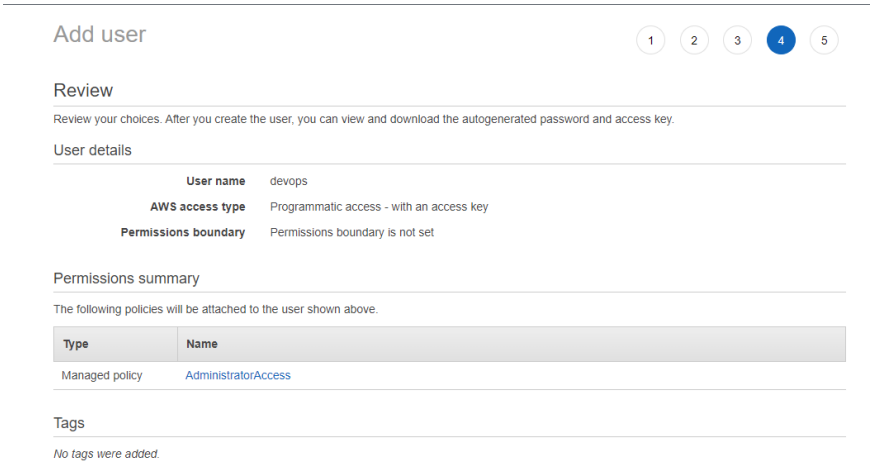
Project On Terraform

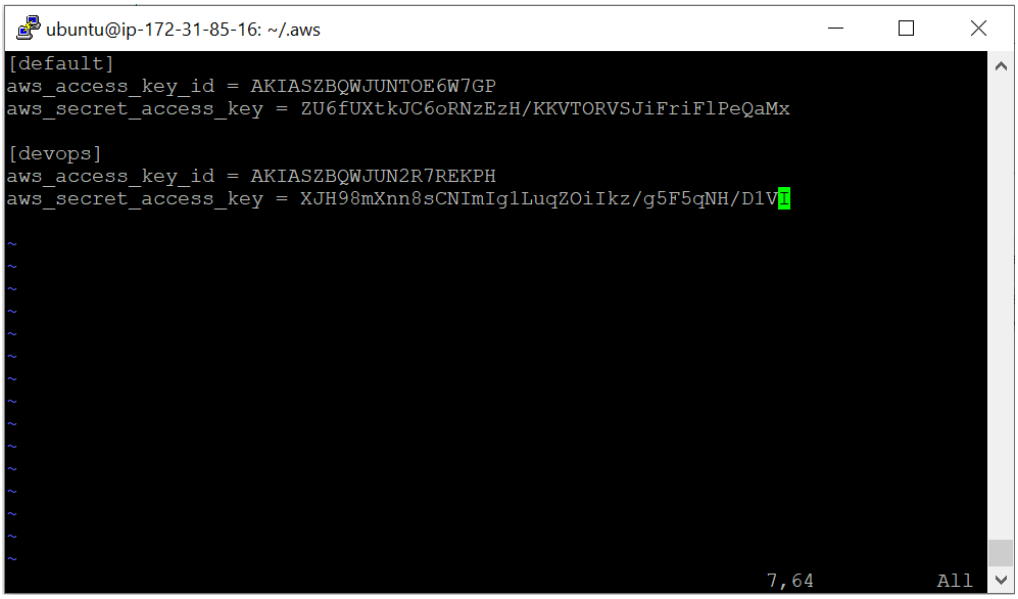
Name: Niranjan Chavan Email: Niranjan_Chavan@epam.com

Add aws credentials with custom profile

https://github.com/niranjan-cell/Assignment/tree/main/Terraform

$ aws configure –profile devops





Make some folders and place your code.

$ mkdir /usr/local/terraform-demo
$ cd /usr/local/terraform-demo
$ mkdir example1
$ cd example1

1. **Create a tf file**
   $ vim example1.tf

```
provider "aws" {
    region = "us-east-1"
    access_key = "AKIASZBQWJUN2NITEKIG"
    secret_key = "BFIyt0BsH4V/3KJxdlHGgBQz4cLu4lPzKBNm+n4D"
}
resource "aws_instance" "instance01" {
    ami = "ami-09d56f8956ab235b3"
    instance_type = "t2.micro"
}
```

$ terraform init

```
ubuntu@ip-172-31-85-16:/usr/local/terraform-demo/example1$ sudo terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.18.0

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-172-31-85-16:/usr/local/terraform-demo/example1$
```

$ terraform plan

```
      }

    + maintenance_options {
        + auto_recovery = (known after apply)
      }

    + metadata_options {
        + http_endpoint               = (known after apply)
        + http_put_response_hop_limit = (known after apply)
        + http_tokens                 = (known after apply)
        + instance_metadata_tags      = (known after apply)
      }

    + network_interface {
        + delete_on_termination = (known after apply)
        + device_index          = (known after apply)
        + network_card_index    = (known after apply)
        + network_interface_id  = (known after apply)
      }

    + root_block_device {
        + 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.

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.
```

$ terraform 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.instance01: Creating...
aws_instance.instance01: Still creating... [10s elapsed]
aws_instance.instance01: Still creating... [20s elapsed]
aws_instance.instance01: Still creating... [30s elapsed]
aws_instance.instance01: Creation complete after 31s [id=i-06e1e87fa0a0bdffd]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-85-16:/usr/local/terraform-demo/example1$
```

2. Example 2 –

AWS keys within tf

```
provider "aws" {
    region = "us-east-1"
    profile = "devops"

}
resource "aws_instance" "instance01" {
    ami = "ami-09d56f8956ab235b3"
    instance_type = "t2.micro"
    tags = {
    "Name"        = "web-server"
    "environment" = "dev"
}
}
resource "aws_instance" "instance02" {
    ami = "ami-09d56f8956ab235b3"
    instance_type = "t2.micro"
    tags = {
    "Name"        = "app-server"
    "environment" = "stage"
    }
}
```

$

$ terraform plan

$ terraform apply

```
    }

Plan: 2 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.instance02: Creating...
aws_instance.instance01: Creating...
aws_instance.instance02: Still creating... [10s elapsed]
aws_instance.instance01: Still creating... [10s elapsed]
aws_instance.instance02: Still creating... [20s elapsed]
aws_instance.instance01: Still creating... [20s elapsed]
aws_instance.instance02: Still creating... [30s elapsed]
aws_instance.instance01: Still creating... [30s elapsed]
aws_instance.instance02: Creation complete after 32s [id=i-0bab6f8fd6a699d48]
aws_instance.instance01: Creation complete after 33s [id=i-0fdbf069fe6e28f01]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-85-16:~/example2$
```

3. Example 3 - Destroy

$ terraform show

```
ubuntu@ip-172-31-85-16:~/example2$ terraform show
# aws_instance.instance01:
resource "aws_instance" "instance01" {
    ami                                  = "ami-09d56f8956ab235b3"
    arn                                  = "arn:aws:ec2:us-east-1:191228169499:instance/i-0fdbf069fe6e28f01"
    associate_public_ip_address          = true
    availability_zone                    = "us-east-1d"
    cpu_core_count                       = 1
    cpu_threads_per_core                 = 1
    disable_api_termination              = false
    ebs_optimized                        = false
    get_password_data                    = false
    hibernation                          = false
    id                                   = "i-0fdbf069fe6e28f01"
    instance_initiated_shutdown_behavior = "stop"
    instance_state                       = "running"
    instance_type                        = "t2.micro"
    ipv6_address_count                   = 0
    ipv6_addresses                       = []
    monitoring                           = false
    primary_network_interface_id         = "eni-05842c8bc35e38b36"
    private_dns                          = "ip-172-31-95-56.ec2.internal"
    private_ip                           = "172.31.95.56"
    public_dns                           = "ec2-44-204-56-11.compute-1.amazonaws.com"
    public_ip                            = "44.204.56.11"
    secondary_private_ips                = []
    security_groups                      = [
        "default",
    ]
    source_dest_check                    = true
    subnet_id                            = "subnet-08a41c4ead5fb80bc"
    tags                                 = {
        "Name"        = "web-server"
        "environment" = "dev"
    }
    tags_all                             = {
        "Name"        = "web-server"
        "environment" = "dev"
    }
    tenancy                              = "default"
    vpc_security_group_ids               = [
        "sg-09a585e3dec818134",
    ]
```

$ terraform destroy

```
Plan: 0 to add, 0 to change, 2 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.instance01: Destroying... [id=i-0fdbf069fe6e28f01]
aws_instance.instance02: Destroying... [id=i-0bab6f8fd6a699d48]
aws_instance.instance01: Still destroying... [id=i-0fdbf069fe6e28f01, 10s elapsed]
aws_instance.instance02: Still destroying... [id=i-0bab6f8fd6a699d48, 10s elapsed]
aws_instance.instance01: Still destroying... [id=i-0fdbf069fe6e28f01, 20s elapsed]
aws_instance.instance02: Still destroying... [id=i-0bab6f8fd6a699d48, 20s elapsed]
aws_instance.instance01: Destruction complete after 30s
aws_instance.instance02: Still destroying... [id=i-0bab6f8fd6a699d48, 30s elapsed]
aws_instance.instance02: Destruction complete after 30s

Destroy complete! Resources: 2 destroyed.
```

4. Example 4 - Resource Dependency // Implicit

```
provider "aws" {
    profile = "devops"
    region = "us-east-1"
}
resource "aws_instance" "instance01" {
    ami = "ami-09d56f8956ab235b3"
    instance_type = "t2.micro"
    tags = {
    "Name"        = "web-server"
    "environment" = "dev"
}
}
resource "aws_eip" "newIP" {
    instance = "${aws_instance.instance01.id}"
    vpc = true
}
```

$ terraform plan
$ terraform apply

```
Plan: 2 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.instance01: Creating...
aws_instance.instance01: Still creating... [10s elapsed]
aws_instance.instance01: Still creating... [20s elapsed]
aws_instance.instance01: Still creating... [30s elapsed]
aws_instance.instance01: Creation complete after 32s [id=i-04be837c0db2c6940]
aws_eip.newIP: Creating...
aws_eip.newIP: Creation complete after 2s [id=eipalloc-09e55746ebc353fb2]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-85-16:~/example4$
```

5. Example 5 - Resource Dependency // Implicit & Explicit

```
provider "aws" {
    region = "us-east-1"
    profile = "devops"
}
resource "aws_instance" "instance01" {
    ami = "ami-09d56f8956ab235b3"
    availability_zone = "us-east-1a"
    instance_type = "t2.micro"
    tags = {
    "Name"        = "web-server"
    "environment" = "dev"
    }
depends_on = [aws_ebs_volume.diskSize]
}
resource "aws_ebs_volume" "diskSize" {
    availability_zone = "us-east-1a"
    size = 10
}
resource "aws_volume_attachment" "ebs_add" {
    device_name = "/dev/xvdf"
    volume_id   = aws_ebs_volume.diskSize.id
    instance_id = aws_instance.instance01.id
}
resource "aws_eip" "elastic_ip" {
    instance = aws_instance.instance01.id
    vpc = true
}
```

$ terraform apply

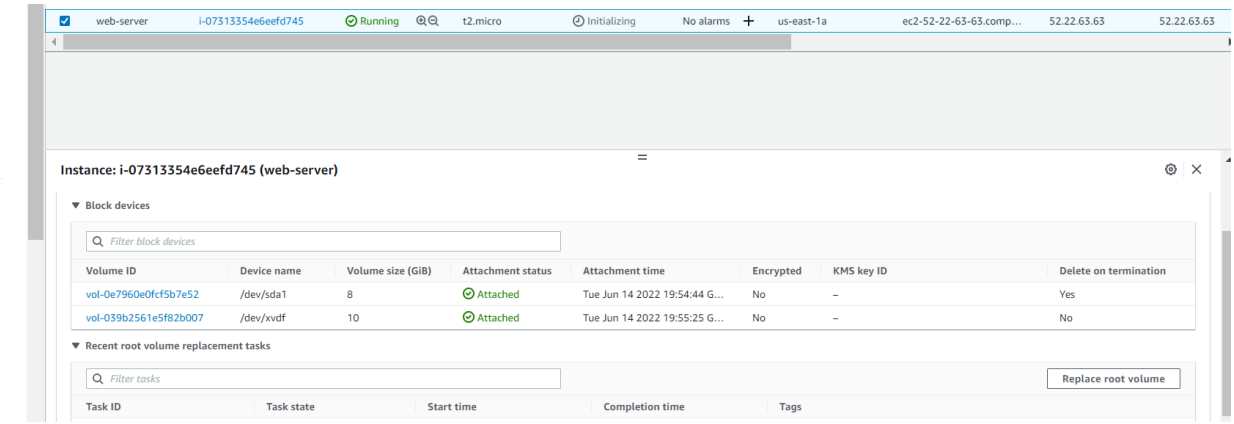
```
Plan: 4 to add, 0 to change, 0 to destroy.

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_ebs_volume.diskSize: Creating...
aws_ebs_volume.diskSize: Still creating... [10s elapsed]
aws_ebs_volume.diskSize: Creation complete after 10s [id=vol-039b2561e5f82b007]
aws_instance.instance01: Creating...
aws_instance.instance01: Still creating... [10s elapsed]
aws_instance.instance01: Still creating... [20s elapsed]
aws_instance.instance01: Still creating... [30s elapsed]
aws_instance.instance01: Creation complete after 32s [id=i-07313354e6eefd745]
aws_volume_attachment.ebs_add: Creating...
aws_eip.elastic_ip: Creating...
aws_eip.elastic_ip: Creation complete after 1s [id=eipalloc-0cd6308216405c4a5]
aws_volume_attachment.ebs_add: Still creating... [10s elapsed]
aws_volume_attachment.ebs_add: Still creating... [20s elapsed]
aws_volume_attachment.ebs_add: Creation complete after 21s [id=vai-1089015779]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-85-16:~/example5$
```



6. Example 6 - Provisioner

```
provider "aws" {
    profile = "devops"
    region = "us-east-1"
}
resource "aws_instance" "instance01" {
    ami = "ami-09d56f8956ab235b3"
    instance_type = "t2.micro"
    tags = {
    "Name"        = "web-server"
    "environment" = "dev"
}
provisioner "local-exec" {
    command = "echo ${aws_instance.instance01.id} > instance_id.txt"
}
}
```

$ terraform apply –auto-approve

```
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.instance01: Creating...
aws_instance.instance01: Still creating... [10s elapsed]
aws_instance.instance01: Still creating... [20s elapsed]
aws_instance.instance01: Still creating... [30s elapsed]
aws_instance.instance01: Provisioning with 'local-exec'...
aws_instance.instance01 (local-exec): Executing: ["/bin/sh" "-c" "echo i-08a15b947fa0b1e04 > instance_id.txt"]
aws_instance.instance01: Creation complete after 31s [id=i-08a15b947fa0b1e04]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
ubuntu@ip-172-31-85-16:~/example6$
```