1.Creating Basic EC2 Instance by using terraform and hiding Aws Access and Secret Keys:

Main.tf

resource "aws\_instance" "provider" {

    ami = "ami-0c7217cdde317cfec"

    instance\_type = "t2.micro"

    key\_name = "Venkat"

    tags = {

        Name = "Terraform-EC2"

    }

}

Provider.tf

provider "aws" {

    region = "us-east-1"

    profile = "Venkat"

}

2.Create Basic EC2 instance and assigns values in variable.tf file

Main.tf

#creating Ec2 Instance and by assigning values in variables

resource "aws\_instance" "myc2" {

    ami = var.ami

    instance\_type = var.instance\_type

    key\_name = var.key\_name

    count = 4

    tags = {

        Name = "Venkat"

    }

}

Provider.tf

terraform {

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 5.0"

    }

  }

}

provider "aws" {

  region = "var.region"

}

Variable.tf

variable "region" {

    default = "us-east-1"

}

variable "ami" {

    default = ""

}

variable "instance\_type" {

    default = "t2.micro"

}

variable "key\_name" {

    default = ""

}

3.Terraform Modules:

--------🡪 create Vcube1 directory in it you have to create

Main.tf

terraform {

  required\_version = ">=0.12"

}

resource "aws\_instance" "ec2\_example" {

  ami                    = "ami-09d8b83b58eabf58b"

  instance\_type          = "t3.micro"

  key\_name               = "default"

  vpc\_security\_group\_ids = [aws\_security\_group.main.id]

  user\_data              = <<-EOF

            #!/bin/bash

            sudo su

            yum update -y

            yum install -y httpd

            cd /var/www/html

            wget https://github.com/azeezsalu/techmax/archive/refs/heads/main.zip

            unzip main.zip

            cp -r techmax-main/\* /var/www/html/

            rm -rf techmax-main main.zip

            systemctl enable httpd

            systemctl start httpd

      EOF

}

resource "aws\_security\_group" "main" {

  name        = "EC2-webserver-SG-1"

  description = "Webserver for EC2 Instances"

  ingress {

    from\_port   = 80

    protocol    = "TCP"

    to\_port     = 80

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 22

    protocol    = "TCP"

    to\_port     = 22

    cidr\_blocks = ["0.0.0.0/0"]

  }

  egress {

    from\_port   = 0

    protocol    = "-1"

    to\_port     = 0

    cidr\_blocks = ["0.0.0.0/0"]

  }

}

Output.tf

output "public\_ip" {

  value = aws\_instance.ec2\_example.public\_ip

}

---------🡪 create another directory Vcube2

Main.tf

terraform {

  required\_version = ">=0.12"

}

resource "aws\_instance" "ec2\_example" {

  ami                    = "ami-09d8b83b58eabf58b"

  instance\_type          = "t3.micro"

  key\_name               = ""

  vpc\_security\_group\_ids = [aws\_security\_group.main.id]

  user\_data = <<-EOF

      #!/bin/bash

      sudo su

      yum update -y

      amazon-linux-extras install nginx1 -y

      systemctl enable nginx

      systemctl start nginx

      systemctl status nginx

      sudo echo <!DOCTYPE html> <html> <head> <meta name="viewport" content="width=device-width, initial-scale=1"> <title>youtube Allow Fullscreen</title> </head> <body> <!--Need Internet Connection--> <!--Fullscreen allow--> <iframe width="420" height="315" src="https://www.vcube.com/embed/OK7fy40Ai6A" allowfullscreen></iframe> </body> </html>" > /usr/share/nginx/html/index.html

      systemctl restart nginx

      EOF

}

resource "aws\_security\_group" "main" {

  name        = "EC2-webserver-SG-2"

  description = "Webserver for EC2 Instances"

  ingress {

    from\_port   = 8080

    protocol    = "TCP"

    to\_port     = 8080

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 80

    protocol    = "TCP"

    to\_port     = 80

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 22

    protocol    = "TCP"

    to\_port     = 22

    cidr\_blocks = ["0.0.0.0/0"]

  }

  egress {

    from\_port   = 0

    protocol    = "-1"

    to\_port     = 0

    cidr\_blocks = ["0.0.0.0/0"]

  }

}

output.tf

output "public\_dns" {

  value = aws\_instance.ec2\_example.public\_dns

}

After that you have to create main.tf and variable.tf outside the module

Main.tf

provider "aws" {

  region  = var.region

  profile = var.profile

}

module "webserver-1" {

  source = ".//module-1"

}

module "webserver-2" {

  source = ".//module-2"

}

Variable.tf

variable "region" {

  default = "ap-south-2"

}

variable "profile" {

  default = ""

}

terraform init /plan/apply

Copy Public\_IP and paste in Browser you see the result

4.Create S3 bucket and upload Multiple files in S3 bucket

#Create S3 Bucket & Upload files using terraform

provider "aws" {

    region = "us-east-1"

}

resource "aws\_s3\_bucket" "mybucket" {

  bucket = "my-bucket-31-1-2024"

  tags = {

    Environment = "dev"

  }

}

resource "aws\_s3\_object" "einstein" {

  bucket = "my-bucket-31-1-2024"

  key    = "einstein.jpg"

  source = "C:\\Users\\singa\\OneDrive\\Pictures\\einstein.jpg"

}

resource "aws\_s3\_bucket\_versioning" "versioning" {

  bucket = "my-bucket-31-1-2024"

  versioning\_configuration {

    status = "Enabled"

  }

}

resource "null\_resource" "multiple-files-uploding" {

  provisioner "local-exec" {

    command = "aws s3 sync D:\\my-folder s3://my-bucket-31-1-2024"

  }

}

5.Create VPC and Load Balancer using terraform:

Main.tf

resource "aws\_vpc" "myvpc" {

  cidr\_block              = var.vpc\_cidr

  instance\_tenancy        = "default"

  enable\_dns\_hostnames    = true

  tags      = {

    Name    = "myvpc"

  }

}

resource "aws\_subnet" "sub1" {

  vpc\_id                  = aws\_vpc.myvpc.id

  cidr\_block              = var.public\_sub1\_cidr

  availability\_zone       = "us-east-1a"

  map\_public\_ip\_on\_launch = true

  tags      = {

    Name    = "sub1"

  }

}

resource "aws\_subnet" "sub2" {

  vpc\_id                  = aws\_vpc.myvpc.id

  cidr\_block              = var.public\_sub2\_cidr

  availability\_zone       = "us-east-1b"

  map\_public\_ip\_on\_launch = true

  tags      = {

    Name    = "sub2"

  }

}

resource "aws\_internet\_gateway" "igw" {

  vpc\_id = aws\_vpc.myvpc.id

  tags      = {

    Name    = "igw"

  }

}

resource "aws\_route\_table" "RT" {

  vpc\_id = aws\_vpc.myvpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.igw.id

  }

  tags      = {

    Name    = "RT"

  }

}

resource "aws\_route\_table\_association" "rta1" {

  subnet\_id      = aws\_subnet.sub1.id

  route\_table\_id = aws\_route\_table.RT.id

}

resource "aws\_route\_table\_association" "rta2" {

  subnet\_id      = aws\_subnet.sub2.id

  route\_table\_id = aws\_route\_table.RT.id

}

resource "aws\_security\_group" "webSg" {

  name   = "web"

  vpc\_id = aws\_vpc.myvpc.id

  ingress {

    description = "HTTP from VPC"

    from\_port   = 80

    to\_port     = 80

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    description = "SSH"

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  egress {

    from\_port   = 0

    to\_port     = 0

    protocol    = "-1"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  tags = {

    Name = "web-Sg"

  }

}

resource "aws\_instance" "webserver1" {

  ami                    = "ami-0c7217cdde317cfec"

  instance\_type          = "t2.micro"

  key\_name               = "Venkat"

  vpc\_security\_group\_ids = [aws\_security\_group.webSg.id]

  subnet\_id              = aws\_subnet.sub1.id

  user\_data              = base64encode(file("user.sh"))

}

resource "aws\_instance" "webserver2" {

  ami                    = "ami-0c7217cdde317cfec"

  instance\_type          = "t2.micro"

  key\_name               = "Venkat"

  vpc\_security\_group\_ids = [aws\_security\_group.webSg.id]

  subnet\_id              = aws\_subnet.sub2.id

  user\_data              = base64encode(file("user1.sh"))

}

#create a Load Balancer

resource "aws\_lb" "myalb" {

  name               = "myalb"

  internal           = false

  load\_balancer\_type = "application"

  security\_groups = [aws\_security\_group.webSg.id]

  subnets         = [aws\_subnet.sub1.id, aws\_subnet.sub2.id]

  tags = {

    Name = "myalb"

  }

}

resource "aws\_lb\_target\_group" "tg" {

  name     = "myTG"

  port     = 80

  protocol = "HTTP"

  vpc\_id   = aws\_vpc.myvpc.id

  health\_check {

    path = "/"

    port = "traffic-port"

  }

}

resource "aws\_lb\_target\_group\_attachment" "attach1" {

  target\_group\_arn = aws\_lb\_target\_group.tg.arn

  target\_id        = aws\_instance.webserver1.id

  port             = 80

}

resource "aws\_lb\_target\_group\_attachment" "attach2" {

  target\_group\_arn = aws\_lb\_target\_group.tg.arn

  target\_id        = aws\_instance.webserver2.id

  port             = 80

}

resource "aws\_lb\_listener" "listener" {

  load\_balancer\_arn = aws\_lb.myalb.arn

  port              = 80

  protocol          = "HTTP"

  default\_action {

    target\_group\_arn = aws\_lb\_target\_group.tg.arn

    type             = "forward"

  }

}

output "loadbalancerdns" {

  value = aws\_lb.myalb.dns\_name

}

Variable.tf

variable "vpc\_cidr" {

  description = "vpc\_cidr"

  type        = string

  default     = "10.0.0.0/16"

}

variable "public\_sub1\_cidr" {

  description = "public sub1"

  type        = string

  default     = "10.0.1.0/24"

}

variable "public\_sub2\_cidr" {

  description = "public sub2"

  type        = string

  default     = "10.0.2.0/24"

}

Provider.tf

provider "aws" {

    region = "us-east-1"

    profile = "Venkat"

}

User.sh

#! /bin/bash

sudo -i

sudo apt-get update

sudo apt-get install -y apache2

sudo systemctl start apache2

sudo systemctl enable apache2

echo "The page was created by the user Vcube" | sudo tee /var/www/html/index.html

User1.sh

#! /bin/bash

sudo -i

sudo apt-get update

sudo apt-get install -y apache2

sudo systemctl start apache2

sudo systemctl enable apache2

echo "The page was created by the user Venkatesh" | sudo tee /var/www/html/index.html

6.3-Tier VPC by using Terraform:

Main.tf

# create vpc

resource "aws\_vpc" "My\_Vpc" {

  cidr\_block              = var.vpc\_cidr

  instance\_tenancy        = "default"

  enable\_dns\_hostnames    = true

  tags      = {

    Name    = "My\_Vpc"

  }

}

# create internet gateway and attach it to vpc

resource "aws\_internet\_gateway" "My\_IGT" {

  vpc\_id    = aws\_vpc.My\_Vpc.id

  tags      = {

    Name    = "My\_IGT"

  }

}

# create public subnet1

resource "aws\_subnet" "public\_subnet1" {

  vpc\_id                  = aws\_vpc.My\_Vpc.id

  cidr\_block              = var.public\_subnet1\_cidr

  availability\_zone       = "us-east-1a"

  map\_public\_ip\_on\_launch = true

  tags      = {

    Name    = "public\_subnet1"

  }

}

# create public subnet2

resource "aws\_subnet" "public\_subnet2" {

  vpc\_id                  = aws\_vpc.My\_Vpc.id

  cidr\_block              = var.public\_subnet2\_cidr

  availability\_zone       = "us-east-1b"

  map\_public\_ip\_on\_launch = true

  tags      = {

    Name    = "public\_subnet2"

  }

}

# create route table and add public route

resource "aws\_route\_table" "public\_route\_table" {

  vpc\_id       = aws\_vpc.My\_Vpc.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.My\_IGT.id

  }

  tags       = {

    Name     = "public\_route\_table"

  }

}

# associate public subnet1 to "public route table"

# terraform aws associate subnet with route table

resource "aws\_route\_table\_association" "public\_subnet1\_route\_table\_association" {

  subnet\_id           = aws\_subnet.public\_subnet1.id

  route\_table\_id      = aws\_route\_table.public\_route\_table.id

}

# associate public subnet2 to "public route table"

# terraform aws associate subnet with route table

resource "aws\_route\_table\_association" "public\_subnet\_2\_route\_table\_association" {

  subnet\_id           = aws\_subnet.public\_subnet2.id

  route\_table\_id      = aws\_route\_table.public\_route\_table.id

}

# create private app subnet1

# terraform aws create subnet

resource "aws\_subnet" "private\_app\_subnet1" {

  vpc\_id                   = aws\_vpc.My\_Vpc.id

  cidr\_block               = var.private\_app\_subnet1\_cidr

  availability\_zone        = "us-east-1a"

  map\_public\_ip\_on\_launch  = false

  tags      = {

    Name    = "private\_app\_subnet1"

  }

}

# create private app subnet2

# terraform aws create subnet

resource "aws\_subnet" "private\_app\_subnet2" {

  vpc\_id                   = aws\_vpc.My\_Vpc.id

  cidr\_block               = var.private\_app\_subnet2\_cidr

  availability\_zone        = "us-east-1b"

  map\_public\_ip\_on\_launch  = false

  tags      = {

    Name    = "private\_app\_subnet2"

  }

}

# create private data subnet1

# terraform aws create subnet

resource "aws\_subnet" "private\_data\_subnet1" {

  vpc\_id                   = aws\_vpc.My\_Vpc.id

  cidr\_block               = var.private\_data\_subnet1\_cidr

  availability\_zone        = "us-east-1a"

  map\_public\_ip\_on\_launch  = false

  tags      = {

    Name    = "private\_data\_subnet1"

  }

}

# create private data subnet2

# terraform aws create subnet

resource "aws\_subnet" "private\_data\_subnet2" {

  vpc\_id                   = aws\_vpc.My\_Vpc.id

  cidr\_block               = var.private\_data\_subnet2\_cidr

  availability\_zone        = "us-east-1b"

  map\_public\_ip\_on\_launch  = false

  tags      = {

    Name    = "private\_data\_subnet2"

  }

}

Provider.tf

provider "aws" {

    region = "us-east-1"

    profile = "Venkat"

}

Variable.tf

# create vpc cidr blocks

variable "vpc\_cidr" {

  description = "vpc\_cidr"

  type        = string

  default     = "10.0.0.0/16"

}

variable "public\_subnet1\_cidr" {

  description = "public subnet1"

  type        = string

  default     = "10.0.1.0/24"

}

variable "public\_subnet2\_cidr" {

  description = "public subnet2"

  type        = string

  default     = "10.0.2.0/24"

}

variable "private\_app\_subnet1\_cidr" {

  description = "private app subnet1"

  type        = string

  default     = "10.0.3.0/24"

}

variable "private\_app\_subnet2\_cidr" {

  description = "private app subnet2"

  type        = string

  default     = "10.0.4.0/24"

}

variable "private\_data\_subnet1\_cidr" {

  description = "private data subnet1"

  type        = string

  default     = "10.0.5.0/24"

}

variable "private\_data\_subnet2\_cidr" {

  description = "private data subnet2"

  type        = string

  default     = "10.0.6.0/24"

}

7.How to store our terraform.tfstate file in our S3 bucket by using terraform Backend concept:

Method1:

Step1: First create directory mkdir terraform

Step2: Go to aws account create S3 Bucket

Main.tf

resource "aws\_instance" "terraform\_backend" {

    ami = "ami-0c7217cdde317cfec"

    availability\_zone = "us-east-1a"

    instance\_type = "t2.micro"

    key\_name = "vscode"

    tags = {

        Name ="terraform\_backend"

    }

}

Provider.tf

terraform {

  required\_providers {

    aws = {

      source = "hashicorp/aws"

      version = "5.34.0"

    }

  }

}

provider "aws" {

  profile = "Venkat"

  region = "us-east-1"

}

Backend.tf

terraform {

  backend "s3" {

    bucket = "venkatesh-bucket-1-2-2024"

    key    = "C:\\Users\\singa\\OneDrive\\Desktop\\terraform backend\\terraform.tfstate"

    region = "us-east-1"

  }

}

Method2:

Create directory in the same terraform with mkdir terra123

We can import our terraform.tfstate from our ec2 instance from desktop

Main.tf

resource "aws\_instance" "myec2" {

}

Provider.tf

terraform {

  required\_providers {

    aws = {

      source = "hashicorp/aws"

      version = "5.34.0"

    }

  }

}

provider "aws" {

  profile = "Venkat"

  region = "us-east-1"

}

terraform import aws\_instance.myc2 EC2-id (i-0a37760…….)

Next remove terraformtf.state otherwise it won’t import in next time it show error (Before importing this resource,please create its configuration in the root module)

Main.tf

resource "aws\_s3\_bucket" "my\_bucket" {

  bucket = "venkatesh-bucket-1-2-2024"

  tags = {

    Name        = "my\_bucket"

    Environment = "Dev"

  }

}

provider.tf

terraform {

  required\_providers {

    aws = {

      source = "hashicorp/aws"

      version = "5.34.0"

    }

  }

}

provider "aws" {

  profile = "Venkat"

  region = "us-east-1"

}

terraform import aws\_s3\_bucket.my\_bucket bucket Name(venkatesh-bucket-1-2-2024)

You will see the terraform.tfstate file importing from S3 Bucket

8.Create EC2 Instance without AMI ID:

Ami-resource.tf

# Get latest AMI ID for Amazon Linux2 OS

data "aws\_ami" "amzlinux" {

  most\_recent = true

  owners = [ "amazon" ]

  filter {

    name = "name"

    values = [ "amzn2-ami-hvm-\*-gp2" ]

  }

  filter {

    name = "root-device-type"

    values = [ "ebs" ]

  }

  filter {

    name = "virtualization-type"

    values = [ "hvm" ]

  }

  filter {

    name = "architecture"

    values = [ "x86\_64" ]

  }

}

EC2-instance.tf

resource "aws\_instance" "my-ec2" {

  ami           = "data.aws\_ami.amzlinux"

  instance\_type = "t2.micro"

  key\_name      = "vscode"

  tags = {

    "Name" = "my-ec2"

  }

}