ASSIGNMENT NO- 03

AIM: To understand terraform lifecycle, core concepts/terminologies and install it on Linux Machine.

LO MAPPED: LO1LO5

THEORY:

Terraform is an open-source IaC software tool that provides a consistent command line interface (CLI) workflow to manage hundreds of cloud services. Terraform codifies cloud APIs into declarative configuration files.

For deployment with Terraform, use the same principles used in CDK. The code is structured in modules that allow the networking components to be customized and reused according to the vendor requirements.

The configuration is all parameterized, which allows the deployments to be fully tailored according to providers and ISV recommendations.

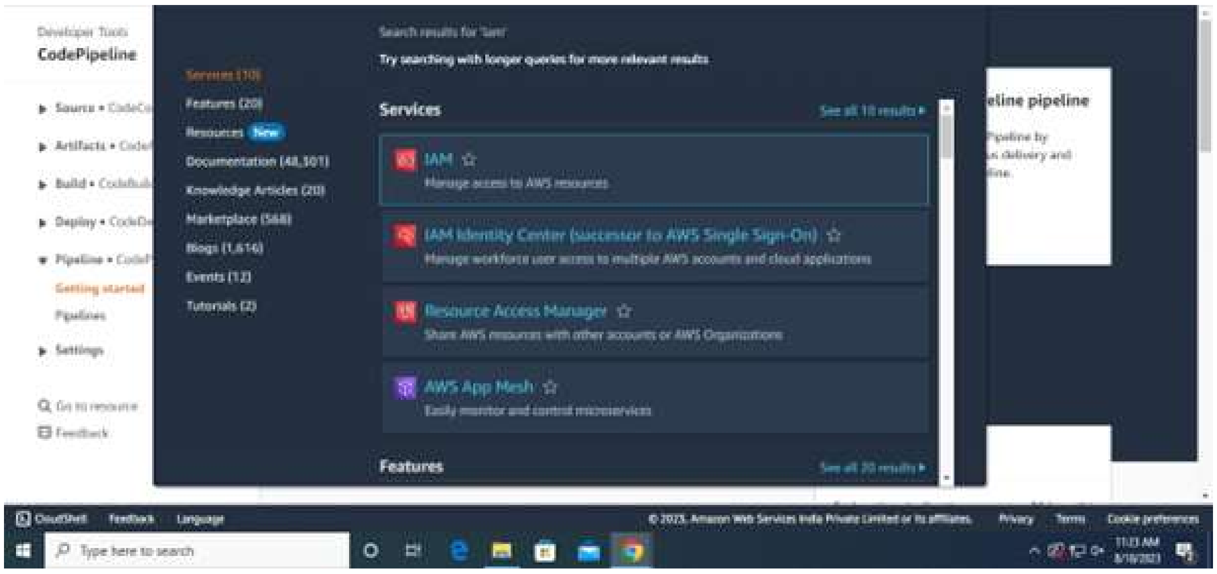
The network functions deployment is separated in two phases:

- The required AWS infrastructure is created and managed via a central repository.
- The configuration and code is centrally stored in a GitHub repository.

After the prerequisites are created, the network function is ready to be deployed by using an application pipeline that was set in the previous stage.

STEPS:

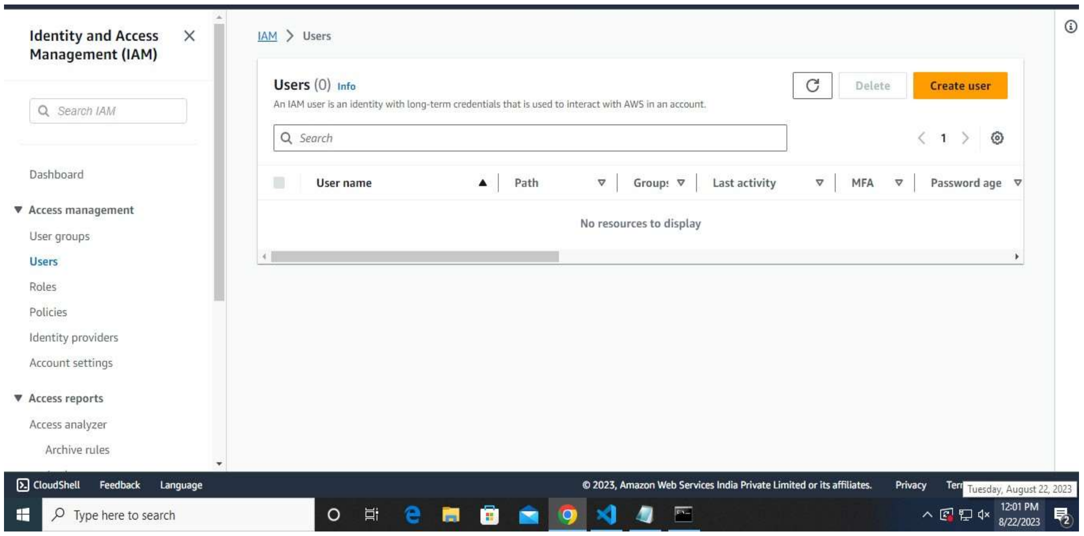
1. Search for ‘IAM’ on the AWS Console.



Search results for 'iam'

Try searching with longer queries for more relevant results

2. Go to 'Users' and click on CREATE USERS



Users (0) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

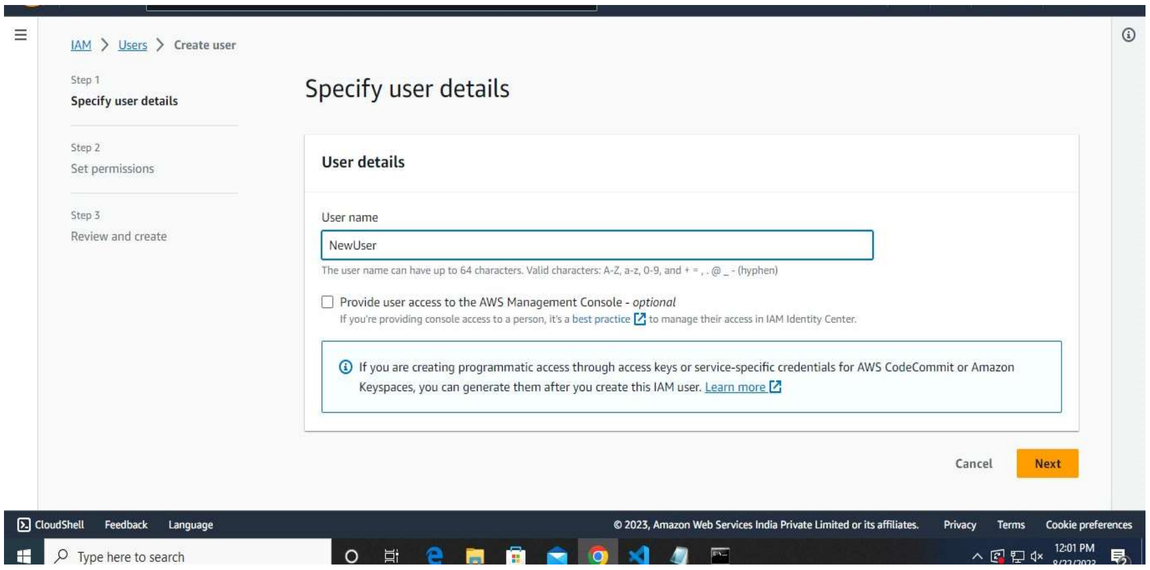
User name	Path	Groups	Last activity	MFA	Password age

No resources to display

© 2023, Amazon Web Services India Private Limited or its affiliates.

Tuesday, August 22, 2023 12:01 PM 8/22/2023

3. Enter the name of new user and click on next.



Specify user details

User details

User name: NewUser

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*

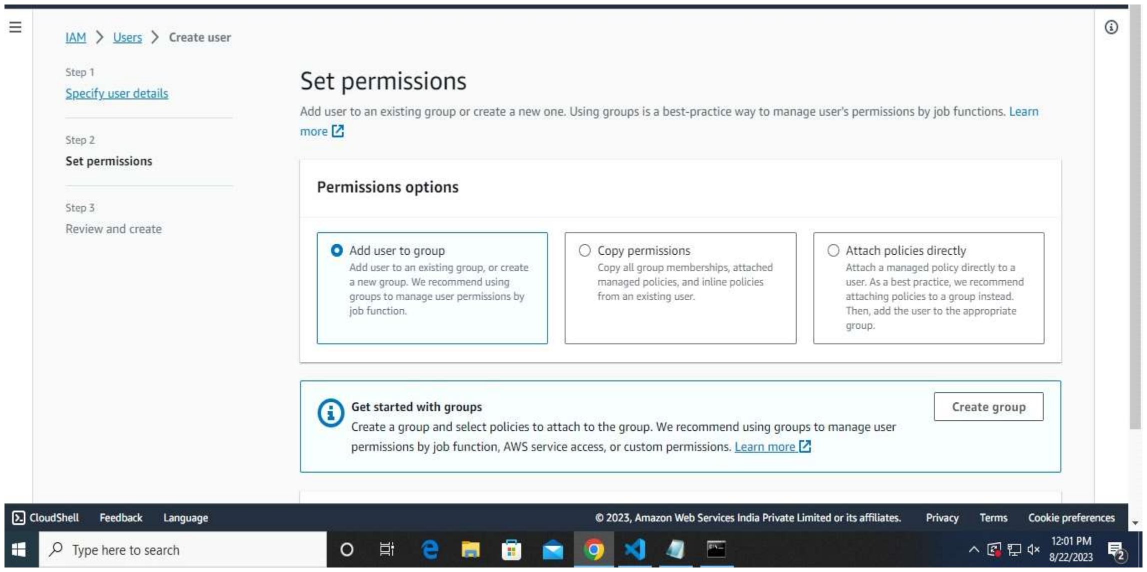
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more

© 2023, Amazon Web Services India Private Limited or its affiliates.

12:01 PM

4. Select the option 'Add user to group' and click on CREATE GROUP



Set permissions

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

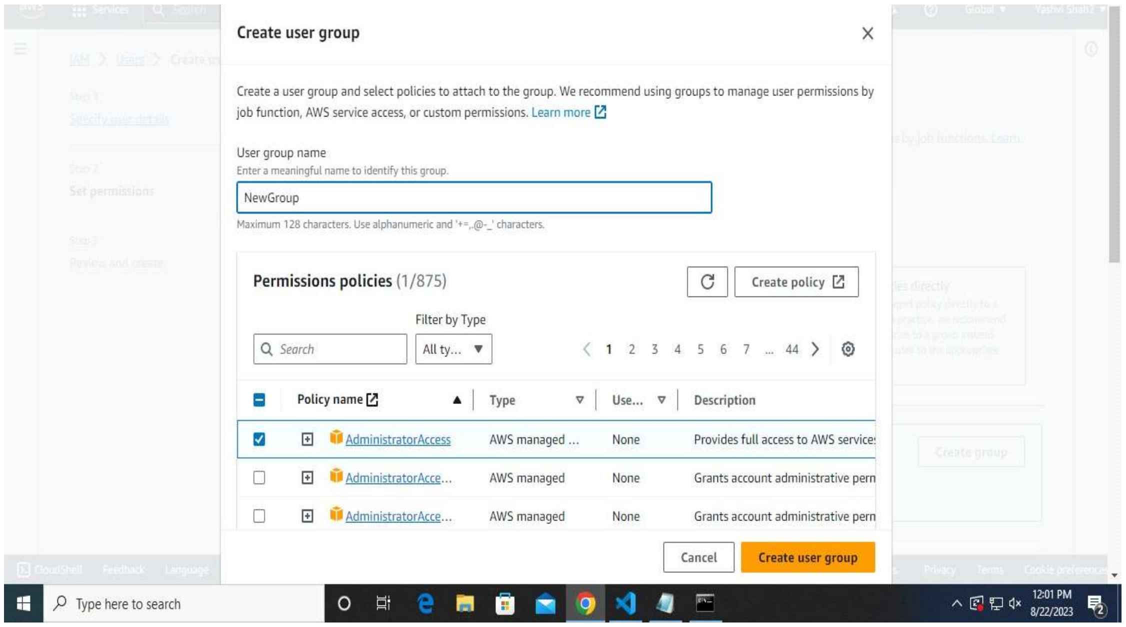
Permissions options

- [✓ Add user to group] Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- Copy permissions: Copy all group memberships, attached managed policies, and inline policies from an existing user.
- Attach policies directly: Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Get started with groups
Create a group and select policies to attach to the group. We recommend using groups to manage user permissions by job function, AWS service access, or custom permissions. Learn more

Create group
CloudShell Feedback Language
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
Type here to search
12:01 PM
8/22/2023

5. Give a name to new user group and give 'Administrative Access'. Click on CREATE USER GROUP.



Create user group

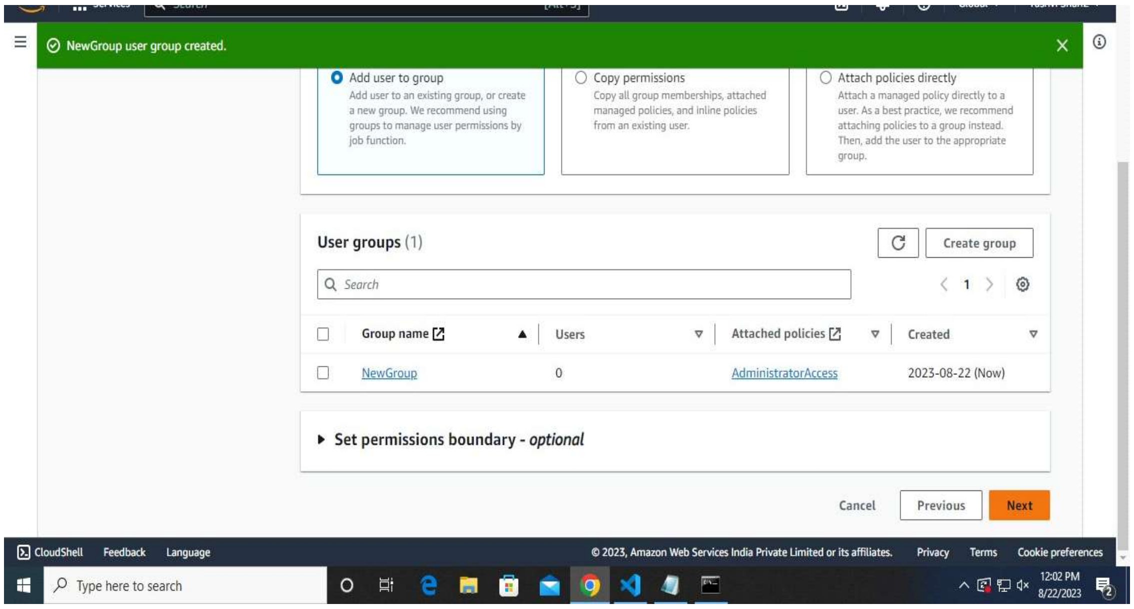
Create a user group and select policies to attach to the group. We recommend using groups to manage user permissions by job function, AWS service access, or custom permissions. Learn more

User group name
Enter a meaningful name to identify this group.
NewGroup
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.

Permissions policies (1/875)

	Policy name	Type	Use...	Description
✓	AdministratorAccess	AWS managed ...	None	Provides full access to AWS service
	AdministratorAcce...	AWS managed	None	Grants account administrative pern
	AdministratorAcce...	AWS managed	None	Grants account administrative pern

6. Now open the new group you created and click on CREATE USER.



NewGroup user group created.

Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

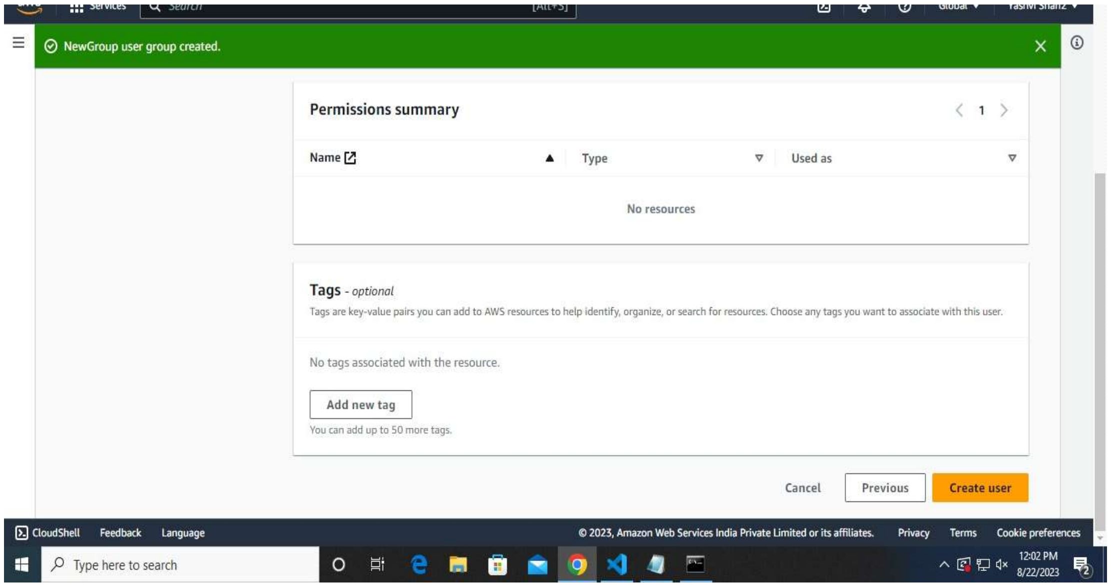
Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

User groups (1)

	Group name	Users	Attached policies	Created
☐	NewGroup	0	AdministratorAccess	2023-08-22 (Now)

Set permissions boundary - *optional*



NewGroup user group created.

Permissions summary

Name	Type	Used as
No resources		

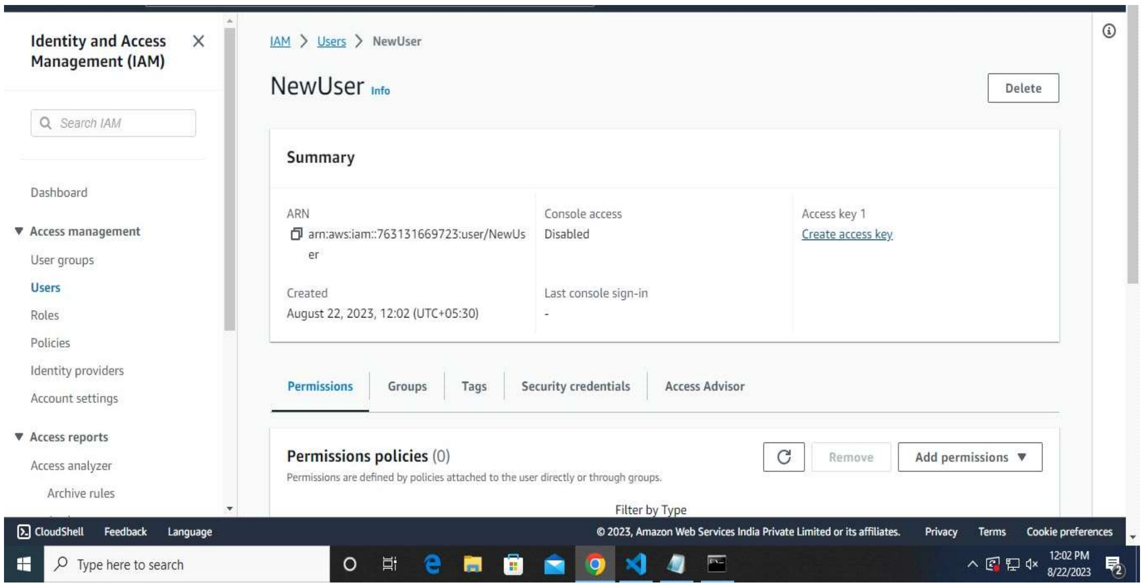
Tags - *optional*
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

7. Click on ‘Create Access Key’ and then select the ‘Command Line Interface’ option.



Identity and Access Management (IAM)

IAM > Users > NewUser

NewUser Info

Summary

ARN
arn:aws:iam::763131669723:user/NewUser

Console access
Disabled

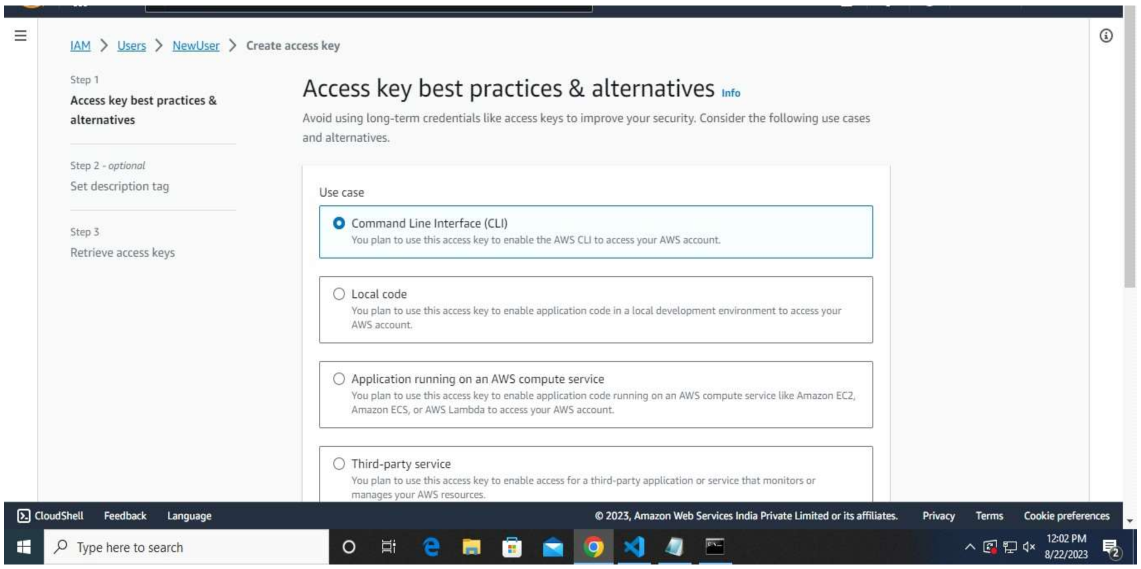
Access key 1
Create access key

Created
August 22, 2023, 12:02 (UTC+05:30)

Last console sign-in
-

Permissions policies (0)
Permissions are defined by policies attached to the user directly or through groups.

© 2023, Amazon Web Services India Private Limited or its affiliates.



IAM > Users > NewUser > Create access key

Step 1
Access key best practices & alternatives

Step 2 - optional
Set description tag

Step 3
Retrieve access keys

Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

● Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

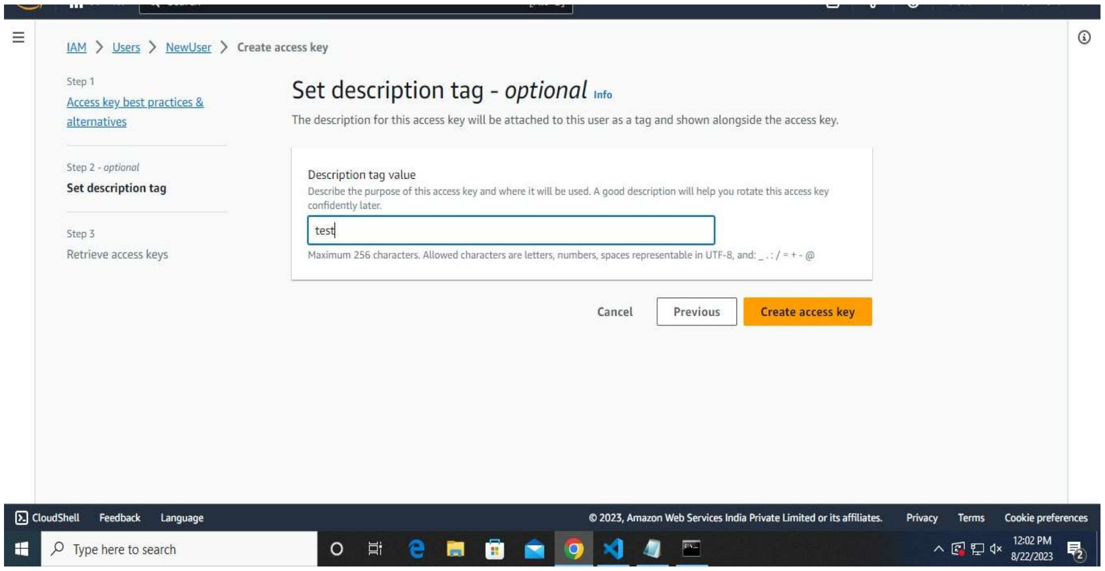
○ Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

○ Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

○ Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

© 2023, Amazon Web Services India Private Limited or its affiliates.

8. Add Descriptive tags (optional) and click on 'Create access key'.



IAM > Users > NewUser > Create access key

Step 1
Access key best practices & alternatives

Step 2 - optional
Set description tag

Step 3
Retrieve access keys

Set description tag - optional Info

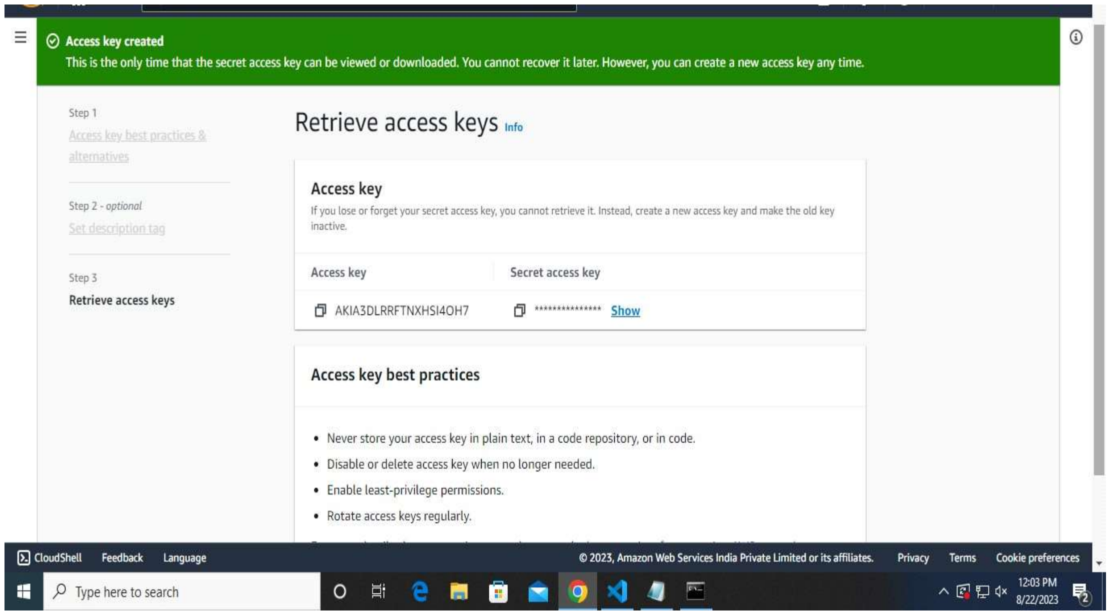
The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

test

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

9. Now copy the Access key and secret access key in a notepad in following format. Save the notepad file as .tf extension. (You will find the ami key when you create a new instance in AWS.)



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.

Step 1
Access key best practices & alternatives

Step 2 - optional
Set description tag

Step 3
Retrieve access keys

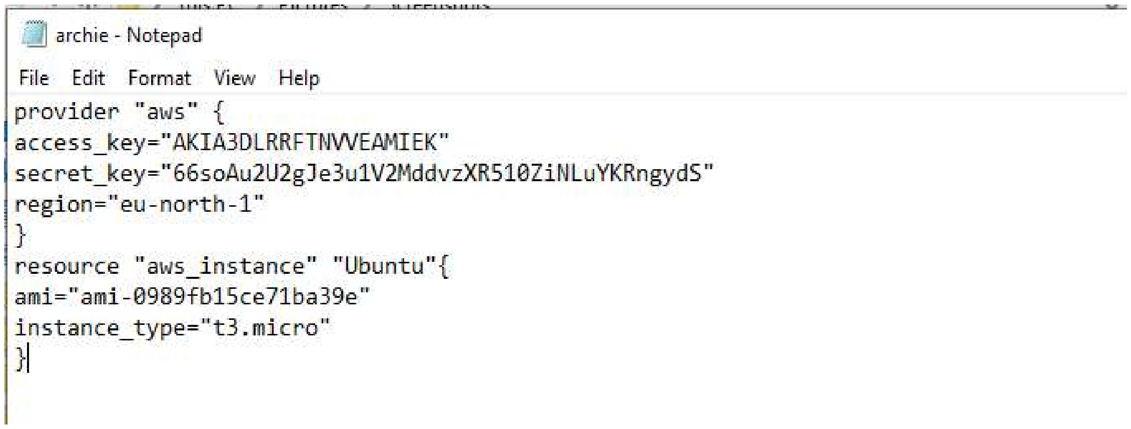
Retrieve access keys Info

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
AKIA3DLRRFTNXHSI4OH7	*************** Show

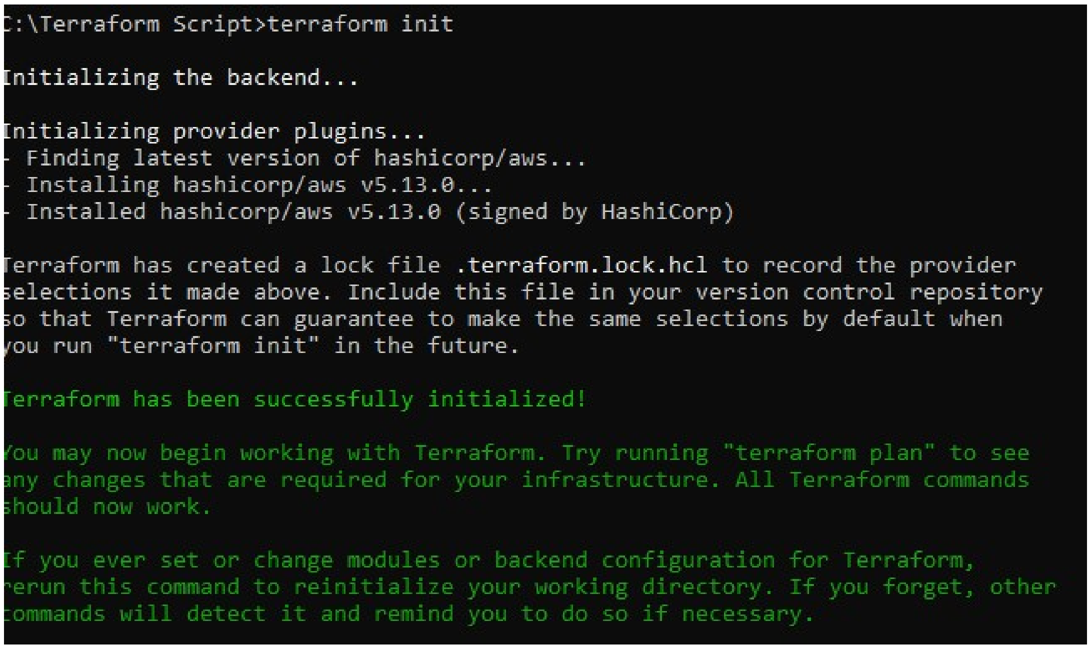
Access key best practices

- 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.

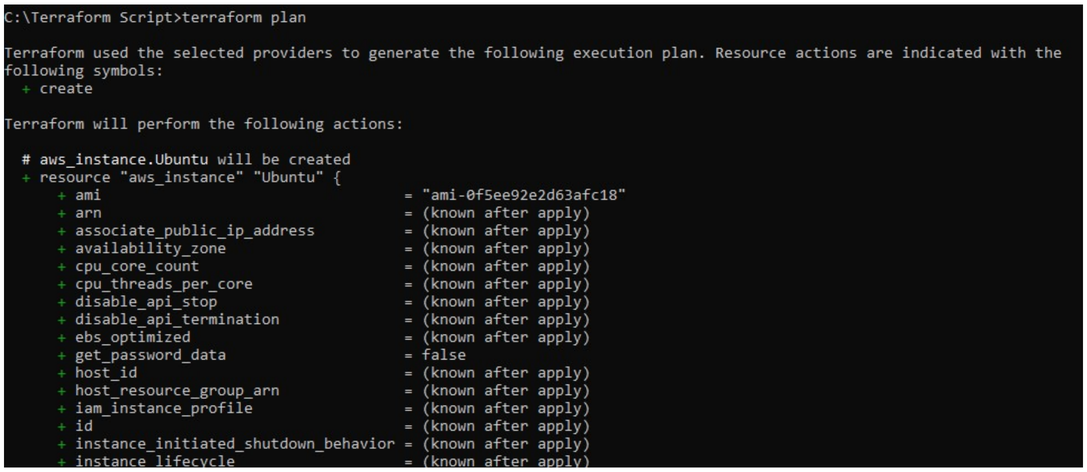


```
provider "aws" {
access_key="AKIA3DLRRFTNVVEAMIEK"
secret_key="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"
region="eu-north-1"
}
resource "aws_instance" "Ubuntu"{
ami="ami-0989fb15ce71ba39e"
instance_type="t3.micro"
}
```

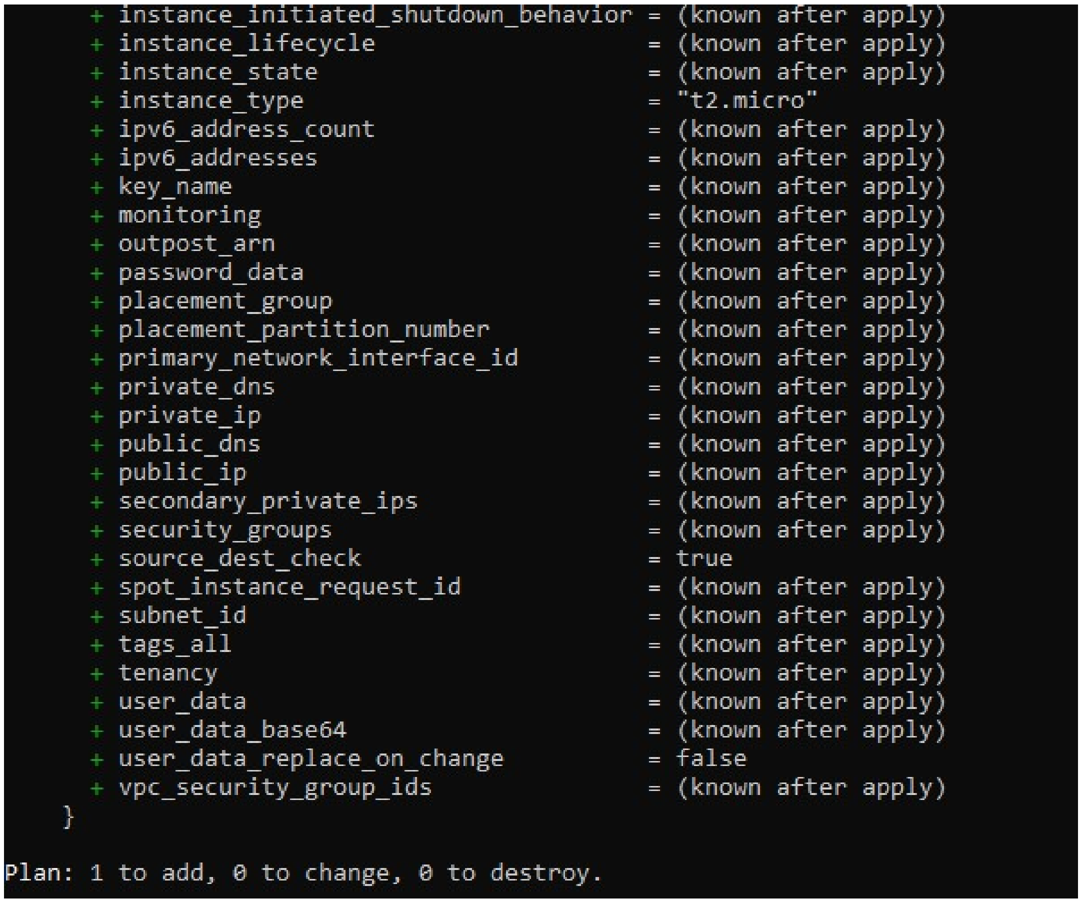
10. Open the CMD in the folder where you saved your .tf file and run the commands
    a. Terraform init
    b. Terraform plan
    c. Terraform apply



```
:\Terraform Script>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.13.0...
- Installed hashicorp/aws v5.13.0 (signed by HashiCorp)

erraform has created a lock file .terraform.lock.hcl to record the provider
elections it made above. Include this file in your version control repository
o that Terraform can guarantee to make the same selections by default when
ou run "terraform init" in the future.

erraform has been successfully initialized!

ou may now begin working with Terraform. Try running "terraform plan" to see
ny changes that are required for your infrastructure. All Terraform commands
hould now work.

f you ever set or change modules or backend configuration for Terraform,
erun this command to reinitialize your working directory. If you forget, other
ommands will detect it and remind you to do so if necessary.
```

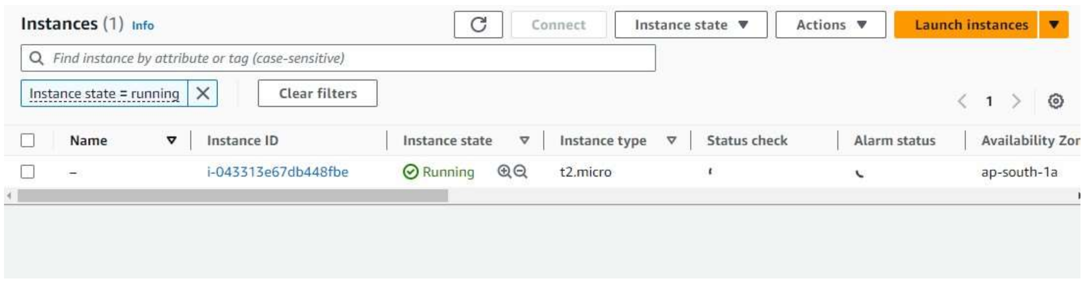


```
C:\Terraform Script>terraform plan

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.Ubuntu will be created
  + resource "aws_instance" "Ubuntu" {
      + ami                                  = "ami-0f5ee92e2d63afc18"
      + 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)
```



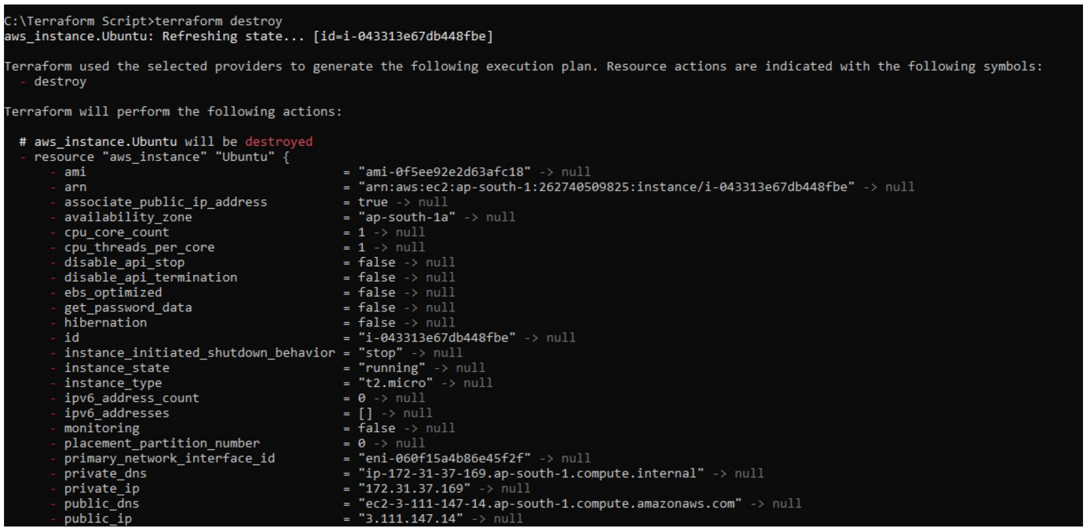
```
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (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)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags_all                             = (known after apply)
      + 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.
```

11. Instance has been created

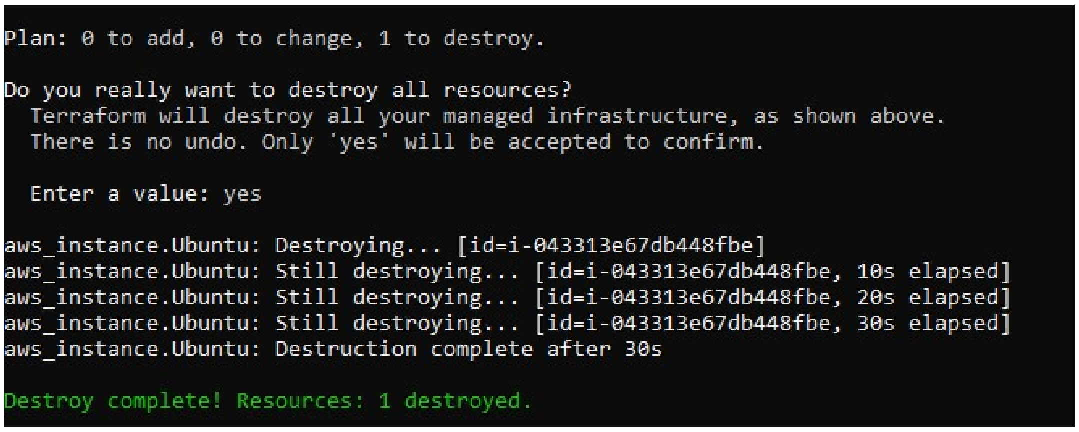


	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zor
	–	i-043313e67db448fbe	Running	t2.micro			ap-south-1a

12. To delete the instance run the command ‘Terraform destroy’.

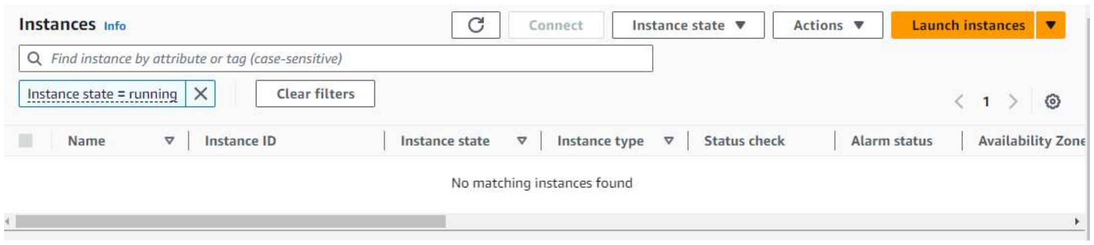


```
C:\Terraform Script>terraform destroy
aws_instance.Ubuntu: Refreshing state... [id=i-043313e67db448fbe]

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.Ubuntu will be destroyed
  - resource "aws_instance" "Ubuntu" {
      - ami                                  = "ami-0f5ee92e2d63afc18" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:262740509825:instance/i-043313e67db448fbe" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1a" -> 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-043313e67db448fbe" -> 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
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-060f15a4b86e45f2f" -> null
      - private_dns                          = "ip-172-31-37-169.ap-south-1.compute.internal" -> null
      - private_ip                           = "172.31.37.169" -> null
      - public_dns                           = "ec2-3-111-147-14.ap-south-1.compute.amazonaws.com" -> null
      - public_ip                            = "3.111.147.14" -> 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.Ubuntu: Destroying... [id=i-043313e67db448fbe]
aws_instance.Ubuntu: Still destroying... [id=i-043313e67db448fbe, 10s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-043313e67db448fbe, 20s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-043313e67db448fbe, 30s elapsed]
aws_instance.Ubuntu: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
```

13. Instance has been destroyed.



Instances 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

No matching instances found

<u>CONCLUSION</u>:

In this assignment we learnt about the terraform lifecycle, core concepts/terminologies and install it on Linux Machine.