Terraform Code:

A) 1 VPC in us-east-1 region. This should be flexible based on region. If no region is provided this should be built in us-east-1

```
provider "aws" {
  region = var.region
}

variable "region" {
  description = "AWS region where VPC will be created"
  default     = "us-east-1"
}

resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "MyVPC"
  }
}

output "vpc_id" {
  value = aws_vpc.my_vpc.id
}
```

B) Two subnets with high availability supported in 2 zones

```
provider "aws" {
  region = var.region
}

variable "region" {
  description = "AWS region where VPC will be created"
  default     = "us-east-1"
}

resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "MyVPC"
  }
}

resource "aws_subnet" "subnet_a" {
  vpc_id                  = aws_vpc.my_vpc.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = true

  tags = {
    Name = "SubnetA"
  }
}
```

```
resource "aws_subnet" "subnet_b" {
  vpc_id                  = aws_vpc.my_vpc.id
  cidr_block              = "10.0.2.0/24"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = true

  tags = {
    Name = "SubnetB"
  }
}

output "vpc_id" {
  value = aws_vpc.my_vpc.id
}

output "subnet_a_id" {
  value = aws_subnet.subnet_a.id
}

output "subnet_b_id" {
  value = aws_subnet.subnet_b.id
}
```

C) 1 Route table not including default one. Routes should not be routed using local route

```
provider "aws" {
  region = var.region
}

variable "region" {
  description = "AWS region where VPC will be created"
  default     = "us-east-1"
}

resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "MyVPC"
  }
}

resource "aws_subnet" "subnet_a" {
  vpc_id                  = aws_vpc.my_vpc.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = true

  tags = {
    Name = "SubnetA"
  }
}

resource "aws_subnet" "subnet_b" {
  vpc_id                  = aws_vpc.my_vpc.id
```

```
  cidr_block              = "10.0.2.0/24"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = true

  tags = {
    Name = "SubnetB"
  }
}

resource "aws_route_table" "custom_route_table" {
  vpc_id = aws_vpc.my_vpc.id

  tags = {
    Name = "CustomRouteTable"
  }
}

resource "aws_route" "route_a" {
  route_table_id         = aws_route_table.custom_route_table.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.my_igw.id
}

resource "aws_route_table_association" "subnet_a_association" {
  subnet_id      = aws_subnet.subnet_a.id
  route_table_id = aws_route_table.custom_route_table.id
}

resource "aws_route_table_association" "subnet_b_association" {
  subnet_id      = aws_subnet.subnet_b.id
  route_table_id = aws_route_table.custom_route_table.id
}

resource "aws_internet_gateway" "my_igw" {
  vpc_id = aws_vpc.my_vpc.id

  tags = {
    Name = "MyIGW"
  }
}

output "vpc_id" {
  value = aws_vpc.my_vpc.id
}

output "subnet_a_id" {
  value = aws_subnet.subnet_a.id
}

output "subnet_b_id" {
  value = aws_subnet.subnet_b.id
}

output "route_table_id" {
  value = aws_route_table.custom_route_table.id
}
```

D) create autoscaling group which creates two EC2 instances and also create a application load balancer with port is flexible based on application

```
provider "aws" {
  region = var.region
}

variable "region" {
  description = "AWS region where resources will be created"
  default     = "us-east-1"
}

# Create a VPC, subnets, and route table
resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "MyVPC"
  }
}

resource "aws_subnet" "subnet_a" {
  vpc_id                  = aws_vpc.my_vpc.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = true

  tags = {
    Name = "SubnetA"
  }
}

resource "aws_subnet" "subnet_b" {
  vpc_id                  = aws_vpc.my_vpc.id
  cidr_block              = "10.0.2.0/24"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = true

  tags = {
    Name = "SubnetB"
  }
}

resource "aws_route_table" "custom_route_table" {
  vpc_id = aws_vpc.my_vpc.id

  tags = {
    Name = "CustomRouteTable"
  }
}

resource "aws_route" "route_a" {
  route_table_id         = aws_route_table.custom_route_table.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.my_igw.id
}
```

```
resource "aws_route_table_association" "subnet_a_association" {
  subnet_id      = aws_subnet.subnet_a.id
  route_table_id = aws_route_table.custom_route_table.id
}

resource "aws_route_table_association" "subnet_b_association" {
  subnet_id      = aws_subnet.subnet_b.id
  route_table_id = aws_route_table.custom_route_table.id
}

resource "aws_internet_gateway" "my_igw" {
  vpc_id = aws_vpc.my_vpc.id

  tags = {
    Name = "MyIGW"
  }
}

# Create an Auto Scaling Group
resource "aws_launch_configuration" "my_launch_config" {
  name          = "my-launch-config"
  image_id      = "ami-0c94855ba95c71c99"  # Specify your desired AMI ID here
  instance_type = "t2.micro"

  security_groups = [aws_security_group.instance_sg.name]

  lifecycle {
    create_before_destroy = true
  }
}

resource "aws_security_group" "instance_sg" {
  name_prefix = "instance-sg-"

  vpc_id = aws_vpc.my_vpc.id

  egress {
    from_port   = 0
    to_port     = 65535
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_security_group_rule" "instance_ingress" {
  type        = "ingress"
  from_port   = 0
  to_port     = 65535
  protocol    = "tcp"
  security_group_id = aws_security_group.instance_sg.id
  source_security_group_id = aws_security_group.instance_sg.id
}

resource "aws_autoscaling_group" "my_asg" {
  launch_configuration       = aws_launch_configuration.my_launch_config.name
  vpc_zone_identifier        = [aws_subnet.subnet_a.id, aws_subnet.subnet_b.id]
```

```
  min_size                  = 2
  max_size                  = 2
  desired_capacity          = 2
  health_check_grace_period = 300
  health_check_type         = "EC2"
}

# Create an Application Load Balancer
resource "aws_security_group" "alb_sg" {
  name_prefix = "alb-sg-"

  vpc_id = aws_vpc.my_vpc.id

  egress {
    from_port   = 0
    to_port     = 65535
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_security_group_rule" "alb_ingress" {
  type        = "ingress"
  from_port   = 0
  to_port     = var.application_port
  protocol    = "tcp"
  security_group_id = aws_security_group.alb_sg.id
  source_security_group_id = aws_security_group.instance_sg.id
}

variable "application_port" {
  description = "Port on which your application listens"
  default     = 80
}

resource "aws_lb" "my_alb" {
  name               = "my-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.alb_sg.id]
  subnets            = [aws_subnet.subnet_a.id, aws_subnet.subnet_b.id]
  enable_deletion_protection = false

  enable_http2             = true
  idle_timeout             = 60
  enable_deletion_protection = false

  enable_deletion_protection = false

  tags = {
    Name = "MyALB"
  }
}

resource "aws_lb_target_group" "my_target_group" {
  name     = "my-target-group"
  port     = var.application_port
```

```
  protocol = "HTTP"
  vpc_id   = aws_vpc.my_vpc.id

  health_check {
    path = "/"
  }
}

resource "aws_lb_listener" "my_listener" {
  load_balancer_arn = aws_lb.my_alb.arn
  port              = var.application_port
  protocol          = "HTTP"

  default_action {
    type             = "fixed-response"
    status_code      = "200"
    content_type     = "text/plain"
    message_body     = "OK"
  }

  depends_on = [aws_lb_target_group.my_target_group]
}

resource "aws_autoscaling_attachment" "asg_attachment" {
  alb_target_group_arn  = aws_lb_target_group.my_target_group.arn
  autoscaling_group_name = aws_autoscaling_group.my_asg.name
}

output "vpc_id" {
  value = aws_vpc.my_vpc.id
}

output "subnet_a_id" {
  value = aws_subnet.subnet_a.id
}

output "subnet_b_id" {
  value = aws_subnet.subnet_b.id
}

output "alb_dns_name" {
  value = aws_lb.my_alb.dns_name
}
```