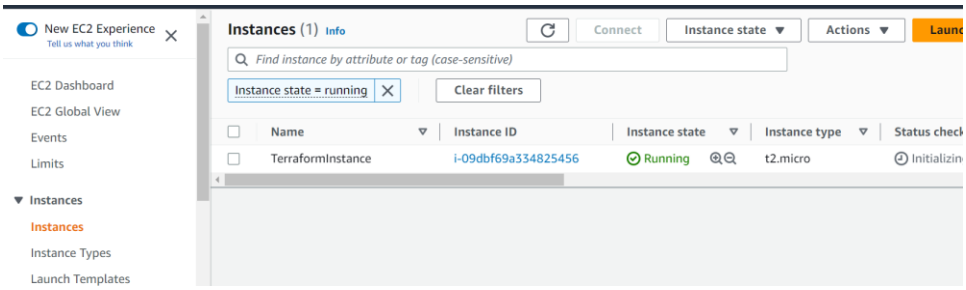
Day 64 - Terraform with AWS

Provisioning on AWS is quite easy and straightforward with Terraform.

Prerequisites
AWS CLI installed
The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

For the base OS, in this demo we will use Ubuntu.

Create a new EC2 Instance



SSH into EC2 instance and install aws-cli

```
sudo apt-get update && sudo apt install awscli
```

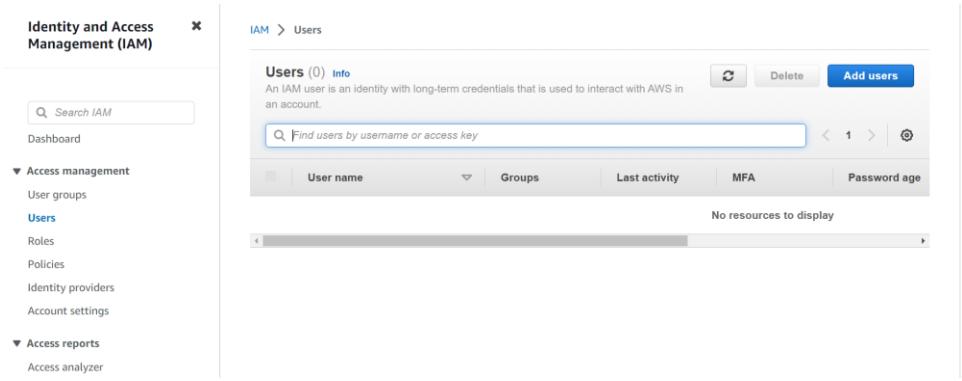
```
ubuntu@ip-172-31-85-217:~$ sudo apt-get update && sudo apt install awscli
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:4 https://apt.releases.hashicorp.com jammy InRelease [12.9 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [578 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [166 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [14.4 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [226 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [33.7 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [887 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [182 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [18.8 kB
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [35.3 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [8452 B]
Get:16 https://apt.releases.hashicorp.com jammy/main amd64 Packages [84.0 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [363 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [108 kB]
Get:19 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [9732 B]
```

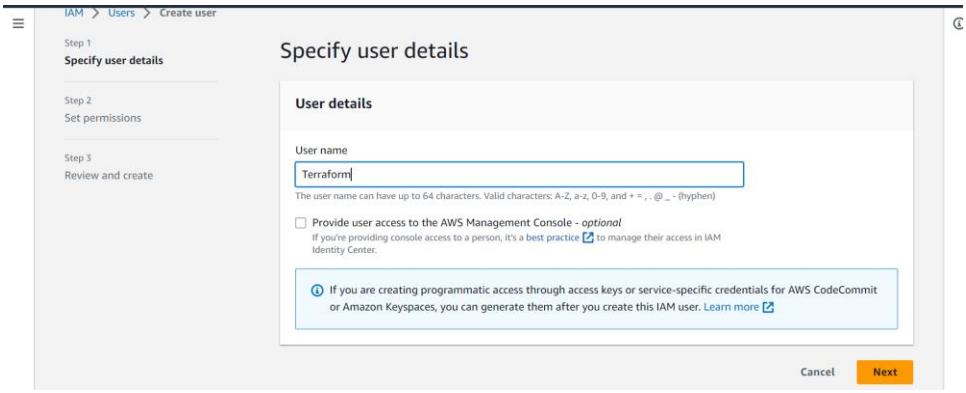
```
ubuntu@ip-172-31-85-217:~$ aws --version
aws-cli/1.22.34 Python/3.10.6 Linux/5.15.0-1031-aws botocore/1.23.34
```

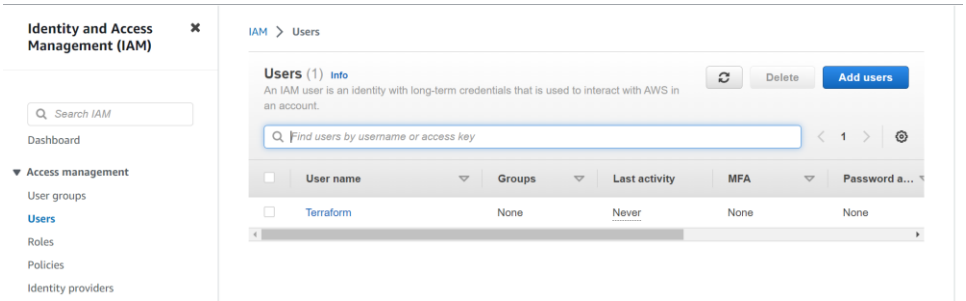
AWS IAM user

IAM (Identity Access Management) AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

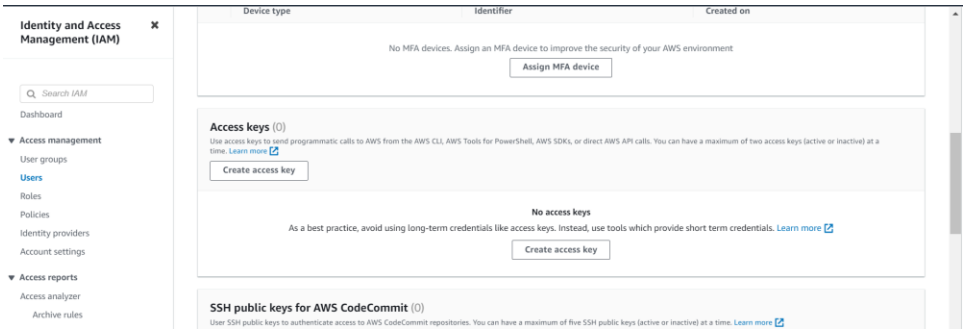
Goto IAM Dashboard and create a User.

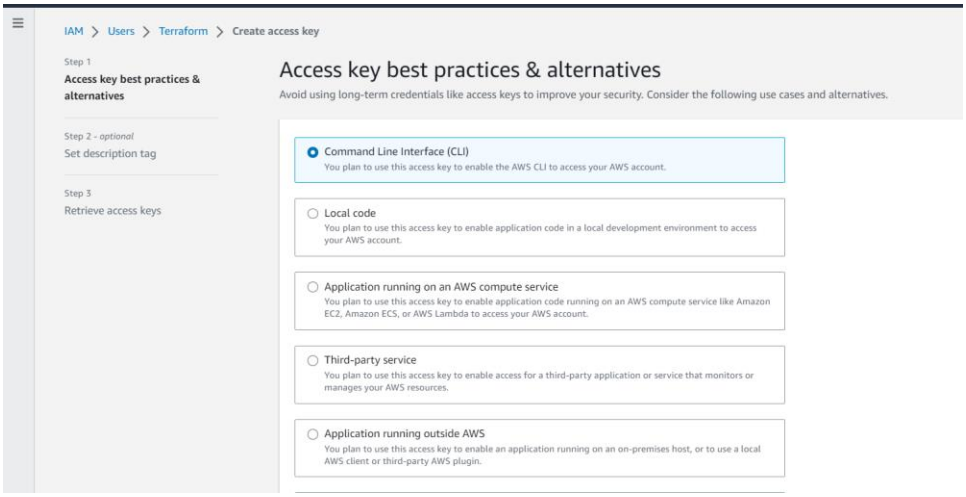


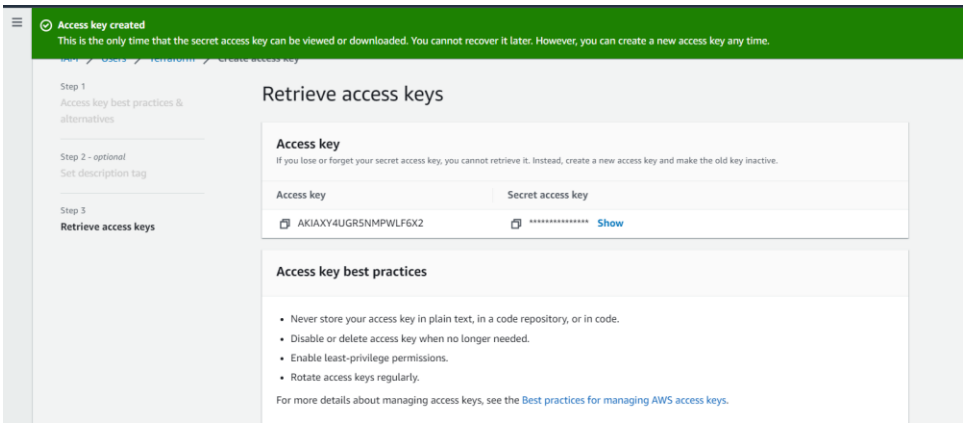




Generate an Access and Secret key







In order to connect your AWS account and Terraform, you need the access keys and secret access keys exported to your machine.

There are Two ways , to provide access key

1] Add these credentials inside main.tf files

2] Export the access key and secret access key

export AWS_ACCESS_KEY_ID=<access key>
export AWS_SECRET_ACCESS_KEY=<secret access key>

Install required providers
You Want any specific version, then you can use required providers block and specify version also.

```
terraform {
 required_providers {
        aws = {
        source = "hashicorp/aws"
        version = "~> 4.16"
}
}
        required version = ">= 1.2.0"
}
```

Add the region where you want your instances to be

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

Task-01

- Provision an AWS EC2 instance using Terraform

Hint:

```
resource "aws_instance" "aws_ec2_test" {
        count = 4
        ami = "ami-08c40ec9ead489470"
        instance_type = "t2.micro"
        tags = {
     Name = "TerraformTestServerInstance"
  }
}
```

Create a main2.tf file

Here we added AWS credentials inside main2.tf file.

```
ubuntu@ip-172-31-85-217:~$ cat main2.tf
provider "aws" {
region = "us-east-1"
access_key =
secret_key =
}

resource "aws_instance" "ec2_example" {
ami = "ami-007855ac798b5175e"
instance_type = "t2.micro"
tags = {
        Name = "Terraform EC2"
}
}
ubuntu@ip-172-31-85-217:~$ terraform validate
```

Run Terraform init command

```
ubuntu@ip-172-31-85-217:~$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.67.0

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.
ubuntu@ip-172-31-85-217:~$ 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.ec2_example will be created
  + resource "aws_instance" "ec2_example" {
      + ami                                  = "ami-007855ac798b5175e"
      + 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)
```

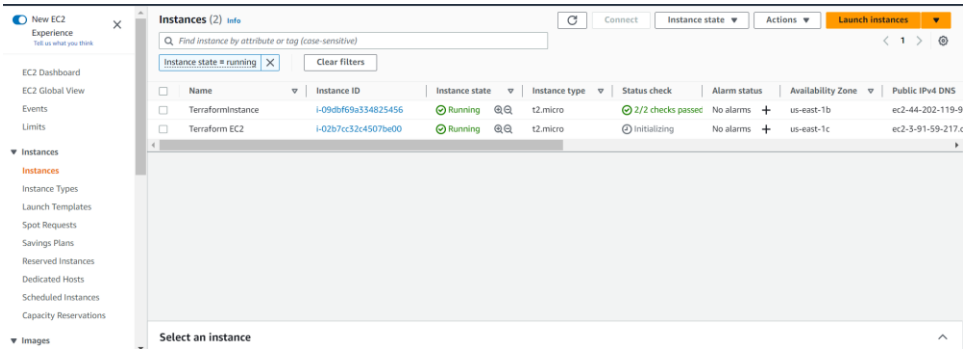
Validate the terraform changes

```
ubuntu@ip-172-31-85-217:~$ terraform validate
Success! The configuration is valid.
```

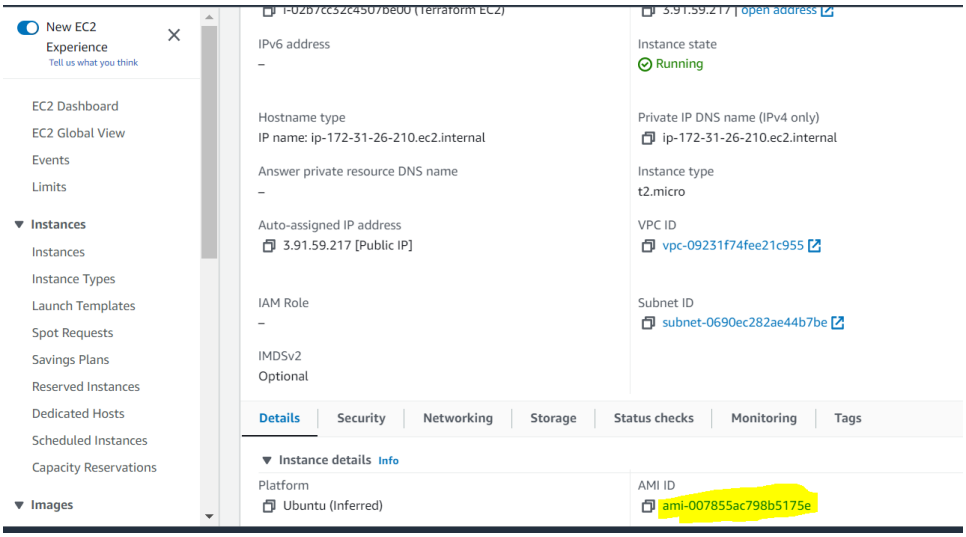
Apply the Terraform files.

```
ubuntu@ip-172-31-85-217:~$ 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.ec2_example will be created
  + resource "aws_instance" "ec2_example" {
      + ami                                  = "ami-007855ac798b5175e"
      + 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_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)
```

You can see, Our EC2 Instance is created



Check the AMI ID



Happy Learning :)