Write Terraform script to create highly available infrastructure in AWS. The infra should have1 vpc, 3 subnets setup in 3 different az and 2 instances setup in 2 different subnets

task12.tf

```
# Provider Configuration
provider "aws" {
  region     = "us-west-2"
  access_key = "******************"
  secret_key = "******************"
}

data "aws_vpc" "default" {
  default = true
}

data "aws_subnet" "default" {
  filter {
    name   = "vpc-id"
    values = [data.aws_vpc.default.id]
  }

  filter {
    name   = "default-for-az"
    values = ["true"]
  }

  filter {
    name = "availability-zone"
    values = ["us-west-2a"]
  }
}

resource "aws_security_group" "allow_ssh" {
  name        = "allow_ssh"
  description = "Allow SSH inbound traffic"
  vpc_id      = data.aws_vpc.default.id

  ingress {
    description = "SSH from VPC"
    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"]
  }
}

resource "aws_instance" "task" {
  ami           = "ami-08116b9957a259459"
  instance_type = "t2.micro"
  subnet_id     = data.aws_subnet.default.id
  vpc_security_group_ids = [aws_security_group.allow_ssh.id]

  tags = {
    Name = "task15"
  }
}
```

#terraform init

```
ubuntu@ip-172-31-23-205:~$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.44.0...
- Installed hashicorp/aws v5.44.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```

#terraform plan

```
ubuntu@ip-172-31-23-205:~/tfproj02$ terraform plan
data.aws_vpc.default: Reading...
data.aws_vpc.default: Read complete after 0s [id=vpc-06cb8a4d8255fb255]
data.aws_subnet.default: Reading...
data.aws_subnet.default: Read complete after 0s [id=subnet-0f9dc02b94e8e758b]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with
  + create

Terraform will perform the following actions:

  # aws_instance.task will be created
  + resource "aws_instance" "task" {
      + ami                                  = "ami-08116b9957a259459"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = [
          + "allow_ssh",
        ]
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = "subnet-0f9dc02b94e8e758b"
      + tags                                 = {
          + "Name" = "task15"
        }
      + tags_all                             = {
          + "Name" = "task15"
        }
      + tenancy                              = (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = (known after apply)
    }

  # aws_security_group.allow_ssh will be created
  + resource "aws_security_group" "allow_ssh" {
      + arn                    = (known after apply)
      + description            = "Allow SSH inbound traffic"
      + egress                 = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 0
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "-1"
              + security_groups  = []
              + self             = false
              + to_port          = 0
            },
        ]
      + id                     = (known after apply)
      + ingress                = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = "SSH from VPC"
              + from_port        = 22
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 22
            },
        ]
      + name                   = "allow_ssh"
      + name_prefix            = (known after apply)
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags_all               = (known after apply)
      + vpc_id                 = "vpc-06cb8a4d8255fb255"
    }

Plan: 2 to add, 0 to change, 0 to destroy.
```

#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_security_group.allow_ssh: Creating...
aws_security_group.allow_ssh: Creation complete after 2s [id=sg-09e22709151931322]
aws_instance.task: Creating...
aws_instance.task: Still creating... [10s elapsed]
aws_instance.task: Still creating... [20s elapsed]
aws_instance.task: Still creating... [30s elapsed]
aws_instance.task: Creation complete after 32s [id=i-08dfa6568aca517a8]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Result