```
# Data source for availability zones
data "aws_availability_zones" "available" {}

# Provider
provider "aws" {
  region = "us-east-1"
}

# Variables
variable "vpc_cidr" {
  default = "10.0.0.0/16"
}

variable "subnet_cidr" {
  default = ["10.0.1.0/24", "10.0.2.0/24"]
}

variable "instance_type" {
  default = "t3.micro"
}

variable "ami_id" {
  default = "ami-0182f373e66f89c85" # Provided AMI ID
}

# VPC
resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr

  tags = {
    Name = "pluralsight-vpc"
  }
}

# Internet Gateway
resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "pluralsight-igw"
  }
}

# Route Table
```

```
resource "aws_route_table" "rt" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gw.id
  }

  tags = {
    Name = "pluralsight-rt"
  }
}

# Subnets
resource "aws_subnet" "subnet" {
  count             = length(var.subnet_cidr)
  vpc_id            = aws_vpc.main.id
  cidr_block        = element(var.subnet_cidr, count.index)
  availability_zone = element(data.aws_availability_zones.available.names, count.index)
  map_public_ip_on_launch = true

  tags = {
    Name = "pluralsight-subnet-${count.index + 1}"
  }
}

# Route Table Association
resource "aws_route_table_association" "a" {
  count          = length(var.subnet_cidr)
  subnet_id      = element(aws_subnet.subnet[*].id, count.index)
  route_table_id = aws_route_table.rt.id
}

# Security Group
resource "aws_security_group" "allow_http" {
  vpc_id = aws_vpc.main.id

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
```

```
  ingress {
    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 = "pluralsight-sg"
  }
}

# Launch Configuration
resource "aws_launch_configuration" "example" {
  name          = "pluralsight-lc"
  image_id      = var.ami_id
  instance_type = var.instance_type
  security_groups = [aws_security_group.allow_http.id]
  associate_public_ip_address = true

  lifecycle {
    create_before_destroy = true
  }
}

# Auto Scaling Group
resource "aws_autoscaling_group" "example" {
  desired_capacity     = 2
  max_size             = 3
  min_size             = 2
  vpc_zone_identifier  = aws_subnet.subnet[*].id
  launch_configuration = aws_launch_configuration.example.id

  tag {
    key                 = "Name"
    value               = "pluralsight-asg"
    propagate_at_launch = true
```

```
  }
}

# Load Balancer
resource "aws_elb" "example" {
  name               = "pluralsight-lb"
  subnets            = aws_subnet.subnet[*].id
  security_groups    = [aws_security_group.allow_http.id]

  listener {
    instance_port     = 80
    instance_protocol = "HTTP"
    lb_port           = 80
    lb_protocol       = "HTTP"
  }

  health_check {
    target              = "HTTP:80/"
    interval            = 30
    timeout             = 5
    healthy_threshold   = 2
    unhealthy_threshold = 2
  }

  tags = {
    Name = "pluralsight-lb"
  }
}

# Attach Auto Scaling Group to Load Balancer
resource "aws_autoscaling_attachment" "asg_attachment" {
  autoscaling_group_name = aws_autoscaling_group.example.name
  elb                    = aws_elb.example.id
}

# CloudWatch Alarm for scaling up
resource "aws_cloudwatch_metric_alarm" "cpu_high" {
  alarm_name          = "cpu_high"
  comparison_operator = "GreaterThanOrEqualToThreshold"
  evaluation_periods  = "2"
  metric_name         = "CPUUtilization"
  namespace           = "AWS/EC2"
  period              = "60"  # 1 minute period
  statistic           = "Average"
```

```
  threshold           = "40"
  alarm_description   = "This metric monitors high CPU utilization"
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.example.name
  }

  alarm_actions = [aws_autoscaling_policy.scale_up.arn]
}

# CloudWatch Alarm for scaling down
resource "aws_cloudwatch_metric_alarm" "cpu_low" {
  alarm_name          = "cpu_low"
  comparison_operator = "LessThanOrEqualToThreshold"
  evaluation_periods  = "2"
  metric_name         = "CPUUtilization"
  namespace           = "AWS/EC2"
  period              = "60"  # 1 minute period
  statistic           = "Average"
  threshold           = "30"
  alarm_description   = "This metric monitors low CPU utilization"
  dimensions = {
    AutoScalingGroupName = aws_autoscaling_group.example.name
  }

  alarm_actions = [aws_autoscaling_policy.scale_down.arn]
}

# Scaling policy to scale up
resource "aws_autoscaling_policy" "scale_up" {
  name                   = "scale_up"
  scaling_adjustment     = 1
  adjustment_type        = "ChangeInCapacity"
  cooldown               = 120  # 2 minutes cooldown
  autoscaling_group_name = aws_autoscaling_group.example.name
}

# Scaling policy to scale down
resource "aws_autoscaling_policy" "scale_down" {
  name                   = "scale_down"
  scaling_adjustment     = -1
  adjustment_type        = "ChangeInCapacity"
  cooldown               = 120  # 2 minutes cooldown
  autoscaling_group_name = aws_autoscaling_group.example.name
}
```