

Aim:

To understand terraform lifecycle, core concepts/terminologies and install it on a linux machine.

LO Mapping: LO1, LO5

Theory:

1. Introduction to Terraform: Terraform is an open-source Infrastructure as Code (IaC) tool created by HashiCorp. It allows you to define and manage your infrastructure using configuration files. Terraform supports a wide range of cloud providers, including AWS, Azure, GCP, and more.

2. Core Concepts and Terminologies:

- **Providers:** Providers are responsible for understanding API interactions with the services Terraform manages. Examples include AWS, Azure, Google Cloud, etc.
- **Resources:** Resources are the most important element in the Terraform language. Each resource block describes one or more infrastructure objects, such as virtual networks, compute instances, or higher-level components.
- **Modules:** Modules are containers for multiple resources that are used together. They are the basic unit of organization within Terraform configurations.
- **State:** Terraform maintains a state file to keep track of the infrastructure it manages. This file maps the configuration to the real-world resources.
- **Plans:** A plan in Terraform represents the actions that Terraform will take to reach the desired state described in the configuration files. It helps you understand the changes before they are applied.
- **Provisioners:** Provisioners are used to execute scripts or commands on a local or remote machine as part of the resource creation or destruction process.
- **Terraform Configuration Files:** These are the .tf files where you define your infrastructure using the HashiCorp Configuration Language (HCL) or JSON.

3. Terraform Lifecycle:

The typical Terraform workflow involves the following stages:

- **Write:** Define infrastructure as code in configuration files.
- **Initialize:** Run `terraform init` to initialize the working directory with configuration files.
- **Plan:** Run `terraform plan` to create an execution plan, which shows what actions Terraform will take to achieve the desired state.
- **Apply:** Run `terraform apply` to execute the actions proposed in the plan.
- **Destroy:** Run `terraform destroy` to remove the infrastructure managed by Terraform.

Installing Terraform on a Linux Machine

Run Commands

```
lab1002@lab1002-HP-280-G3-MT:~$ wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
--2024-08-13 11:12:02-- https://apt.releases.hashicorp.com/gpg
Resolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... [sudo] password for lab1002: 18.172.78.129, 18.172.78.65, 18.172.78.12, .
..
Connecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|18.172.78.129|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3980 (3.9K) [binary/octet-stream]
Saving to: 'STDOUT'

-
          100%[=====] 3.89K  ---KB/s   in 0s

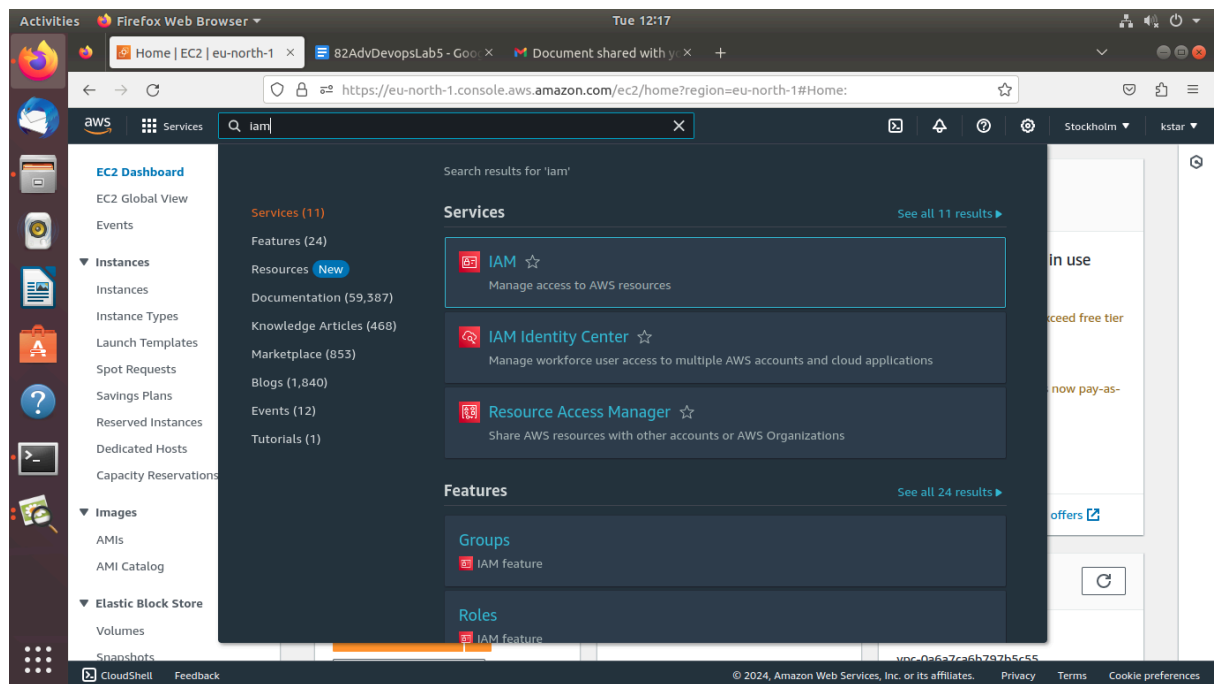
2024-08-13 11:12:02 (237 MB/s) - written to stdout [3980/3980]
```

```
lab1002@lab1002-HP-280-G3-MT:~$ 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
[sudo] password for lab1002:
deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com bionic main
```

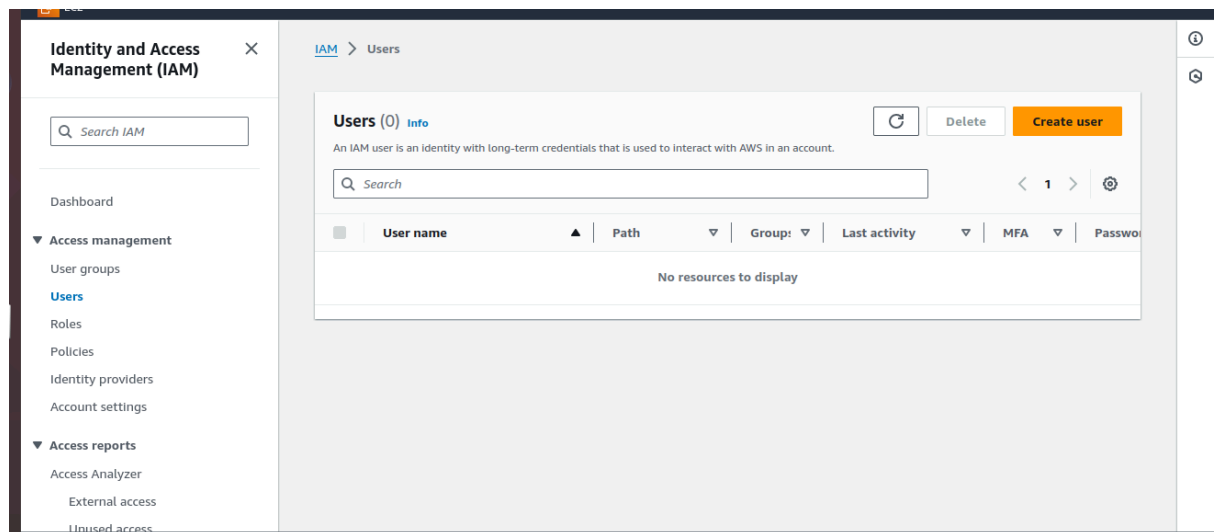
```
Activities  Terminal  Tue 11:18
lab1002@lab1002-HP-280-G3-MT: ~
File Edit View Search Terminal Help
lab1002@lab1002-HP-280-G3-MT:~$ sudo apt update && sudo apt install terraform
Ign:1 https://apt.releases.hashicorp.com bionic InRelease
Err:2 https://apt.releases.hashicorp.com bionic Release
404 Not Found [IP: 18.172.78.65 443]
Get:3 https://prod-cdn.packages.k8s.io/repositories/istio/kubernetes/core/stable/v1.30/deb InRelease [1,186 B]
Hit:4 http://ln.archive.ubuntu.com/ubuntu bionic InRelease
Err:3 https://prod-cdn.packages.k8s.io/repositories/istio/kubernetes/core/stable/v1.30/deb InRelease
The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
Hit:5 http://ln.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:6 http://ln.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:7 http://security.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
E: The repository 'https://apt.releases.hashicorp.com bionic Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
W: GPG error: https://prod-cdn.packages.k8s.io/repositories/istio/kubernetes/core/stable/v1.30/deb InRelease: The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
E: The repository 'https://pkgs.k8s.io/core/stable/v1.30/deb InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
W: Target Packages (main/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list:52 and /etc/apt/sources.list.d/hashicorp.list:1
W: Target Packages (main/binary-all/Packages) is configured multiple times in /etc/apt/sources.list:52 and /etc/apt/sources.list.d/hashicorp.list:1
W: Target Translations (main/i18n/Translation-en_IN) is configured multiple times in /etc/apt/sources.list:52 and /etc/apt/sources.list.d/hashicorp.list:1
W: Target Translations (main/i18n/Translation-en) is configured multiple times in /etc/apt/sources.list:52 and /etc/apt/sources.list.d/hashicorp.list:1
W: Target DEP-11 (main/dep11/components-amd64.yml) is configured multiple times in /etc/apt/sources.list:52 and /etc/apt/sources.list.d/hashicorp.list:1
W: Target DEP-11 (main/dep11/components-all.yml) is configured multiple times in /etc/apt/sources.list:52 and /etc/apt/sources.list.d/hashicorp.list:1
W: Target DEP-11-icons-small (main/dep11/icons-48x48.tar) is configured multiple times in /etc/apt/sources.list:52 and /etc/apt/sources.list.d/hashicorp.list:1
W: Target DEP-11-icons (main/dep11/icons-64x64.tar) is configured multiple times in /etc/apt/sources.list:52 and /etc/apt/sources.list.d/hashicorp.list:1
W: Target CNF (main/cnf/Commands-amd64) is configured multiple times in /etc/apt/sources.list:52 and /etc/apt/sources.list.d/hashicorp.list:1
W: Target CNF (main/cnf/Commands-all) is configured multiple times in /etc/apt/sources.list:52 and /etc/apt/sources.list.d/hashicorp.list:1
```

Output:

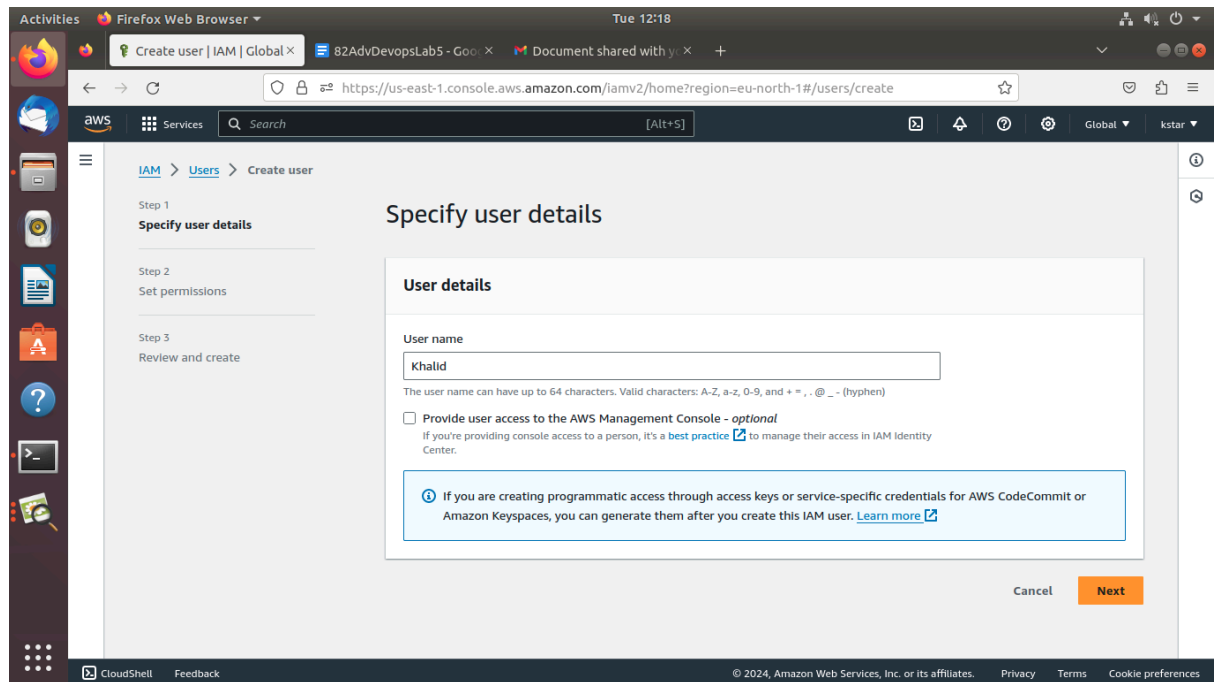
1. Search IAM



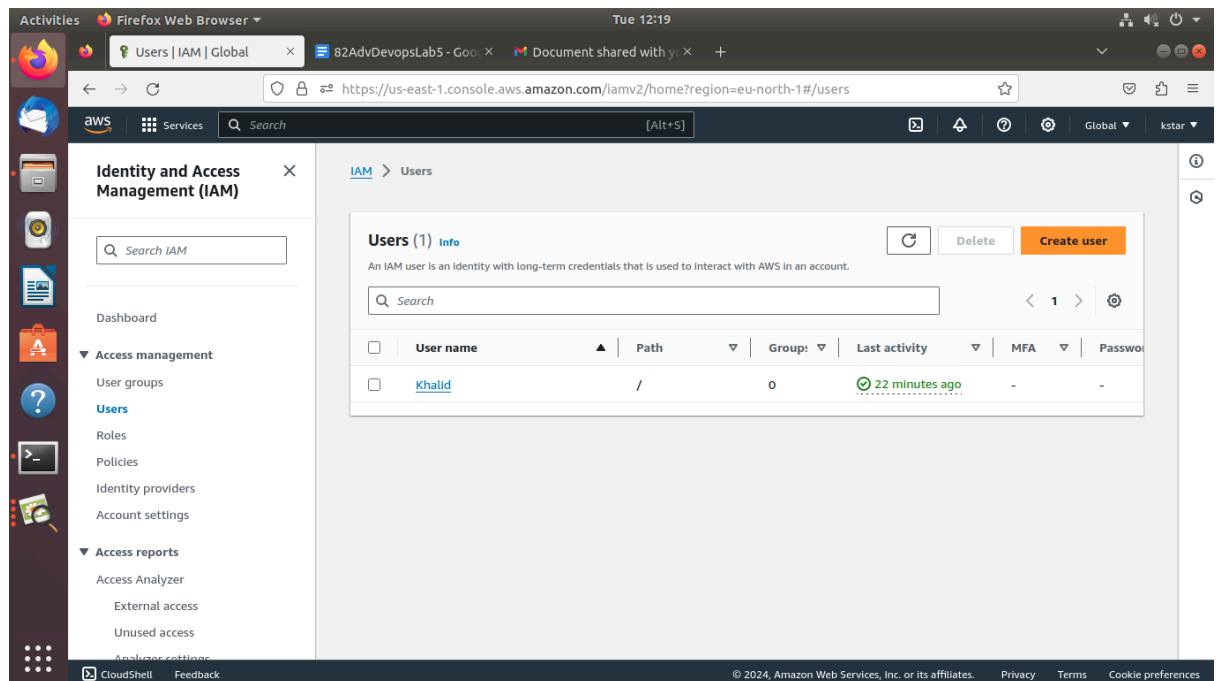
2. Go To Users and Click Yellow Create User Button



3. Click Next



4. User Successfully Created, Click on user and click Create Access Key.



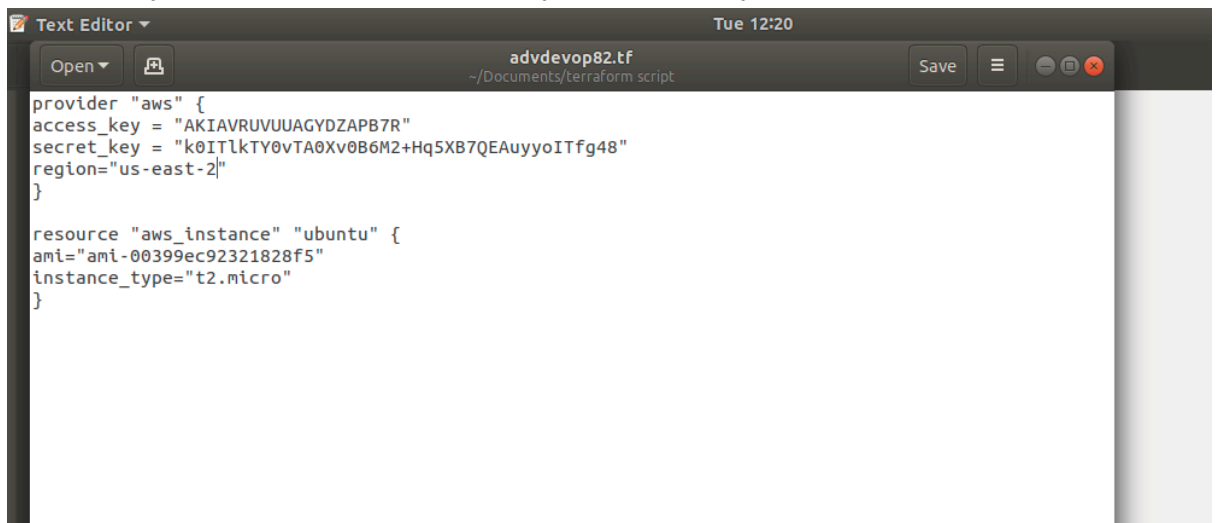
5. Keep the defaults and Complete creation of access keys.

The screenshot displays the AWS IAM console interface in a Firefox browser. The left sidebar shows the 'Identity and Access Management (IAM)' menu with options like Dashboard, Access management, and Access reports. The main content area shows the details for a user named 'Khalid'. The 'Summary' section includes the user's ARN, console access status (Disabled), and creation date (August 13, 2024, 11:26 UTC+05:30). Below this, the 'Permissions policies' section shows three policies attached to the user. The 'Access key 2' is highlighted as 'Used today'.

ARN	Console access	Access key 1
arn:aws:iam::381492174861:user/Khalid	Disabled	AKIAVRUVUJAGTRR83477 - Active Never used. Created today.

Created	Last console sign-in	Access key 2
August 13, 2024, 11:26 (UTC+05:30)	-	AKIAVRUVUJAGYDZAP87R - Active Used today. Created today.

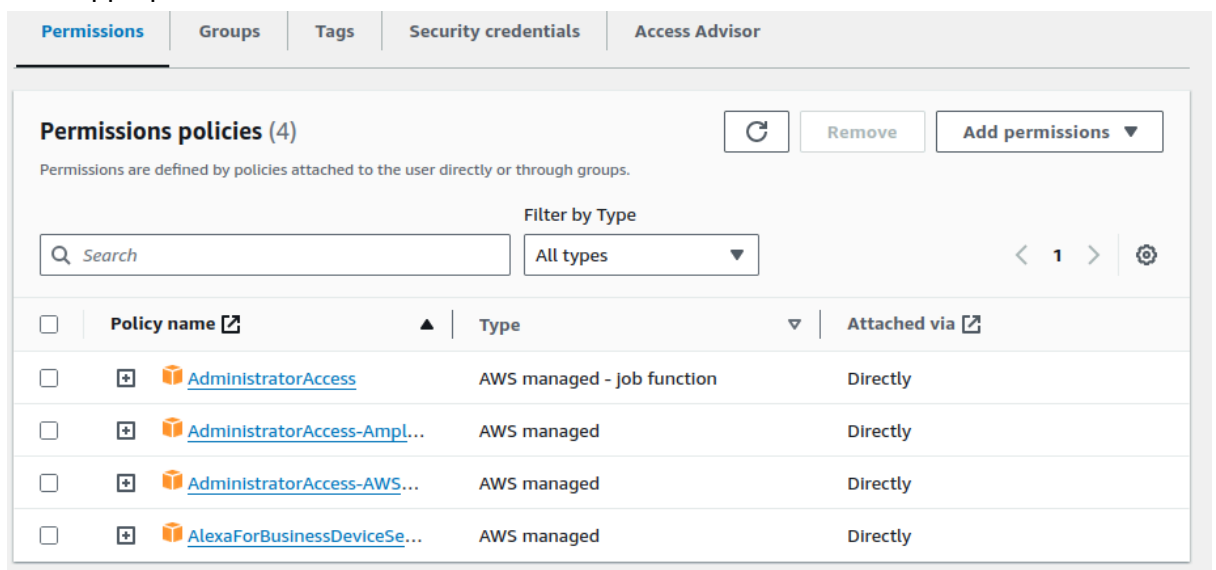
6. Create a Folder 'Terraform Script' (or any name) and create a .tf file with a custom name and type this into it. Replace the key and secret key



The screenshot shows a text editor window titled 'advdevop82.tf' with the file path '~/Documents/terraform script'. The script contains the following Terraform configuration:

```
provider "aws" {  
  access_key = "AKIAVRUVUUAGYDZAPB7R"  
  secret_key = "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"  
  region="us-east-2"  
}  
  
resource "aws_instance" "ubuntu" {  
  ami="ami-00399ec92321828f5"  
  instance_type="t2.micro"  
}
```

7. Give Appropriate Permissions to the user in order to create instances.



The screenshot shows the AWS IAM console 'Permissions' tab for a user. It displays a list of permissions policies attached to the user. The table below summarizes the policies shown:

Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Directly
AdministratorAccess-Ampl...	AWS managed	Directly
AdministratorAccess-AWS...	AWS managed	Directly
AlexaForBusinessDeviceSe...	AWS managed	Directly

8. Run Terraform Init after opening the directory with .tf file.

```
lab1002@lab1002-HP-280-G3-MT:~/Documents/Terraform Script$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.62.0...
- Installed hashicorp/aws v5.62.0 (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.
```

9. Run Terraform plan

```
lab1002@lab1002-HP-280-G3-MT:~/Documents/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-00399ec92321828f5"
  + 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)
  + placement_group         = (known after apply)
  + placement_partition_number = (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)

  + capacity_reservation_specification (known after apply)

  + cpu_options (known after apply)

  + ebs_block_device (known after apply)

  + enclave_options (known after apply)

  + ephemeral_block_device (known after apply)

  + instance_market_options (known after apply)

  + maintenance_options (known after apply)

  + metadata_options (known after apply)

  + network_interface (known after apply)

  + private_dns_name_options (known after apply)

  + root_block_device (known after apply)
}

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

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

10. Run Terraform Apply

```
Lab1002@lab1002-HP-280-G3-MT:~/Documents/Terraform Script$ 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.Ubuntu will be created
+ resource "aws_instance" "Ubuntu" {
  + ami                        = "ami-00399ec92321828f5"
  + 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)
  + placement_group           = (known after apply)
  + placement_partition_number = (known after apply)

  + cpu_options (known after apply)
  + ebs_block_device (known after apply)
  + enclave_options (known after apply)
  + ephemeral_block_device (known after apply)
  + instance_market_options (known after apply)
  + maintenance_options (known after apply)
  + metadata_options (known after apply)
  + network_interface (known after apply)
  + private_dns_name_options (known after apply)
  + root_block_device (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.Ubuntu: Creating...
aws_instance.Ubuntu: Still creating... [10s elapsed]
aws_instance.Ubuntu: Still creating... [20s elapsed]
aws_instance.Ubuntu: Still creating... [30s elapsed]
aws_instance.Ubuntu: Creation complete after 35s [id=i-0475c06df86ffeeff]

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

11. Run Terraform Destroy

```
lab1002@lab1002-HP-280-G3-MT:~/Documents/Terraform Scripts$ terraform destroy
aws_instance.Ubuntu: Refreshing state... [id=i-0475c06df86ffeeff]

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-00399ec92321828f5" -> null
  - arn                  = "arn:aws:ec2:us-east-2:058264306232:instance/i-0475c06df86ffeeff" -> null
  - associate_public_ip_address = true -> null
  - availability_zone      = "us-east-2a" -> 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-0475c06df86ffeeff" -> 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-01100d4760c3736b5" -> null
  - private_dns              = "ip-172-31-0-193.us-east-2.compute.internal" -> null
  - private_ip               = "172.31.0.193" -> null
  - public_dns                = "ec2-3-145-103-163.us-east-2.compute.amazonaws.com" -> null
  - public_ip                 = "3.145.103.163" -> null
  - secondary_private_ips     = [] -> null

  - private_dns_name_options {
    - enable_resource_name_dns_a_record = false -> null
    - enable_resource_name_dns_aaaa_record = false -> null
    - hostname_type                       = "ip-name" -> null
  }

  - root_block_device {
    - delete_on_termination = true -> null
    - device_name           = "/dev/sda1" -> null
    - encrypted              = false -> null
    - iops                   = 100 -> null
    - tags                   = {} -> null
    - tags_all               = {} -> null
    - throughput             = 0 -> null
    - volume_id              = "vol-00709ed4a8b64b805" -> null
    - volume_size            = 8 -> null
    - volume_type             = "gp2" -> null
    # (1 unchanged attribute hidden)
  }
}

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-0475c06df86ffeeff]
aws_instance.Ubuntu: Still destroying... [id=i-0475c06df86ffeeff, 10s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0475c06df86ffeeff, 20s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0475c06df86ffeeff, 30s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0475c06df86ffeeff, 40s elapsed]
aws_instance.Ubuntu: Destruction complete after 42s

Destroy complete! Resources: 1 destroyed.
```

Conclusion:

Understanding Terraform involves grasping its core concepts such as providers, resources, modules, state, and plans. The Terraform lifecycle guides the process from writing configurations to applying and destroying infrastructure. Installing Terraform on a Linux machine is straightforward and involves adding the HashiCorp repository and using the package manager to install the tool. Once installed, Terraform provides a robust way to manage infrastructure as code, ensuring consistency, repeatability, and scalability across your cloud environments.