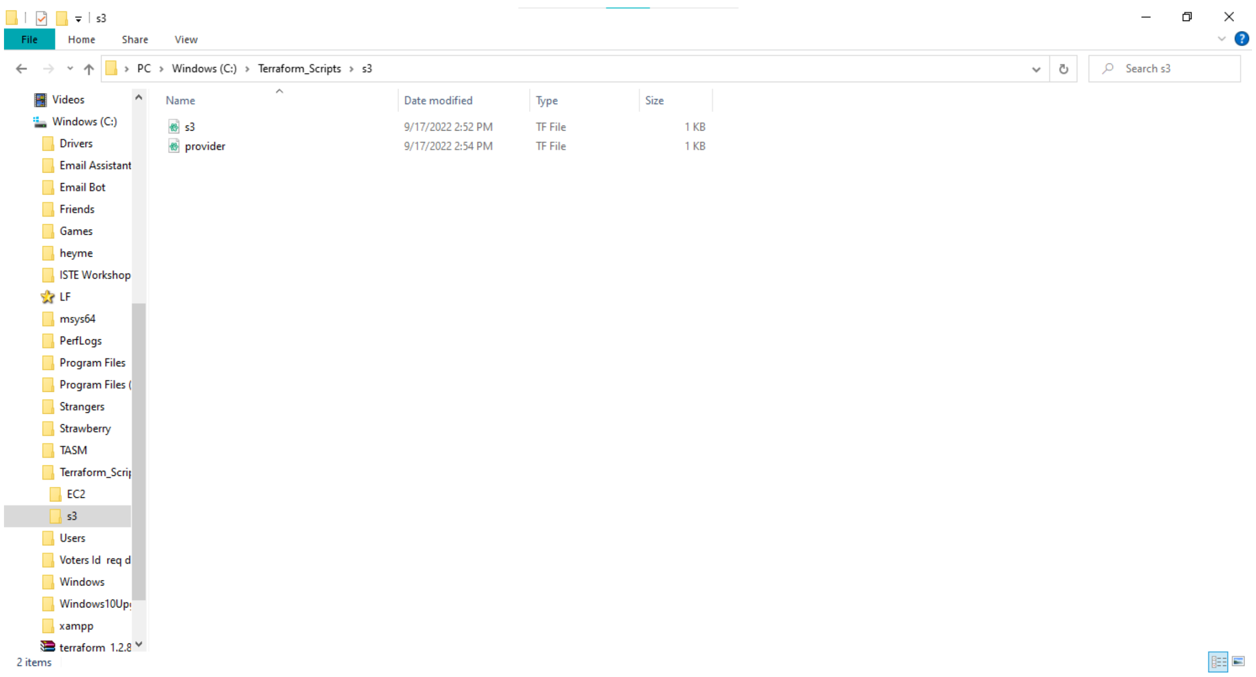
```
C:\Terraform_Scripts\EC2>terraform destroy
aws_instance.terraform-ec2: Refreshing state... [id=i-03ec6f998bb3f236e]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.terraform-ec2 will be destroyed
  - resource "aws_instance" "terraform-ec2" {
      - ami                                  = "ami-05fa00d4c63e32376" -> null
      - arn                                  = "arn:aws:ec2:us-east-1:378963872694:instance/i-03ec6f998bb3f236e" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "us-east-1b" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-03ec6f998bb3f236e" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - primary_network_interface_id         = "eni-0473c41ffe94f92a8" -> null
```



```
      - root_block_device {
          - delete_on_termination = true -> null
          - device_name           = "/dev/xvda" -> null
          - encrypted             = false -> null
          - iops                  = 100 -> null
          - tags                  = {} -> null
          - throughput            = 0 -> null
          - volume_id             = "vol-0e49a8ef3576cd902" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp2" -> null
        }
    }

Plan: 0 to add, 0 to change, 1 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.terraform-ec2: Destroying... [id=i-03ec6f998bb3f236e]
aws_instance.terraform-ec2: Still destroying... [id=i-03ec6f998bb3f236e, 10s elapsed]
aws_instance.terraform-ec2: Still destroying... [id=i-03ec6f998bb3f236e, 20s elapsed]
aws_instance.terraform-ec2: Still destroying... [id=i-03ec6f998bb3f236e, 30s elapsed]
aws_instance.terraform-ec2: Destruction complete after 31s

Destroy complete! Resources: 1 destroyed.

C:\Terraform_Scripts\EC2>
```

S3 bucket

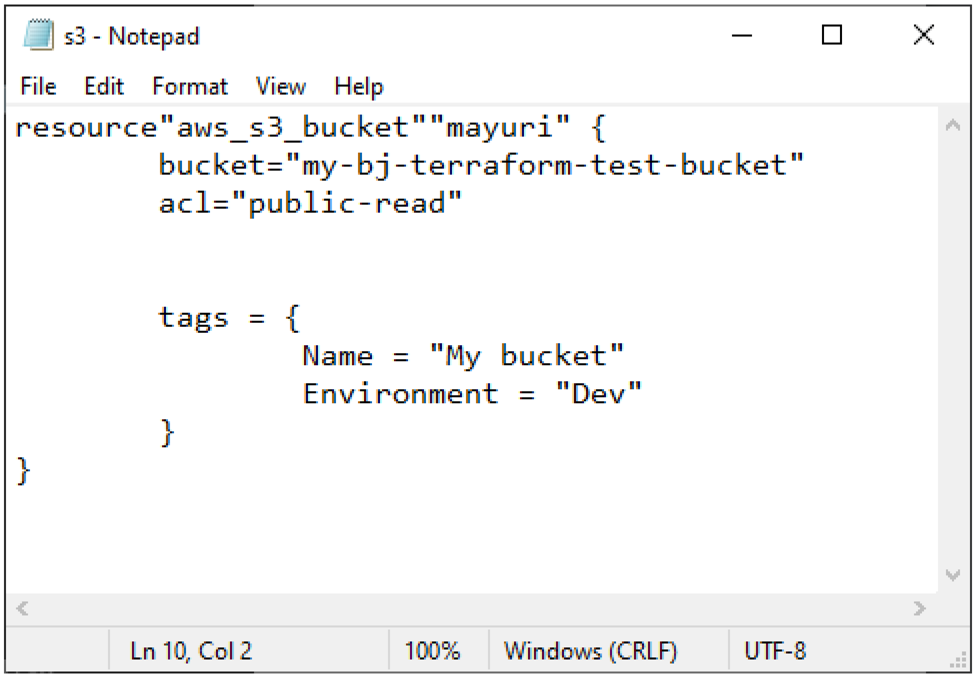
a. Write a Terraform Script in Atom for creating S3 Bucket on Amazon AWS. Create a new provider.tf file and write the following contents into it.



PC › Windows (C:) › Terraform_Scripts › s3

Name	Date modified	Type	Size
s3	9/17/2022 2:52 PM	TF File	1 KB
provider	9/17/2022 2:54 PM	TF File	1 KB

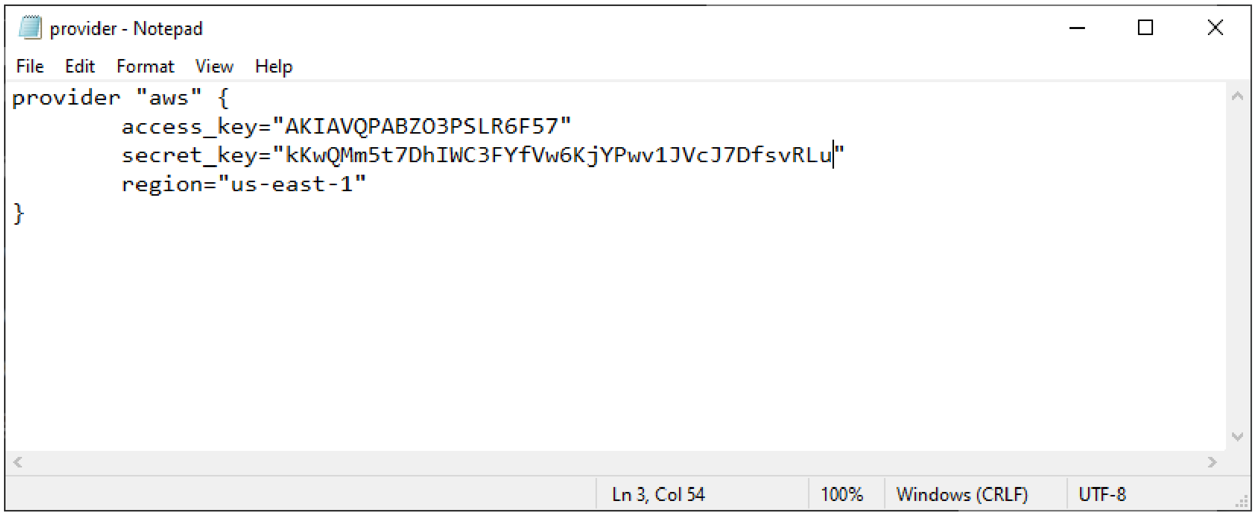
2 items



s3 - Notepad

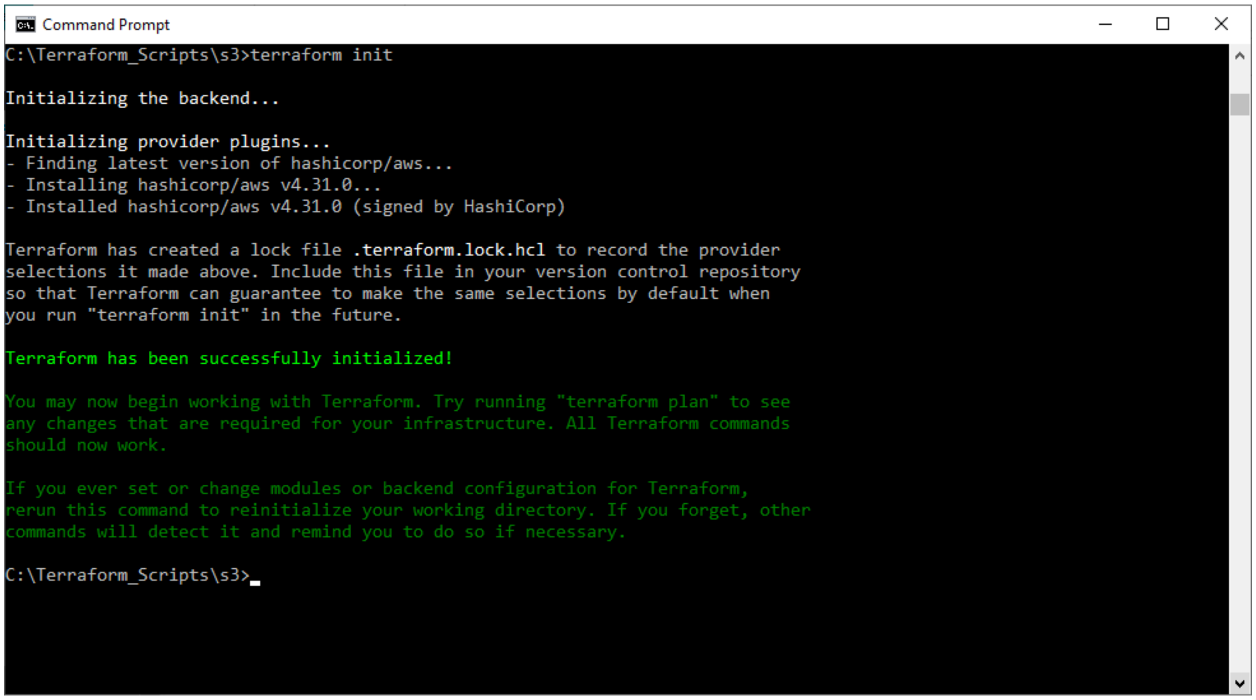
```
resource"aws_s3_bucket""mayuri" {
        bucket="my-bj-terraform-test-bucket"
        acl="public-read"


        tags = {
                Name = "My bucket"
                Environment = "Dev"
        }
}
```

Ln 10, Col 2   100%   Windows (CRLF)   UTF-8



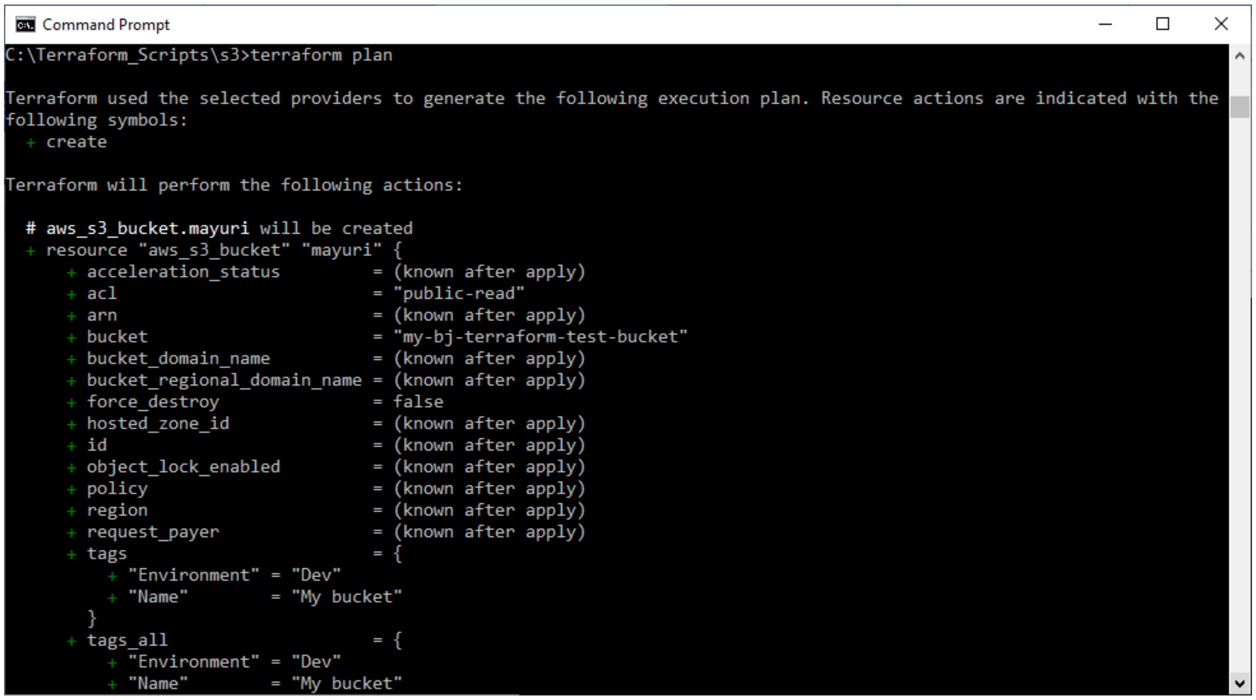
```
provider "aws" {
        access_key="AKIAVQPABZO3PSLR6F57"
        secret_key="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"
        region="us-east-1"
}
```

b. Open Command Prompt and go to Terraform_Script\S3 directory where our .tf files are stored. Execute Terraform Init command to initialize the resources



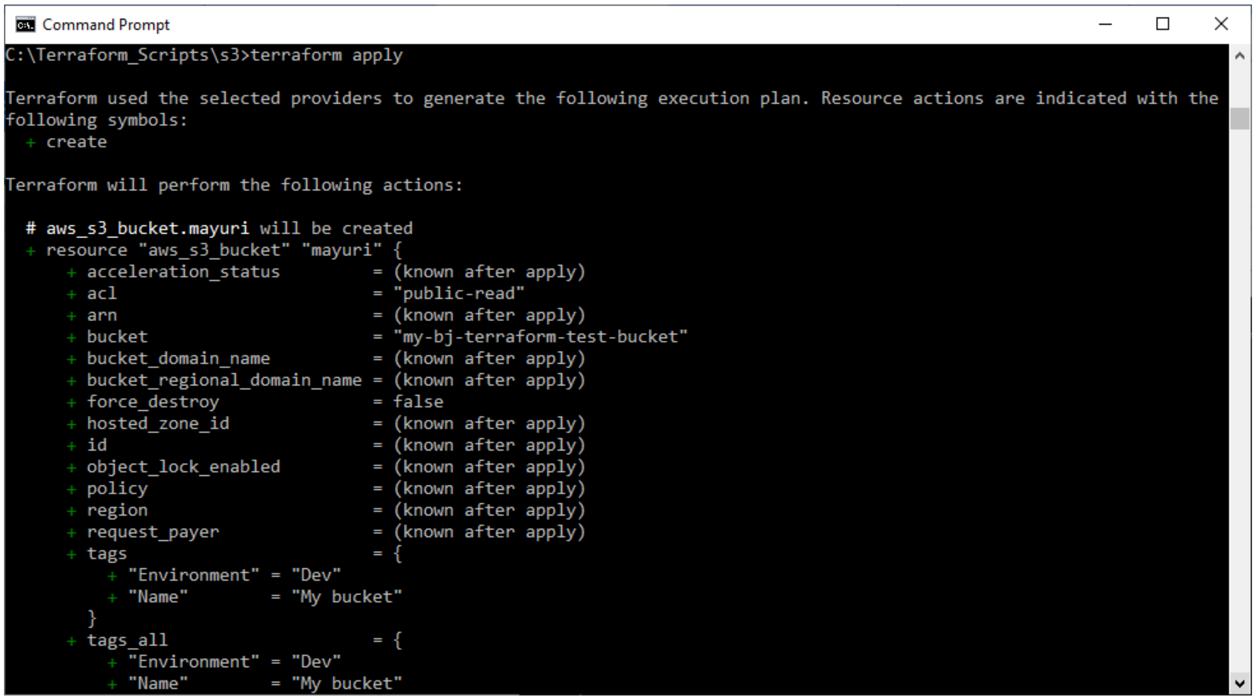
```
C:\Terraform_Scripts\s3>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.31.0...
- Installed hashicorp/aws v4.31.0 (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:\Terraform_Scripts\s3>_
```

c. Execute Terraform plan to see the available resources

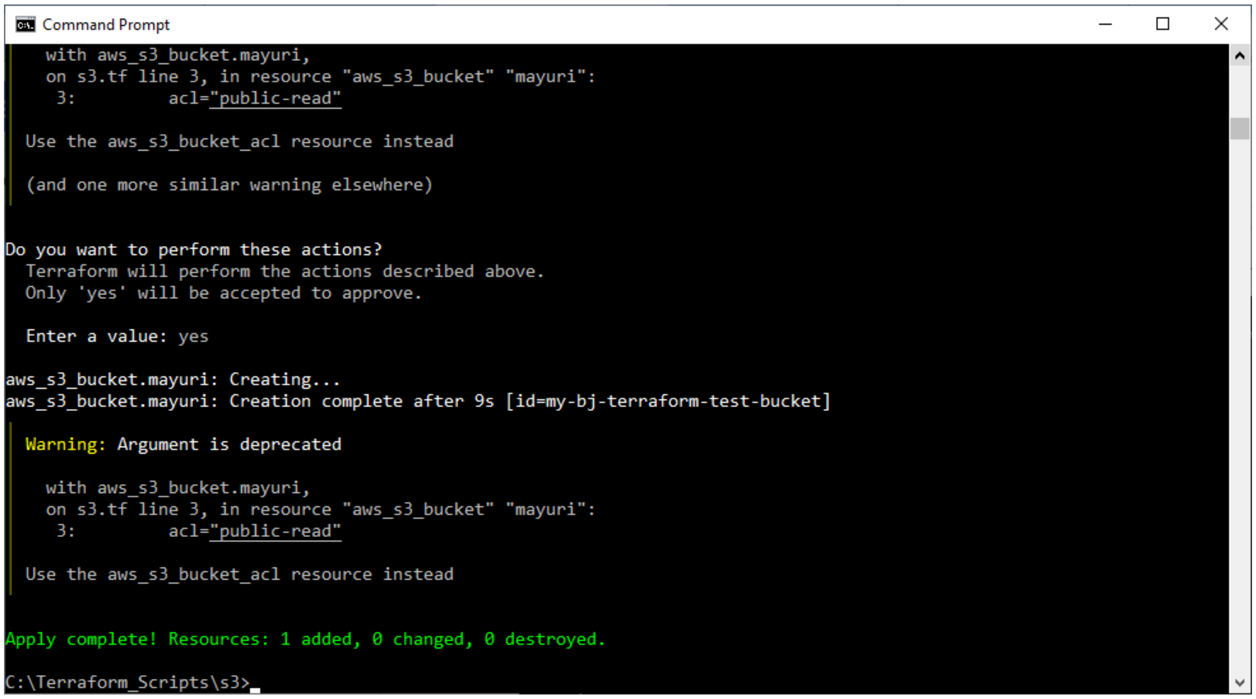

Command Prompt

```
C:\Terraform_Scripts\s3>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_s3_bucket.mayuri will be created
  + resource "aws_s3_bucket" "mayuri" {
      + acceleration_status         = (known after apply)
      + acl                         = "public-read"
      + arn                         = (known after apply)
      + bucket                      = "my-bj-terraform-test-bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
        }
      + tags_all                    = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
```

d. Execute ‘Terraform apply’ to apply the configuration, which will automatically create an S3 bucket based on our configuration.

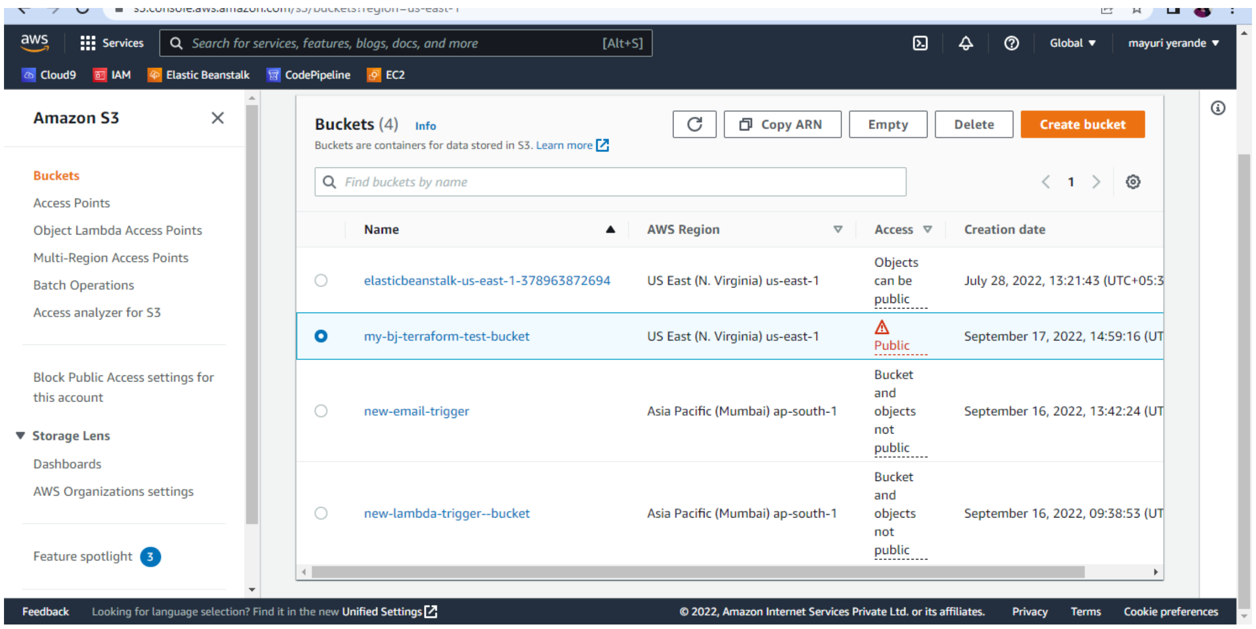

Command Prompt

```
C:\Terraform_Scripts\s3>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_s3_bucket.mayuri will be created
  + resource "aws_s3_bucket" "mayuri" {
      + acceleration_status         = (known after apply)
      + acl                         = "public-read"
      + arn                         = (known after apply)
      + bucket                      = "my-bj-terraform-test-bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
        }
      + tags_all                    = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
```



```
    with aws_s3_bucket.mayuri,
    on s3.tf line 3, in resource "aws_s3_bucket" "mayuri":
     3:          acl="public-read"

  Use the aws_s3_bucket_acl resource instead

  (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_s3_bucket.mayuri: Creating...
aws_s3_bucket.mayuri: Creation complete after 9s [id=my-bj-terraform-test-bucket]

  Warning: Argument is deprecated

    with aws_s3_bucket.mayuri,
    on s3.tf line 3, in resource "aws_s3_bucket" "mayuri":
     3:          acl="public-read"

  Use the aws_s3_bucket_acl resource instead


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

C:\Terraform_Scripts\s3>_
```

e. The bucket has been created.

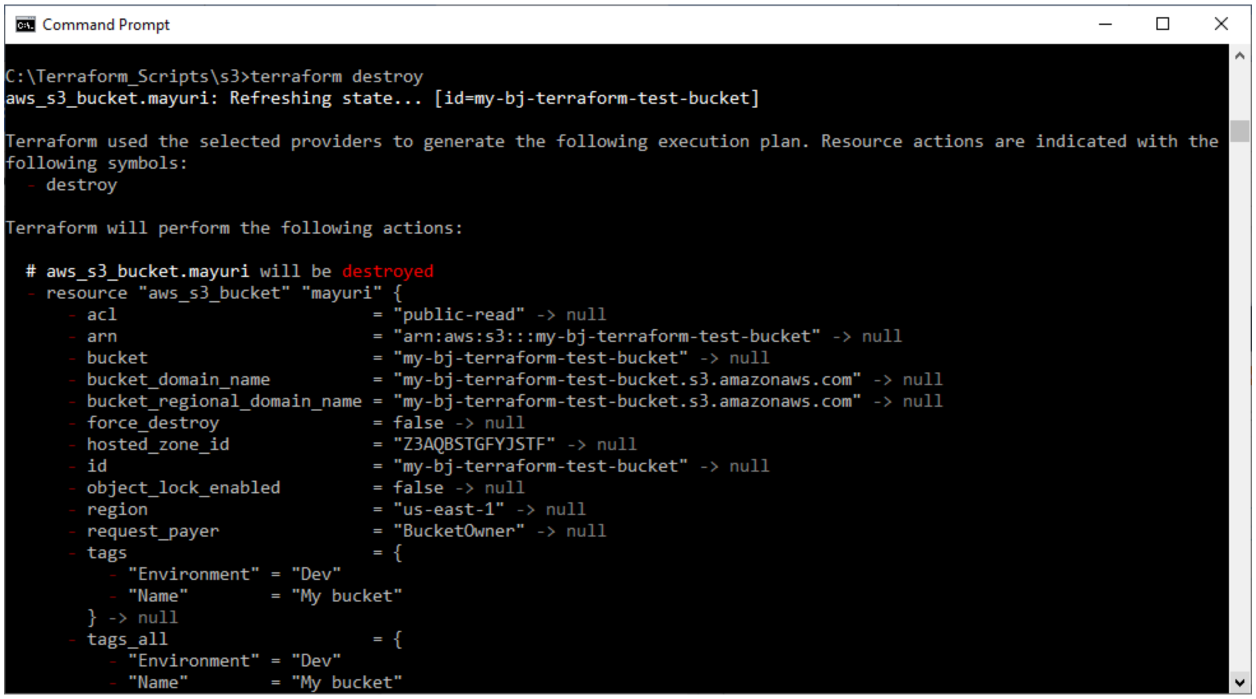

aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Global
mayuri yerande
Cloud9
IAM
Elastic Beanstalk
CodePipeline
EC2
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight 3
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Internet Services Private Ltd. or its affiliates.
Privacy
Terms
Cookie preferences

Buckets (4) Info

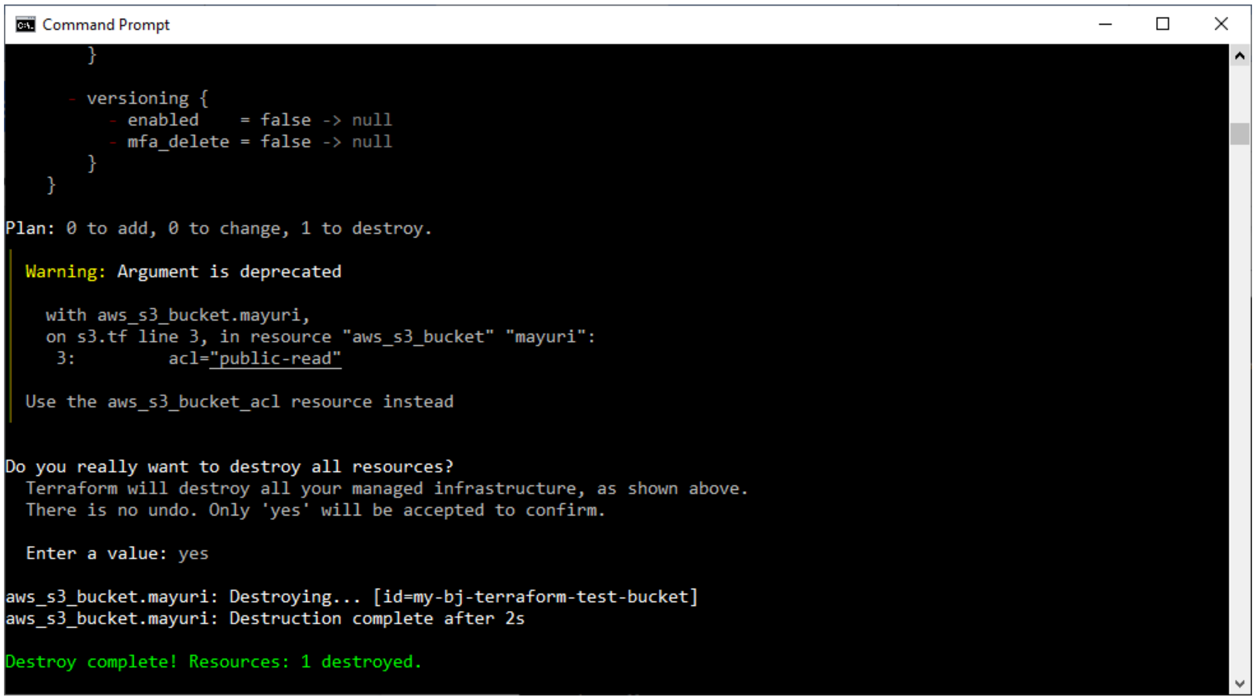
Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
elasticbeanstalk-us-east-1-378963872694	US East (N. Virginia) us-east-1	Objects can be public	July 28, 2022, 13:21:43 (UTC+05:3
my-bj-terraform-test-bucket	US East (N. Virginia) us-east-1	⚠ Public	September 17, 2022, 14:59:16 (UT
new-email-trigger	Asia Pacific (Mumbai) ap-south-1	Bucket and objects not public	September 16, 2022, 13:42:24 (UT
new-lambda-trigger--bucket	Asia Pacific (Mumbai) ap-south-1	Bucket and objects not public	September 16, 2022, 09:38:53 (UT

f. Execute Terraform destroy to delete the configuration, which will automatically delete the recently created Bucket.



```
C:\Terraform_Scripts\s3>terraform destroy
aws_s3_bucket.mayuri: Refreshing state... [id=my-bj-terraform-test-bucket]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_s3_bucket.mayuri will be destroyed
  - resource "aws_s3_bucket" "mayuri" {
      - acl                         = "public-read" -> null
      - arn                         = "arn:aws:s3:::my-bj-terraform-test-bucket" -> null
      - bucket                      = "my-bj-terraform-test-bucket" -> null
      - bucket_domain_name          = "my-bj-terraform-test-bucket.s3.amazonaws.com" -> null
      - bucket_regional_domain_name = "my-bj-terraform-test-bucket.s3.amazonaws.com" -> null
      - force_destroy               = false -> null
      - hosted_zone_id              = "Z3AQBSTGFYJSTF" -> null
      - id                          = "my-bj-terraform-test-bucket" -> null
      - object_lock_enabled         = false -> null
      - region                      = "us-east-1" -> null
      - request_payer               = "BucketOwner" -> null
      - tags                        = {
          - "Environment" = "Dev"
          - "Name"        = "My bucket"
        } -> null
      - tags_all                    = {
          - "Environment" = "Dev"
          - "Name"        = "My bucket"
```

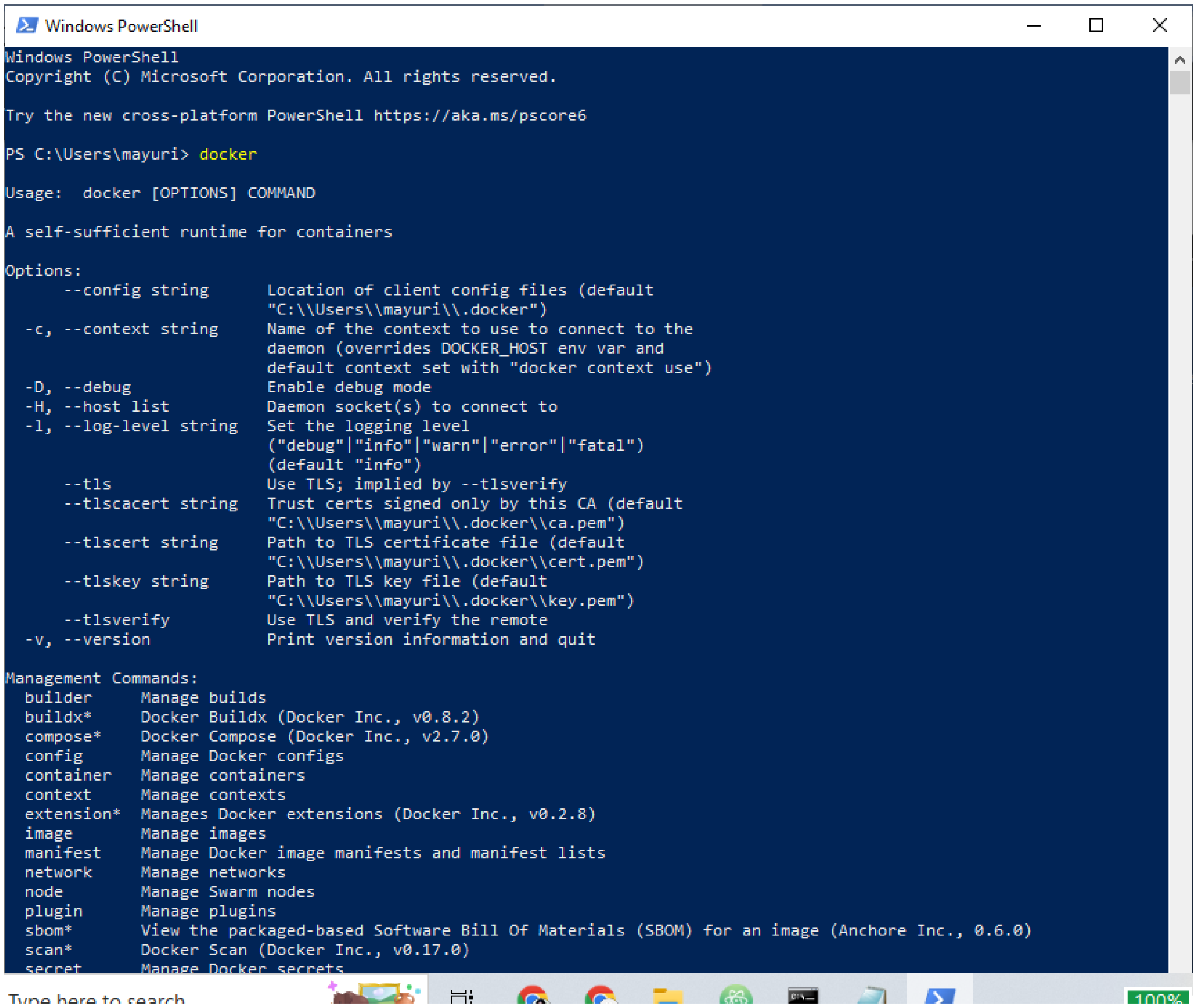


```
        }

      - versioning {
          - enabled    = false -> null
          - mfa_delete = false -> null
        }
    }

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

  Warning: Argument is deprecated

    with aws_s3_bucket.mayuri,
    on s3.tf line 3, in resource "aws_s3_bucket" "mayuri":
     3:         acl="public-read"

  Use the aws_s3_bucket_acl resource instead


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_s3_bucket.mayuri: Destroying... [id=my-bj-terraform-test-bucket]
aws_s3_bucket.mayuri: Destruction complete after 2s

Destroy complete! Resources: 1 destroyed.
```

Docker

- Download and Install Docker Desktop from https://www.docker.com/products/docker-desktop

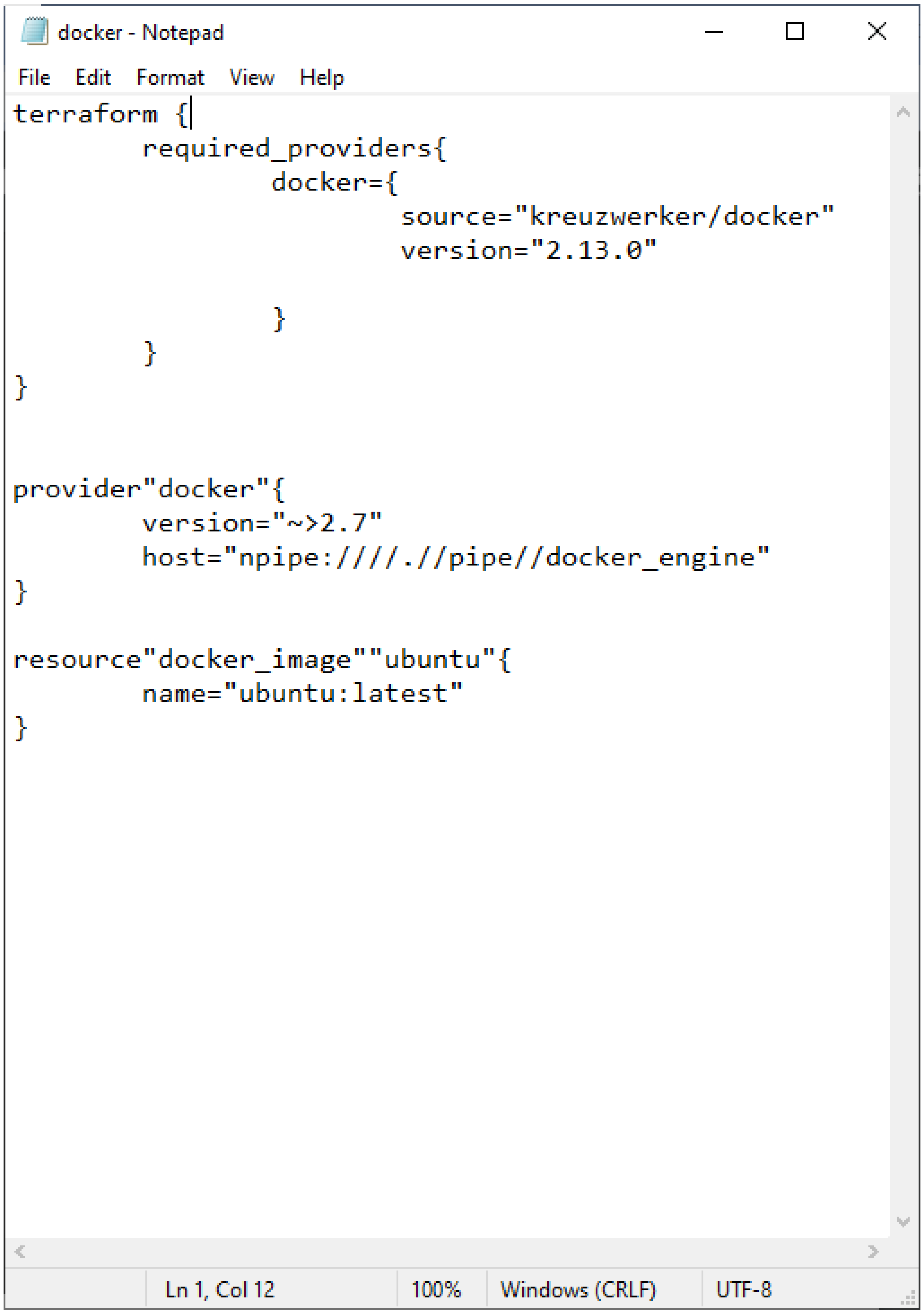
- Check the docker functionality using powershell.



Windows PowerShell

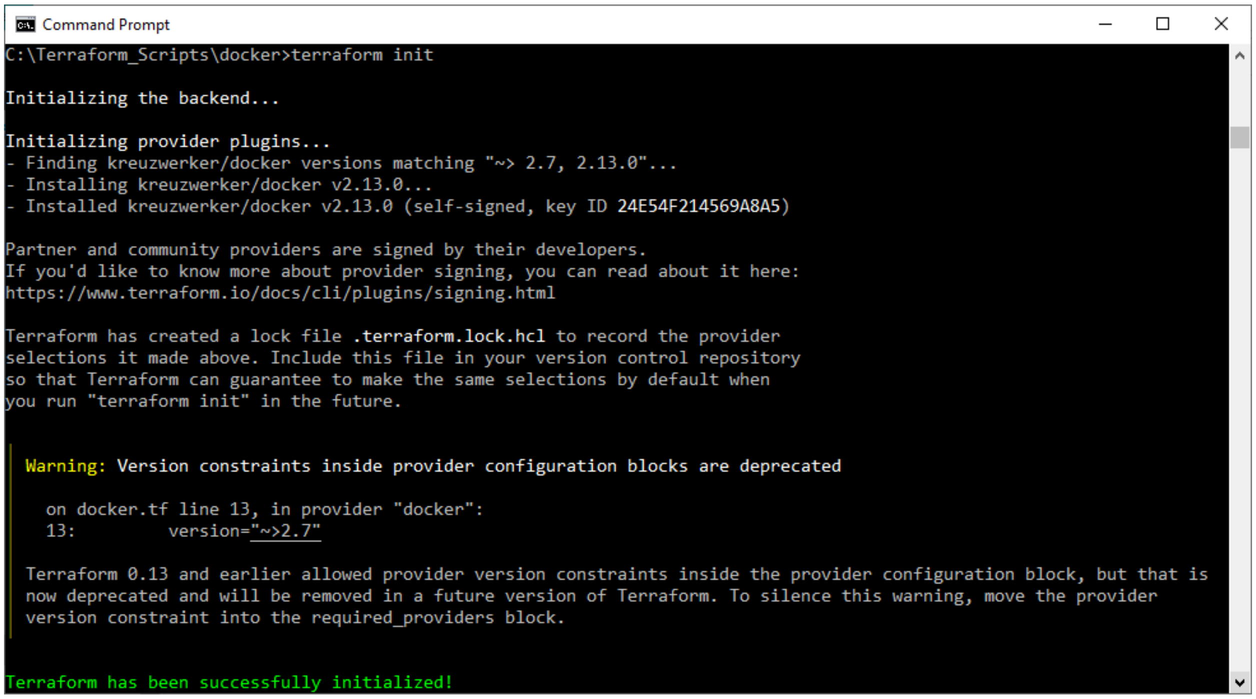
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\mayuri> docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default
                           "C:\\Users\\mayuri\\.docker")
  -c, --context string     Name of the context to use to connect to the
                           daemon (overrides DOCKER_HOST env var and
                           default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
                           ("debug"|"info"|"warn"|"error"|"fatal")
                           (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "C:\\Users\\mayuri\\.docker\\ca.pem")
      --tlscert string     Path to TLS certificate file (default
                           "C:\\Users\\mayuri\\.docker\\cert.pem")
      --tlskey string      Path to TLS key file (default
                           "C:\\Users\\mayuri\\.docker\\key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx (Docker Inc., v0.8.2)
  compose*    Docker Compose (Docker Inc., v2.7.0)
  config      Manage Docker configs
  container   Manage containers
  context     Manage contexts
  extension*  Manages Docker extensions (Docker Inc., v0.2.8)
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc., 0.6.0)
  scan*       Docker Scan (Docker Inc., v0.17.0)
  secret      Manage Docker secrets
```

- Write a terraform script to create a Ubuntu Linux container. Create a new docker.tf file and write the following contents into it. Save the file.



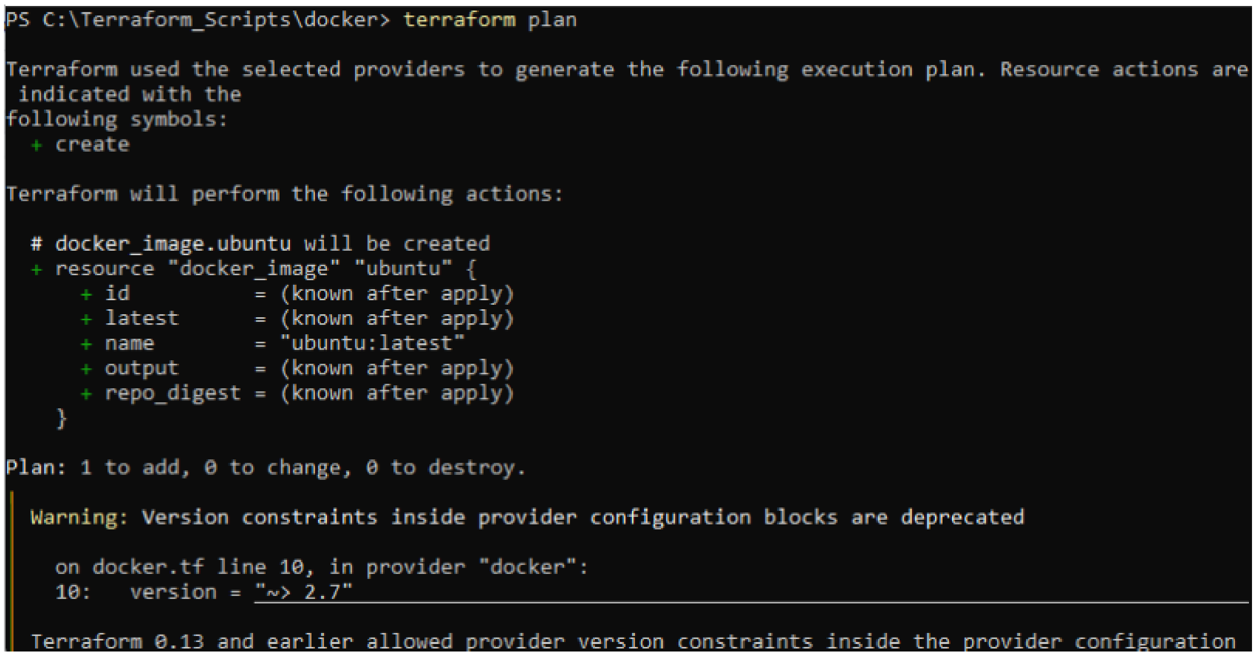
```
terraform {
        required_providers{
                docker={
                        source="kreuzwerker/docker"
                        version="2.13.0"

                }
        }
}


provider"docker"{
        version="~>2.7"
        host="npipe:////.//pipe//docker_engine"
}

resource"docker_image""ubuntu"{
        name="ubuntu:latest"
}
```

● Open Command Prompt and go to Terraform_Script\docker directory where our .tf file is stored. Execute Terraform Init command to initialize the resources.



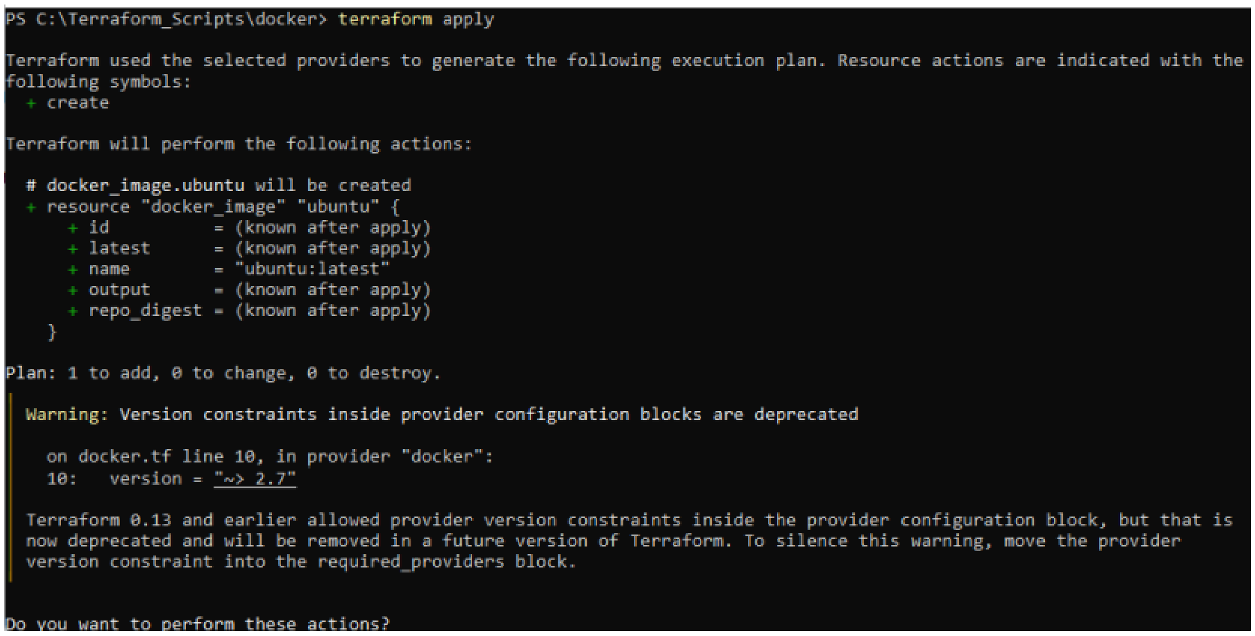
```
C:\Terraform_Scripts\docker>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "~> 2.7, 2.13.0"...
- Installing kreuzwerker/docker v2.13.0...
- Installed kreuzwerker/docker v2.13.0 (self-signed, key ID 24E54F214569A8A5)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

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.

  Warning: Version constraints inside provider configuration blocks are deprecated

    on docker.tf line 13, in provider "docker":
    13:        version="~>2.7"

  Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration block, but that is
  now deprecated and will be removed in a future version of Terraform. To silence this warning, move the provider
  version constraint into the required_providers block.


Terraform has been successfully initialized!
```

● Execute Terraform plan to see the available resources

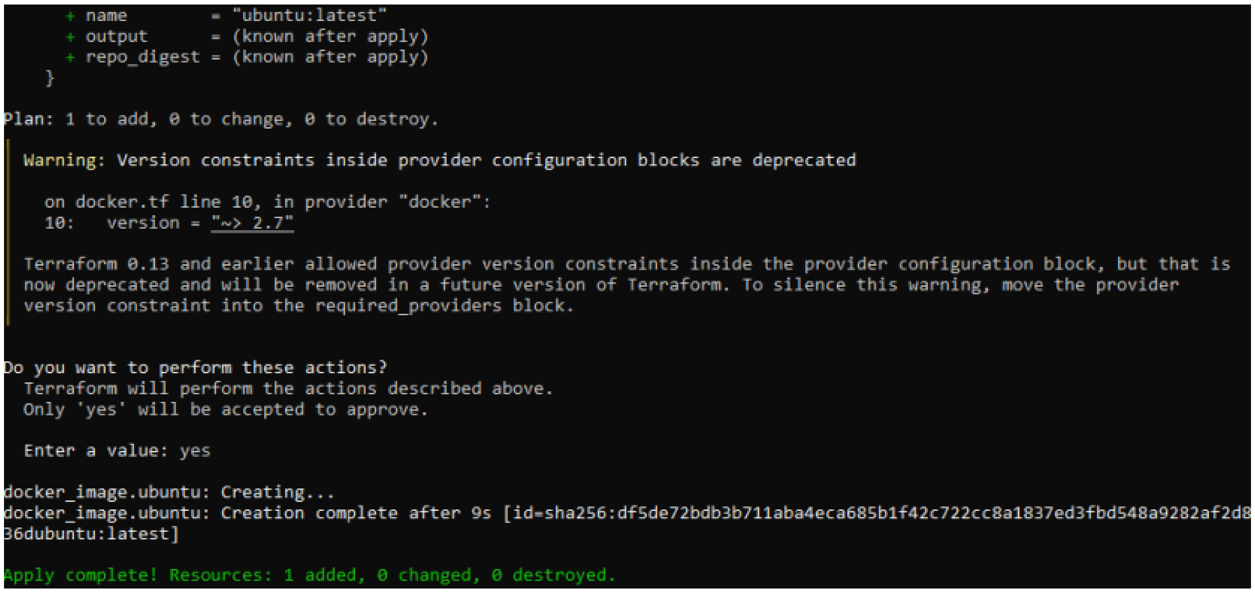


```
PS C:\Terraform_Scripts\docker> terraform plan

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:

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id           = (known after apply)
      + latest       = (known after apply)
      + name         = "ubuntu:latest"
      + output       = (known after apply)
      + repo_digest = (known after apply)
    }

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

  Warning: Version constraints inside provider configuration blocks are deprecated

    on docker.tf line 10, in provider "docker":
    10:   version = "~> 2.7"

  Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration
```

● Execute ‘Terraform apply’ to apply the configuration, which will automatically create and run the ubuntu Linux container based on our configuration.

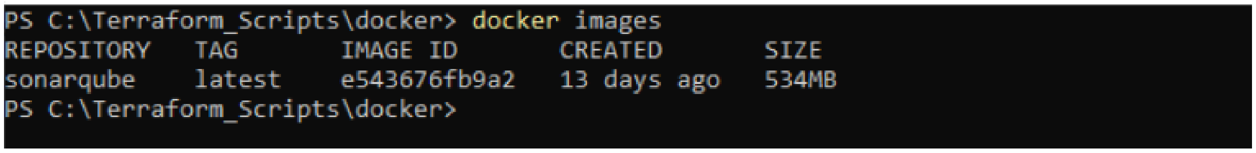


```
PS C:\Terraform_Scripts\docker> terraform apply

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:

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

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

  Warning: Version constraints inside provider configuration blocks are deprecated

    on docker.tf line 10, in provider "docker":
    10:   version = "~> 2.7"

  Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration block, but that is
  now deprecated and will be removed in a future version of Terraform. To silence this warning, move the provider
  version constraint into the required_providers block.


Do you want to perform these actions?
```



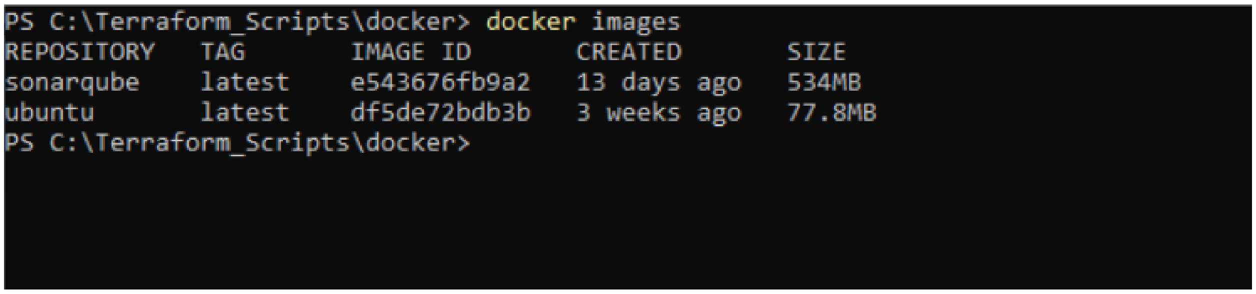
```
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

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

  Warning: Version constraints inside provider configuration blocks are deprecated

    on docker.tf line 10, in provider "docker":
    10:   version = "~> 2.7"

  Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration block, but that is
  now deprecated and will be removed in a future version of Terraform. To silence this warning, move the provider
  version constraint into the required_providers block.


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

docker_image.ubuntu: Creating...
docker_image.ubuntu: Creation complete after 9s [id=sha256:df5de72bdb3b711aba4eca685b1f42c722cc8a1837ed3fbd548a9282af2d8
36dubuntu:latest]

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

- Docker images Before Executing Apply command:



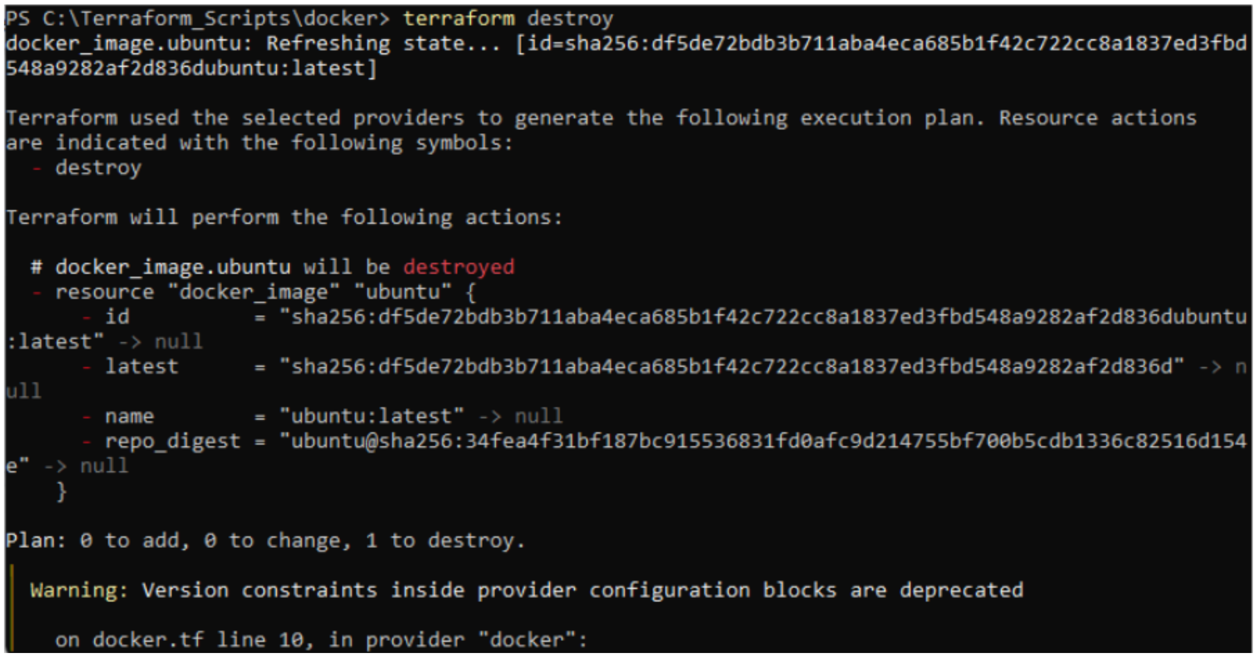
```
PS C:\Terraform_Scripts\docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
sonarqube    latest    e543676fb9a2   13 days ago    534MB
PS C:\Terraform_Scripts\docker>
```

- Docker images, After Executing Apply step:

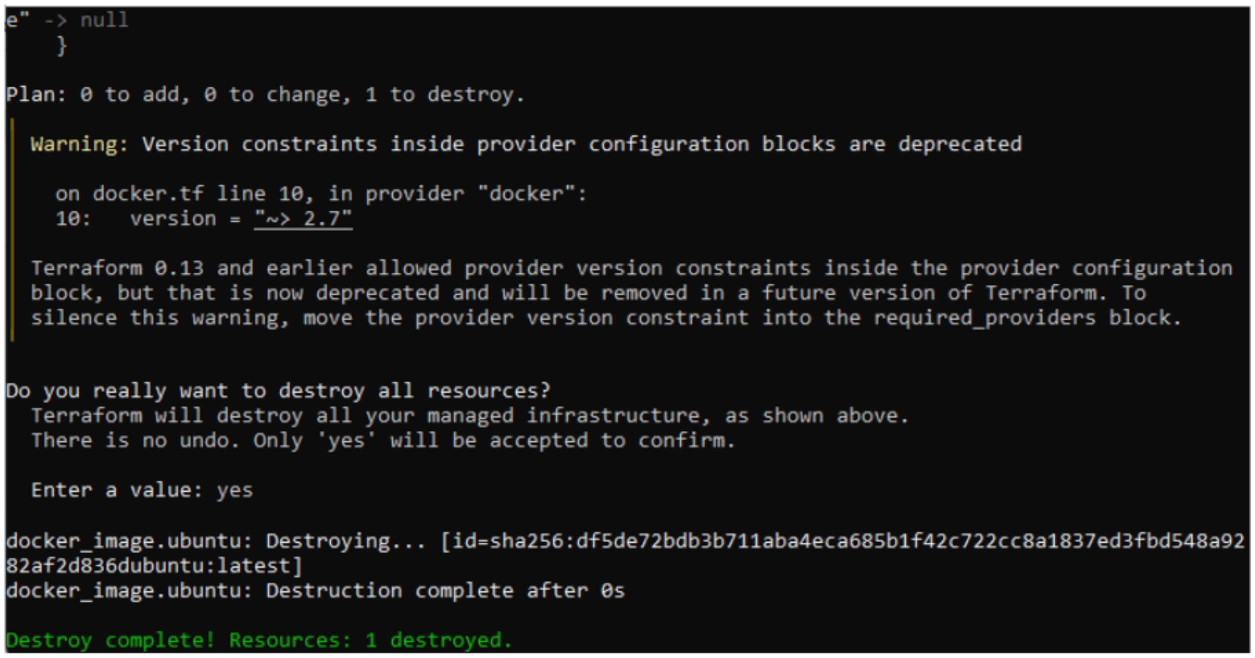


```
PS C:\Terraform_Scripts\docker> docker images
REPOSITORY   TAG      IMAGE ID       CREATED       SIZE
sonarqube    latest   e543676fb9a2   13 days ago   534MB
ubuntu       latest   df5de72bdb3b   3 weeks ago   77.8MB
PS C:\Terraform_Scripts\docker>
```

● Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container

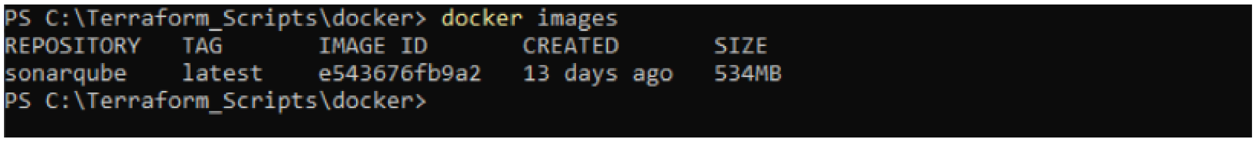


```
PS C:\Terraform_Scripts\docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:df5de72bdb3b711aba4eca685b1f42c722cc8a1837ed3fbd
548a9282af2d836dubuntu:latest]

Terraform used the selected providers to generate the following execution plan. Resource actions
are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # docker_image.ubuntu will be destroyed
  - resource "docker_image" "ubuntu" {
      - id          = "sha256:df5de72bdb3b711aba4eca685b1f42c722cc8a1837ed3fbd548a9282af2d836dubuntu
:latest" -> null
      - latest      = "sha256:df5de72bdb3b711aba4eca685b1f42c722cc8a1837ed3fbd548a9282af2d836d" -> n
ull
      - name        = "ubuntu:latest" -> null
      - repo_digest = "ubuntu@sha256:34fea4f31bf187bc915536831fd0afc9d214755bf700b5cdb1336c82516d154
e" -> null
    }

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

│ Warning: Version constraints inside provider configuration blocks are deprecated
│
│   on docker.tf line 10, in provider "docker":
```



```
e" -> null
    }

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

  Warning: Version constraints inside provider configuration blocks are deprecated

    on docker.tf line 10, in provider "docker":
    10:   version = "~> 2.7"

  Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration
  block, but that is now deprecated and will be removed in a future version of Terraform. To
  silence this warning, move the provider version constraint into the required_providers block.


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

docker_image.ubuntu: Destroying... [id=sha256:df5de72bdb3b711aba4eca685b1f42c722cc8a1837ed3fbd548a92
82af2d836dubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 1 destroyed.
```

● Docker images After Executing Destroy step:



```
PS C:\Terraform_Scripts\docker> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
sonarqube    latest    e543676fb9a2   13 days ago   534MB
PS C:\Terraform_Scripts\docker>
```