St. Francis Institute of Technology, Mumbai-400103

Department of Information Technology

A.Y. 2023-2024

Class: TE-ITA/B, Semester: V

Subject: **Advanced DevOps Lab**

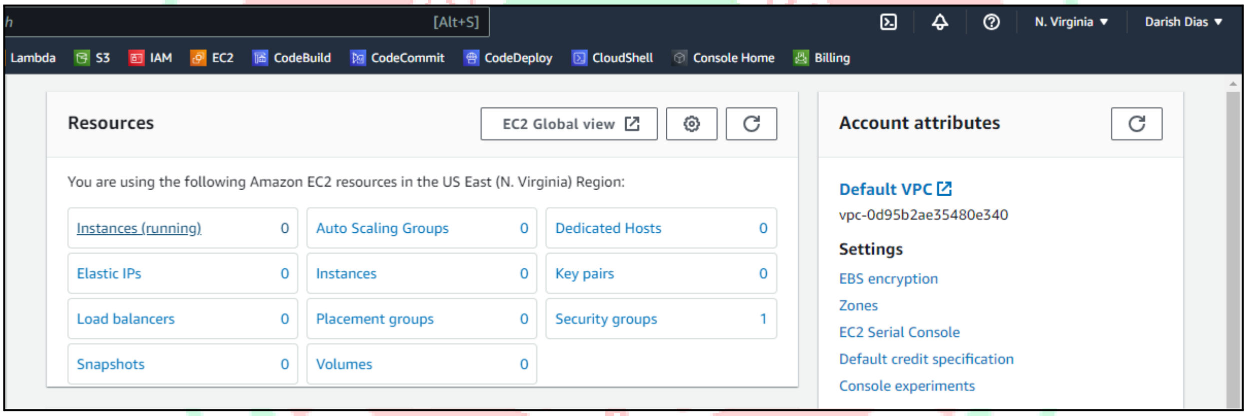
Experiment – 8: To build, apply and destroy AWS Resources using Terraform.

1. **Aim:** To build, apply and destroy AWS using Terraform.
2. **Objectives:** After study of this experiment, the student will be able to
   - Understand basic Terraform commands and concept of creating instance on EC2 using terraform.
3. **Lab objective mapped :** ITL504.3: To be familiarized with infrastructure as code for provisioning, compliance, and management of any cloud infrastructure and d service.
4. **Prerequisite:** Fundamentals of cloud computing and AWS account.
5. **Requirements:** PC and Internet
6. **Pre-Experiment Exercise:**

Brief Theory:

7. **Laboratory Exercise**

Step 1: First we will check that no instance is running on EC2.



Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	0	Auto Scaling Groups	0	Dedicated Hosts	0
Elastic IPs	0	Instances	0	Key pairs	0
Load balancers	0	Placement groups	0	Security groups	1
Snapshots	0	Volumes	0		

Account attributes

Default VPC

vpc-0d95b2ae35480e340

Settings

EBS encryption

Zones

EC2 Serial Console

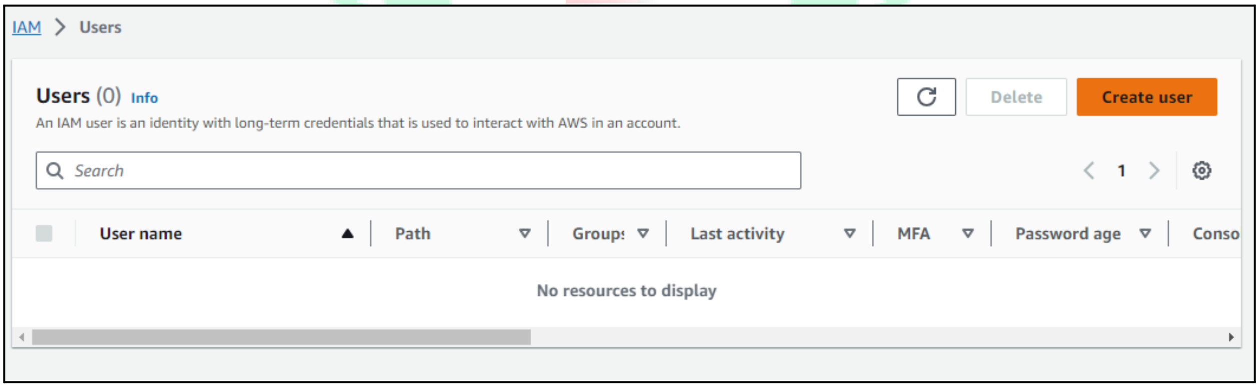
Default credit specification

Console experiments

Step 2: Create an IAM user with Programmatic Password, Administrator access and download access key and secret key from download.csv

IAM:

USERS:



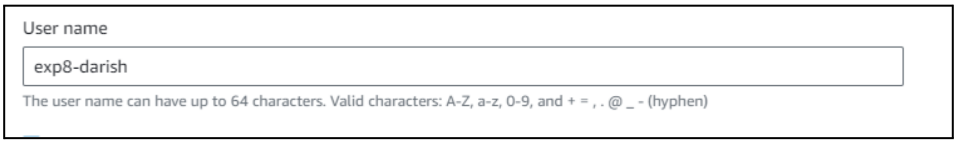
Users (0) Info

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	Conso

No resources to display

ADD USER:

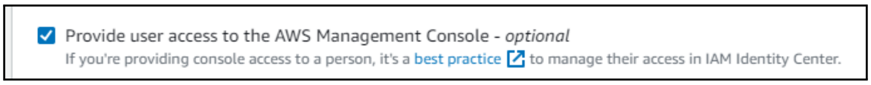


User name

exp8-darish

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

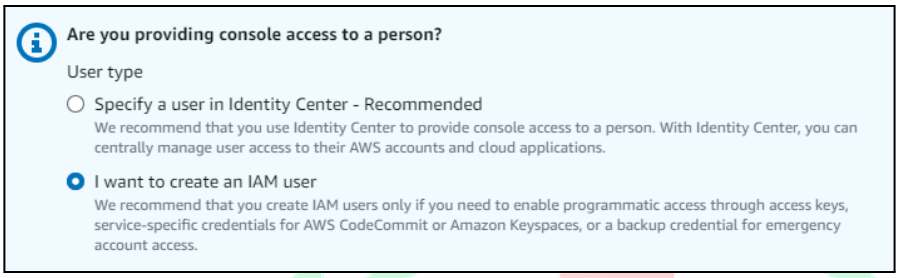
Give console access



☑ 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.

I want to create user



Are you providing console access to a person?

User type

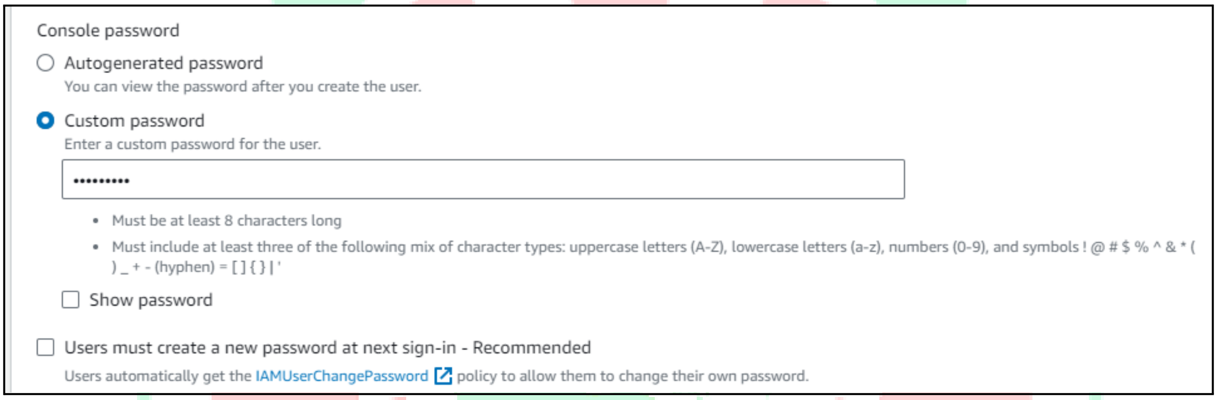
○ Specify a user in Identity Center - Recommended

We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

● I want to create an IAM user

We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

custom pwd



Console password

○ Autogenerated password

You can view the password after you create the user.

● Custom password

Enter a custom password for the user.

•••••••••

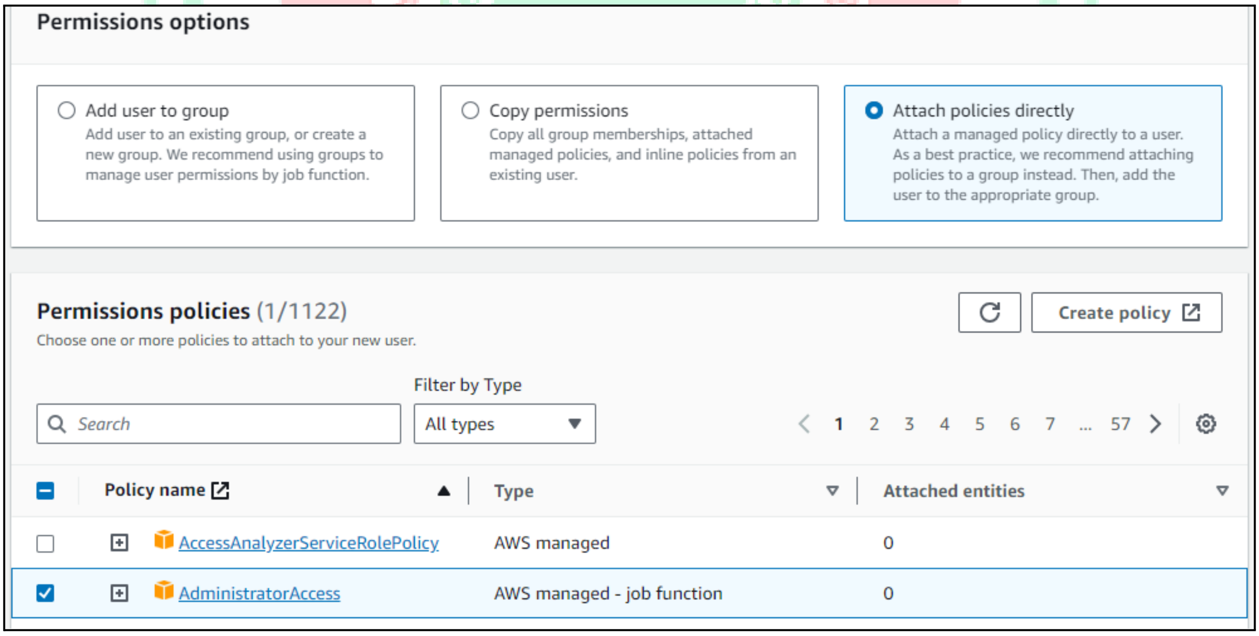
- Must be at least 8 characters long
- Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # $ % ^ & * ( ) _ + - (hyphen) = [ ] { } | '

□ Show password

□ Users must create a new password at next sign-in - Recommended

Users automatically get the IAMUserChangePassword policy to allow them to change their own password.

Attach policy directly : Administrator Access



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

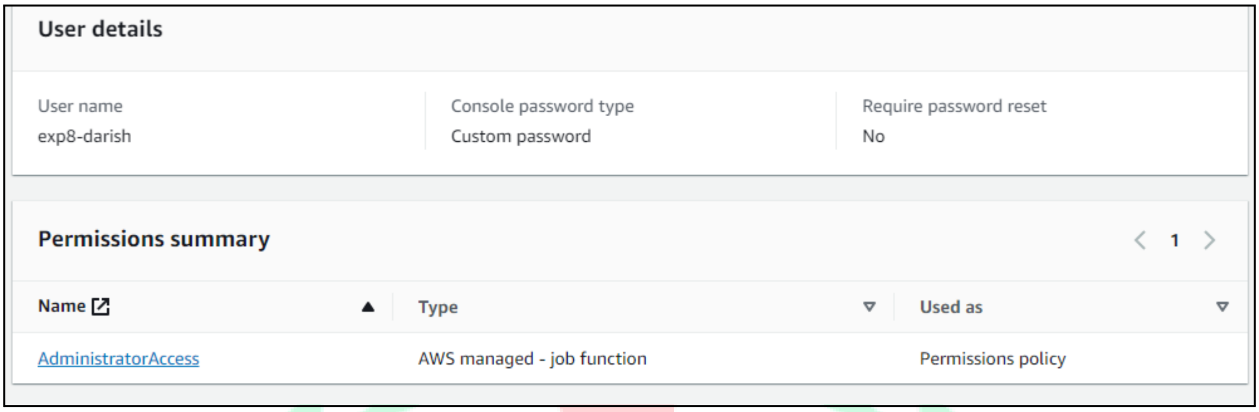
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1122)

Choose one or more policies to attach to your new user.

▬	Policy name	Type	Attached entities
□	AccessAnalyzerServiceRolePolicy	AWS managed	0
☑	AdministratorAccess	AWS managed - job function	0

Review:



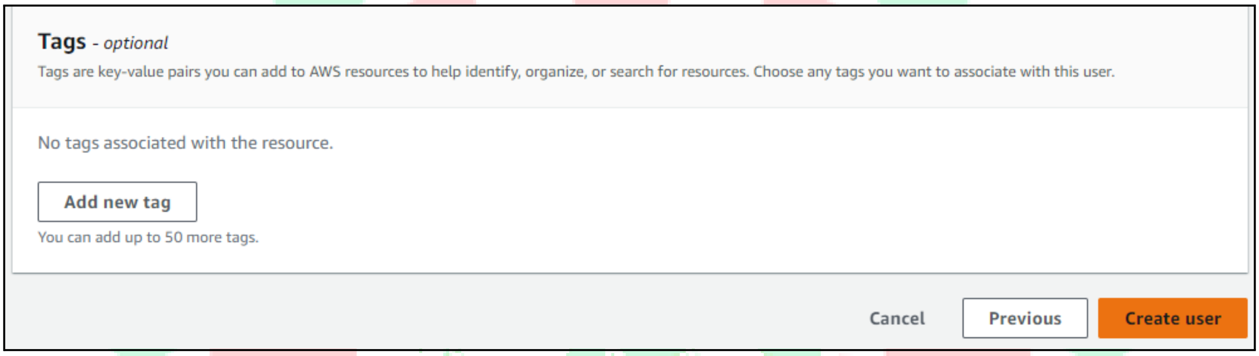
User details

User name	Console password type	Require password reset
exp8-darish	Custom password	No

Permissions summary

Name	Type	Used as
AdministratorAccess	AWS managed - job function	Permissions policy

Create user



Tags - *optional*

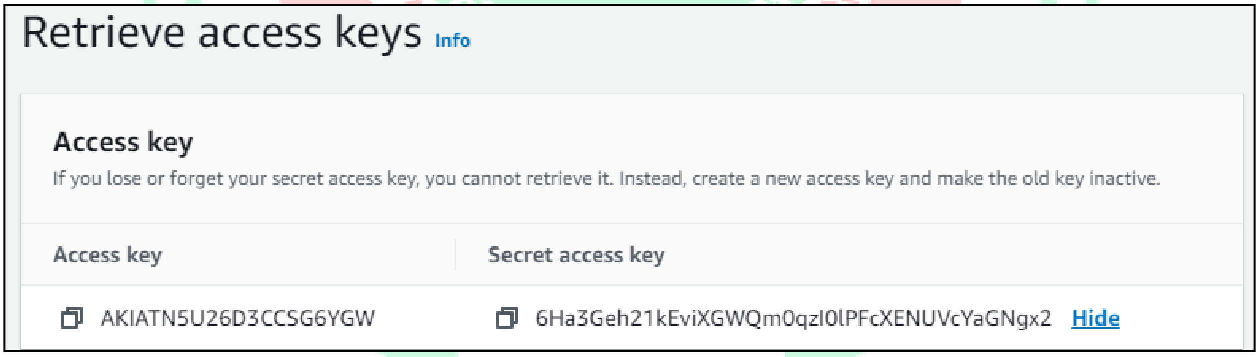
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

You can add up to 50 more tags.

Clk on user name

Create access key



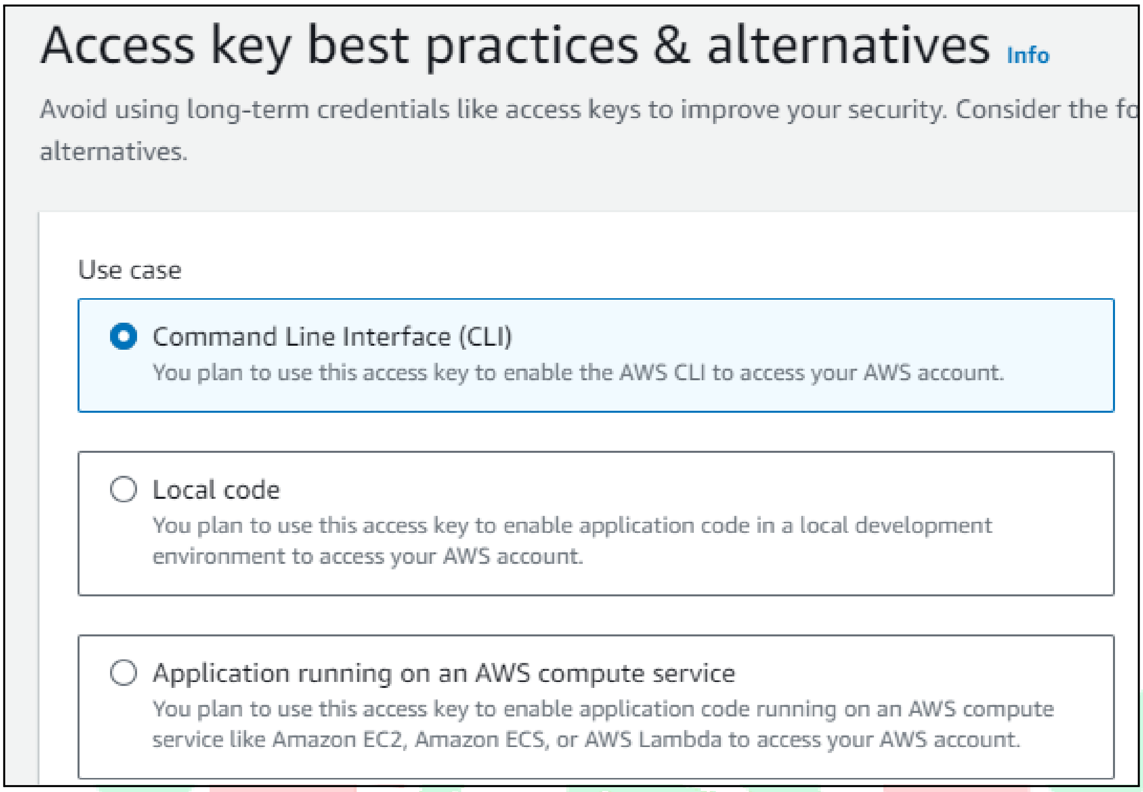
Retrieve access keys

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
AKIATN5U26D3CCSG6YGW	6Ha3Geh21kEviXGWQm0qzI0lPFcXENUVcYaGNgx2 Hide

Command line interface



Access key best practices & alternatives

Avoid using long-term credentials like access keys to improve your security. Consider the fo alternatives.

Use case

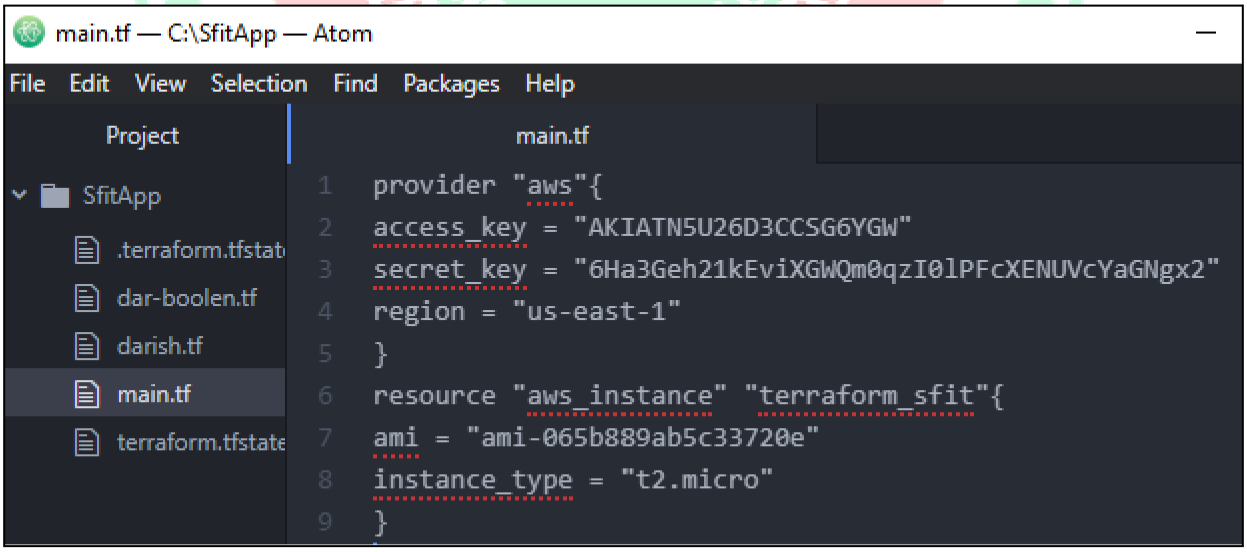
● Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

○ 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.

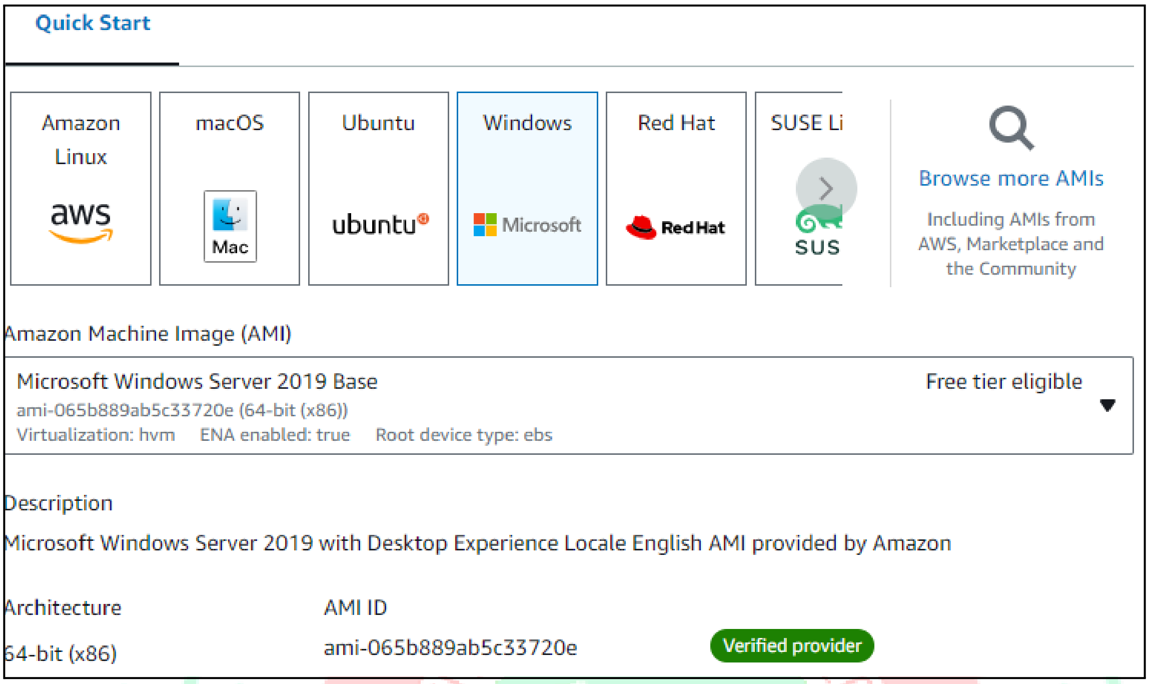
Create access key

Step 3: Now write a Terraform program in vs code, create new file with .tf extension



```
provider "aws"{
access_key = "AKIATN5U26D3CCSG6YGW"
secret_key = "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"
region = "us-east-1"
}
resource "aws_instance" "terraform_sfit"{
ami = "ami-065b889ab5c33720e"
instance_type = "t2.micro"
}
```

In Launch instance, you will get ami : amazon machine image



Amazon Machine Image (AMI)

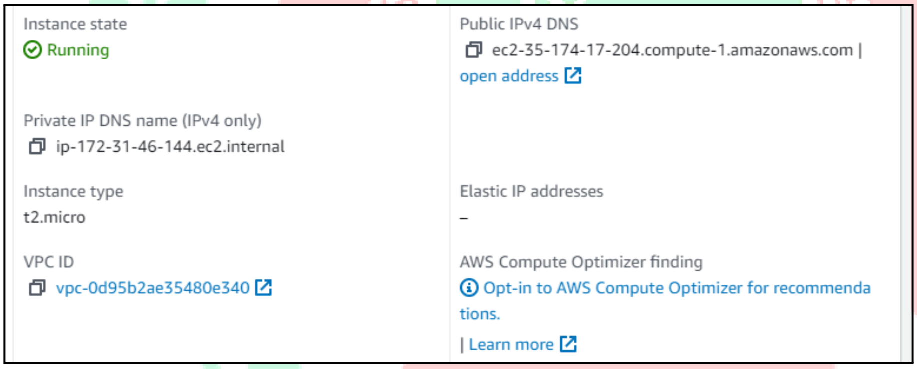
Microsoft Windows Server 2019 Base — Free tier eligible
ami-065b889ab5c33720e (64-bit (x86))
Virtualization: hvm   ENA enabled: true   Root device type: ebs

Description

Microsoft Windows Server 2019 with Desktop Experience Locale English AMI provided by Amazon

Architecture	AMI ID	
64-bit (x86)	ami-065b889ab5c33720e	Verified provider

For Instance type : t2.micro is freely available



Instance state
Running

Public IPv4 DNS
ec2-35-174-17-204.compute-1.amazonaws.com | open address

Private IP DNS name (IPv4 only)
ip-172-31-46-144.ec2.internal

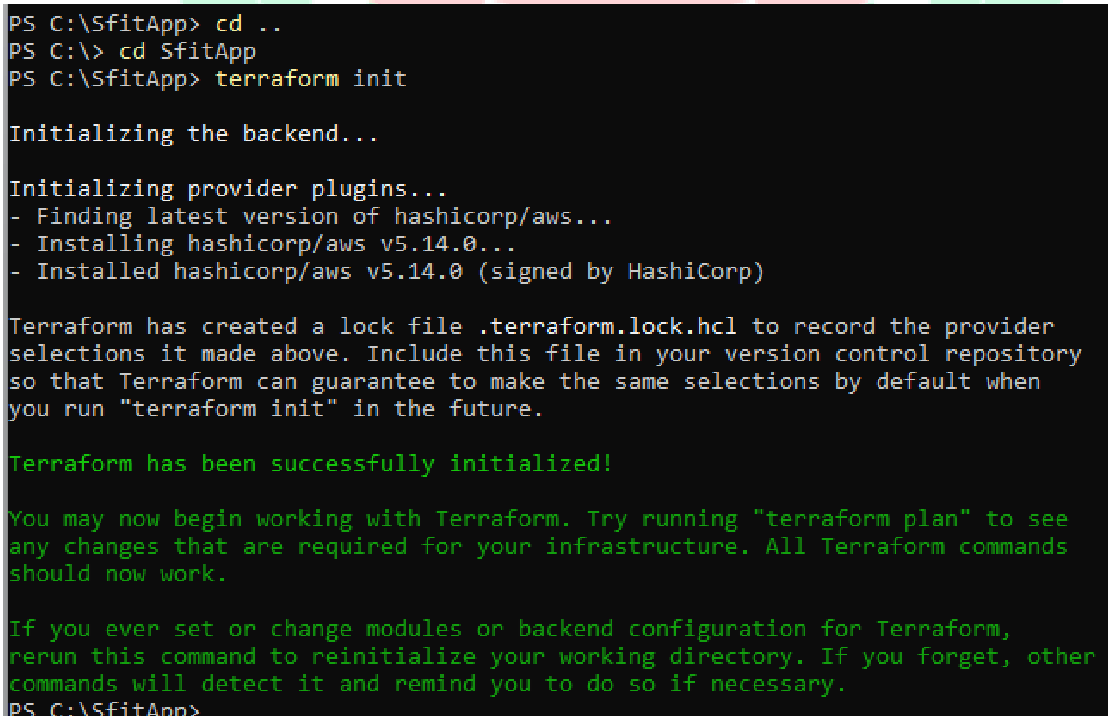
Instance type
t2.micro

Elastic IP addresses
–

VPC ID
vpc-0d95b2ae35480e340

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
| Learn more

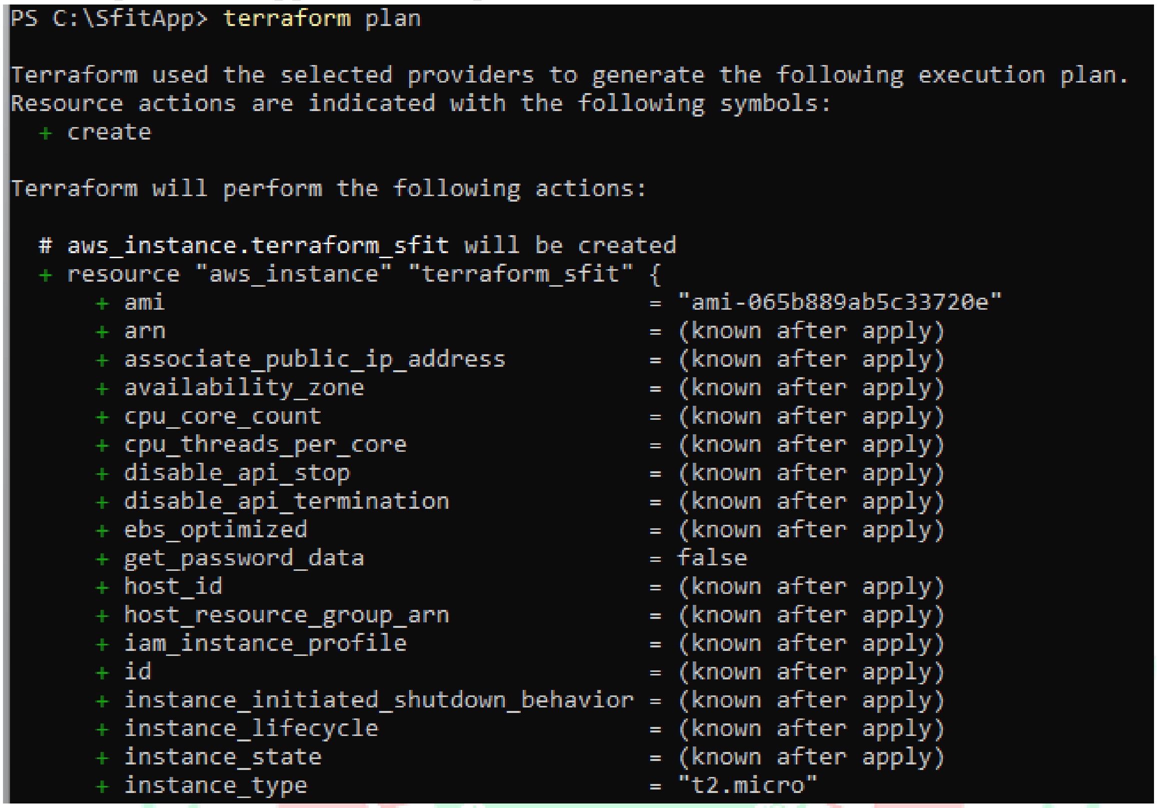
Step 4: Now initialize the terraform …type c:\SfitApp> terraform init



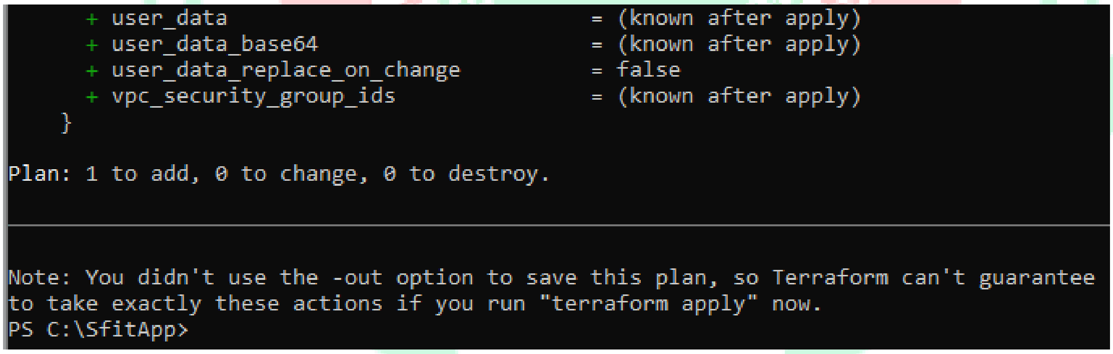
```
PS C:\SfitApp> cd ..
PS C:\> cd SfitApp
PS C:\SfitApp> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.14.0...
- Installed hashicorp/aws v5.14.0 (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.
PS C:\SfitApp>
```

Terraform has been initialized successfully.

Step 5: c:\sfitApp>terraform plan

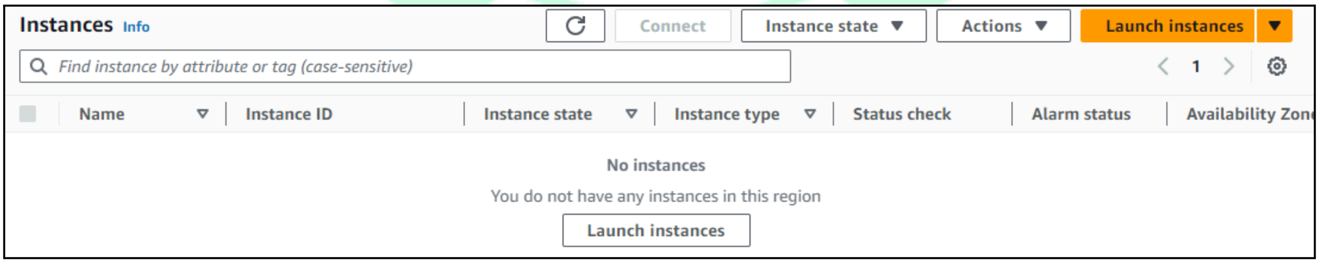


```
PS C:\SfitApp> terraform plan

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_sfit will be created
  + resource "aws_instance" "terraform_sfit" {
      + ami                                  = "ami-065b889ab5c33720e"
      + 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"
```



```
      + 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.

─────────────────────────────────────────────────────────────────────────────

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.
PS C:\SfitApp>
```

Step 6: Check the instance on Ec2 before terraform apply

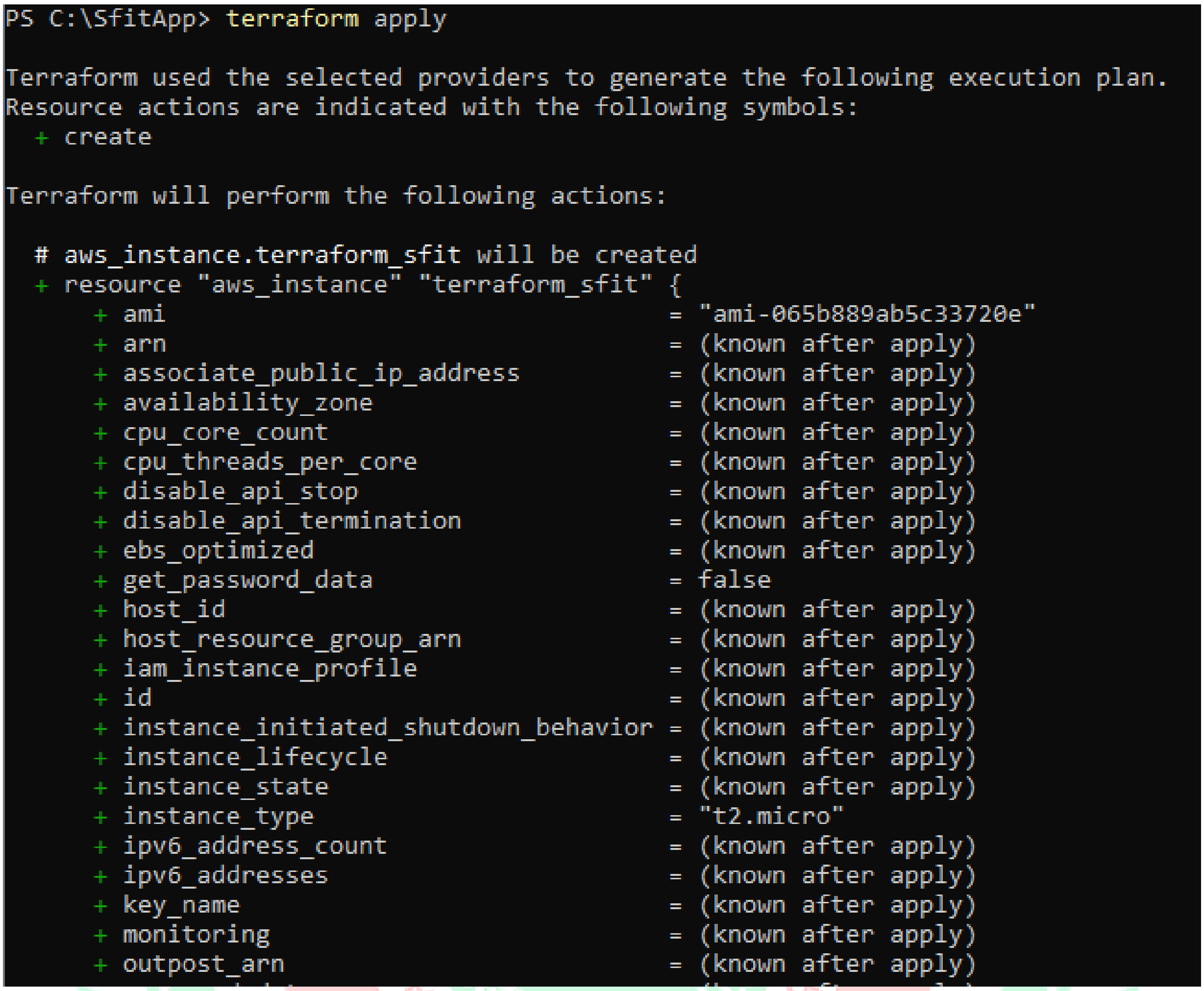


No instances

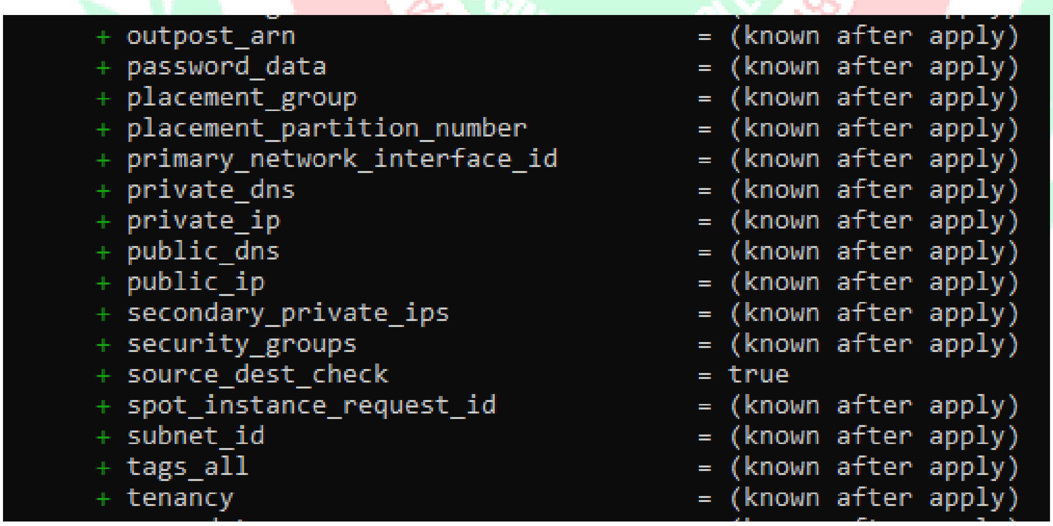
You do not have any instances in this region

Instance is not yet created.

Step 7: Terraform apply

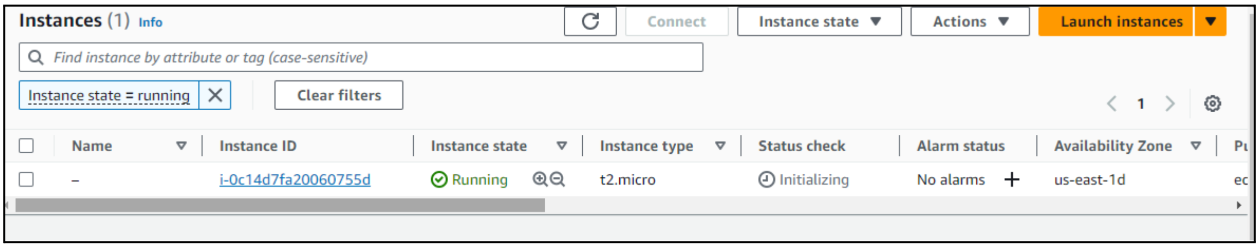


```
PS C:\SfitApp> terraform apply

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_sfit will be created
  + resource "aws_instance" "terraform_sfit" {
      + ami                                  = "ami-065b889ab5c33720e"
      + 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)
```



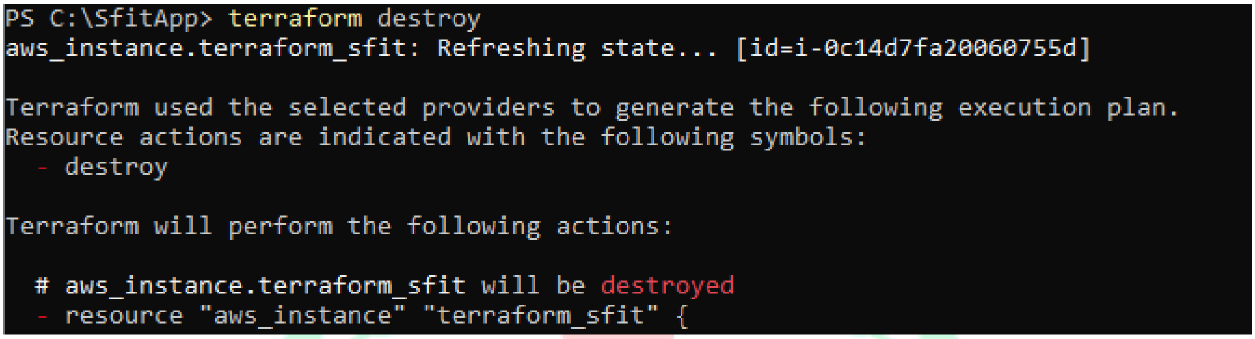
```
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (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)
```

Step 8: Check terraform created instance on EC2…we have created 3 instances.

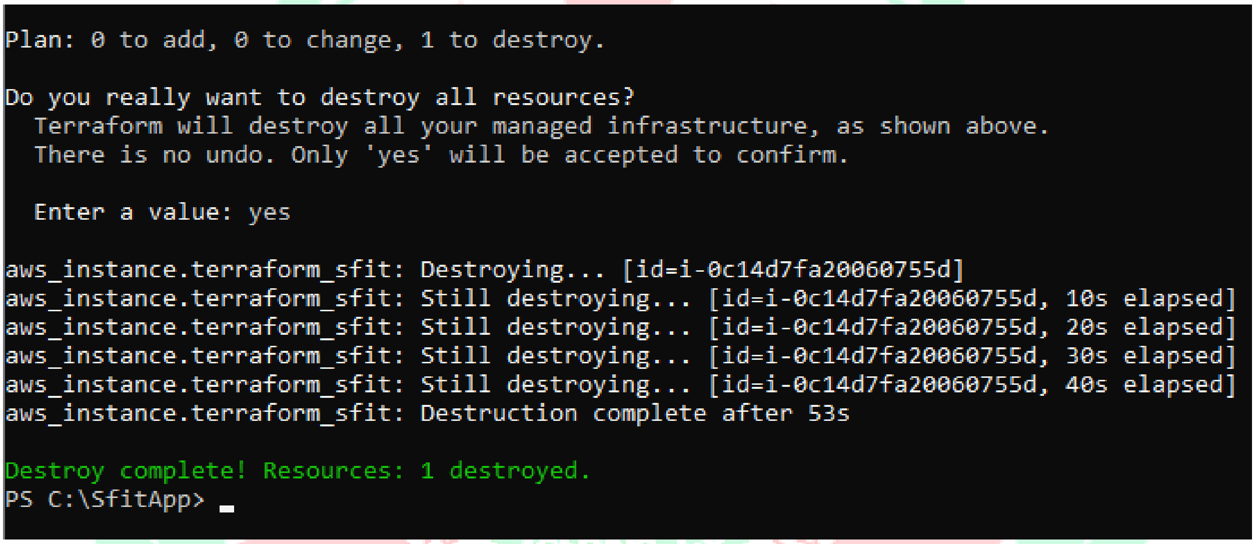


☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Pu
☐	–	i-0c14d7fa20060755d	✓ Running	t2.micro	Initializing	No alarms	us-east-1d	ec

Step 9: Now destroy the instance from command prompt….c:\SfitApp> terraform destroy



```
PS C:\SfitApp> terraform destroy
aws_instance.terraform_sfit: Refreshing state... [id=i-0c14d7fa20060755d]

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_sfit will be destroyed
  - resource "aws_instance" "terraform_sfit" {
```



```
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_sfit: Destroying... [id=i-0c14d7fa20060755d]
aws_instance.terraform_sfit: Still destroying... [id=i-0c14d7fa20060755d, 10s elapsed]
aws_instance.terraform_sfit: Still destroying... [id=i-0c14d7fa20060755d, 20s elapsed]
aws_instance.terraform_sfit: Still destroying... [id=i-0c14d7fa20060755d, 30s elapsed]
aws_instance.terraform_sfit: Still destroying... [id=i-0c14d7fa20060755d, 40s elapsed]
aws_instance.terraform_sfit: Destruction complete after 53s

Destroy complete! Resources: 1 destroyed.
PS C:\SfitApp> _
```

8. Post-Experiments Exercise

A. Extended Theory: (write in hand)

How to create AWS S3 Bucket using Terraform? (Only write Terraform Code)

B. Questions:(soft copy)

1. Name some major competitors of Terraform.
2. Why is Terraform preferred as one of the DevOps tools?

C. Conclusion:

1. Write what was performed in the experiment
2. Mention few applications of what was studied.
3. Write the significance of the studied topic

D. References:

1. https://www.bacancytechnology.com/blog/aws-s3-bucket-using-terraform 2. https://developer.hashicorp.com/terraform/tutorials?product_intent=terrafor m

----