**Autoscaling:**

provider "aws" {

  region     = "us-east-1"

  access\_key = "AKIA4MTWNZNVF6GRIEOW"

  secret\_key = "ClSXEIoyKueYh+BeDWXPXVTNXfDQkHqfJGDgBAQs"

}

# Data sources to retrieve default VPC and subnets

data "aws\_vpc" "default" {

  default = true

}

data "aws\_subnets" "default" {

  filter {

    name   = "vpc-id"

    values = [data.aws\_vpc.default.id]

  }

}

# Security group definition

resource "aws\_security\_group" "SG02" {

  name        = "SG02"

  description = "Allow SSH and HTTP traffic"

  vpc\_id      = data.aws\_vpc.default.id

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 80

    to\_port     = 80

    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"]

  }

}

# Create a launch template

resource "aws\_launch\_template" "example" {

  name\_prefix   = "example-template-"

  image\_id      = "ami-0bb84b8ffd87024d8"  # Replace with your AMI ID

  instance\_type = "t2.micro"

  key\_name      = "Key-govicloud@1"           # Replace with your key pair name

  # Other configuration options such as security groups, user data, etc. can be added here

}

# Create an auto-scaling group

resource "aws\_autoscaling\_group" "example" {

  name                 = "example-asg"

  launch\_template {

    id      = aws\_launch\_template.example.id

    version = "$Latest"

  }

  min\_size             = 1

  max\_size             = 5

  desired\_capacity     = 2

  vpc\_zone\_identifier  = data.aws\_subnets.default.ids  # Use the correct subnet IDs retrieved from the data source

}

**Creating a EC2 with Default VPC, Subnet:**

provider "aws" {

  region     = "us-east-1"

  access\_key = "AKIA4MTWNZNVF6GRIEOW"

  secret\_key = "ClSXEIoyKueYh+BeDWXPXVTNXfDQkHqfJGDgBAQs"

}

# Data sources to retrieve default VPC and subnets

data "aws\_vpc" "default" {

  default = true

}

data "aws\_subnets" "default" {

  filter {

    name   = "vpc-id"

    values = [data.aws\_vpc.default.id]

  }

}

# Security group definition

resource "aws\_security\_group" "SG02" {

  name        = "SG01"

  description = "SSH and HTTP security group"

  vpc\_id      = data.aws\_vpc.default.id

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 80

    to\_port     = 80

    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"]

  }

}

# EC2 instance for Wikipedia server

resource "aws\_instance" "Wikipediaserver" {

  ami                         = "ami-0bb84b8ffd87024d8"

  instance\_type               = "t2.micro"

  associate\_public\_ip\_address = true

  security\_groups             = [aws\_security\_group.SG02.id]

  subnet\_id                   = data.aws\_subnets.default.ids[0]

  user\_data = <<-EOF

              #!/bin/bash

              yum install httpd -y

              service httpd start

              wget wikipedia.com -O /var/www/html/index.html

              EOF

  tags = {

    Name = "Wikipedia server"

  }

}

**Load Balancer:**

provider "aws" {

  region     = "us-east-1"

  access\_key = "AKIA4MTWNZNVF6GRIEOW"

  secret\_key = "ClSXEIoyKueYh+BeDWXPXVTNXfDQkHqfJGDgBAQs"

}

# Data sources to retrieve default VPC and subnets

data "aws\_vpc" "default" {

  default = true

}

data "aws\_subnets" "default" {

  filter {

    name   = "vpc-id"

    values = [data.aws\_vpc.default.id]

  }

}

# Security group definition

resource "aws\_security\_group" "SG01" {

  name        = "SG01"

  description = "SSH and HTTP security group"

  vpc\_id      = data.aws\_vpc.default.id

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

  ingress {

    from\_port   = 80

    to\_port     = 80

    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"]

  }

}

# EC2 instance for Google server

resource "aws\_instance" "googleserver" {

  ami                         = "ami-0bb84b8ffd87024d8"

  instance\_type               = "t2.micro"

  associate\_public\_ip\_address = true

  security\_groups             = [aws\_security\_group.SG01.id]

  subnet\_id                   = data.aws\_subnets.default.ids[0]

  user\_data = <<-EOF

              #!/bin/bash

              yum install httpd -y

              service httpd start

              wget google.com -O /var/www/html/index.html

              EOF

  tags = {

    Name = "Google server"

  }

}

# EC2 instance for Yahoo server

resource "aws\_instance" "yahooServer" {

  ami                         = "ami-07caf09b362be10b8"

  instance\_type               = "t2.micro"

  associate\_public\_ip\_address = true

  security\_groups             = [aws\_security\_group.SG01.id]

  subnet\_id                   = data.aws\_subnets.default.ids[0]

  user\_data = <<-EOF

              #!/bin/bash

              yum install httpd -y

              service httpd start

              wget yahoo.com -O /var/www/html/index.html

              EOF

  tags = {

    Name = "YahooServer"

  }

}

# Load balancer

resource "aws\_lb" "test" {

  name               = "my-load-balancer"

  internal           = false

  load\_balancer\_type = "application"

  security\_groups    = [aws\_security\_group.SG01.id]

  subnets            = data.aws\_subnets.default.ids

  tags = {

    Name = "test-lb"

  }

}

# Target group

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

  name     = "my-target-group"

  port     = 80

  protocol = "HTTP"

  vpc\_id   = data.aws\_vpc.default.id

  health\_check {

    path                = "/"

    interval            = 30

    timeout             = 5

    healthy\_threshold   = 5

    unhealthy\_threshold = 2

    matcher             = "200"

  }

}

# Listener

resource "aws\_lb\_listener" "test" {

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

  port              = 80

  protocol          = "HTTP"

  default\_action {

    type             = "forward"

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

  }

}

# Attach instances to the target group

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

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

  target\_id        = aws\_instance.googleserver.id

  port             = 80

}

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

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

  target\_id        = aws\_instance.yahooServer.id

  port             = 80

}

**VPC Ceation with two subnets, routing table and IGW:**

provider "aws" {

  region  = "us-east-1"

  access\_key = "AKIA4MTWNZNVF6GRIEOW"

  secret\_key = "ClSXEIoyKueYh+BeDWXPXVTNXfDQkHqfJGDgBAQs"

}

    resource "aws\_vpc" "VPC-Auto" {

  cidr\_block       = "10.0.0.0/16"

  tags = {

    Name = "VPC-Auto"

  }

}

resource "aws\_subnet" "Subnet-Public" {

  vpc\_id     = aws\_vpc.VPC-Auto.id

  cidr\_block = "10.0.1.0/24"

  tags = {

    Name = "Subnet-Public"

  }

}

resource "aws\_subnet" "Subnet-Private" {

  vpc\_id     = aws\_vpc.VPC-Auto.id

  cidr\_block = "10.0.2.0/24"

  tags = {

    Name = "Subnet-Private"

  }

}

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

  vpc\_id = aws\_vpc.VPC-Auto.id

  tags = {

    Name = "igw"

  }

}

# Create the main route table for the VPC

resource "aws\_route\_table" "main" {

  vpc\_id = aws\_vpc.VPC-Auto.id

   tags = {

    Name = "main"

  }

}

# Associate the private subnet with the main route table

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

  subnet\_id      = aws\_subnet.Subnet-Private.id

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

}

# Create a custom route table for the public subnet with an internet gateway

resource "aws\_route\_table" "custom" {

  vpc\_id = aws\_vpc.VPC-Auto.id

  route {

    cidr\_block = "0.0.0.0/0"

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

  }

   tags = {

    Name = "custom"

  }

}

# Associate the public subnet with the custom route table

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

  subnet\_id      = aws\_subnet.Subnet-Public.id

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

}

**2 X EC2 creation with the above VPC:**

# Define a security group

resource "aws\_security\_group" "SG02" {

  name        = "SG02"

  description = "SSH and ICMP security group"

  vpc\_id      = aws\_vpc.VPC-Auto.id

  // Define your security group rules here

  ingress {

    from\_port   = 22

    to\_port     = 22

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

}

  ingress {

    from\_port   = -1

    to\_port     = -1

    protocol    = "icmp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

}

# Define the EC2 instance

resource "aws\_instance" "PublicServer" {

  ami                    = "ami-07caf09b362be10b8"

  instance\_type          = "t2.micro"

  associate\_public\_ip\_address = true

  subnet\_id              = aws\_subnet.Subnet-Public.id

  security\_groups        = [aws\_security\_group.SG02.id]

  // User data script

  user\_data = <<-EOF

              #!/bin/bash

              sudo sed -i 's/#PermitRootLogin yes/PermitRootLogin yes/' /etc/ssh/sshd\_config

              sudo sed -i 's/PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd\_config

              sudo service sshd restart

              sudo echo 'root:Govi@123' | chpasswd

              hostname "PublicServer"

              EOF

  tags = {

    Name = "PublicServer"

  }

}

# Define the EC2 instance in a private subnet

resource "aws\_instance" "PrivateServer" {

  ami                    = "ami-07caf09b362be10b8"

  instance\_type          = "t2.micro"

  associate\_public\_ip\_address = true

  subnet\_id              = aws\_subnet.Subnet-Private.id

  security\_groups        = [aws\_security\_group.SG02.id]

  // User data script

  user\_data = <<-EOF

              #!/bin/bash

              sudo sed -i 's/#PermitRootLogin yes/PermitRootLogin yes/' /etc/ssh/sshd\_config

              sudo sed -i 's/PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd\_config

              sudo service sshd restart

              sudo echo 'root:Govi@123' | chpasswd

              hostname "PrivateServer"

              EOF

  tags = {

    Name = "PrivateServer"

  }

}