Provisioning a Full AWS Network and EC2 Instance

```
my-terraform-project/
├── main.tf
├── variables.tf
├── outputs.tf
└── terraform.tfvars
```

`main.tf` File

main.tf

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

VPC

```
resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr
  tags = {
    Name = "main-vpc"
  }
}
```

Subnet

```
resource "aws_subnet" "main" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = var.subnet_cidr
  availability_zone       = var.availability_zone
  map_public_ip_on_launch = true

  tags = {
    Name = "main-subnet"
  }
}
```

Internet Gateway

```
resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id
  tags = {
    Name = "main-igw"
  }
}
```

Route Table

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

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

  tags = {
    Name = "main-route-table"
  }
}
```

Route Table Association

```
resource "aws_route_table_association" "main" {
  subnet_id      = aws_subnet.main.id
  route_table_id = aws_route_table.main.id
}
```

Security Group

```
resource "aws_security_group" "allow_ssh_http" {
  vpc_id = aws_vpc.main.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"]
  }

  tags = {
    Name = "allow_ssh_http"
  }
}
```

Key Pair

```
resource "aws_key_pair" "main" {
  key_name   = var.key_name
  public_key = file(var.public_key_path)
}
```

EC2 Instance

```
resource "aws_instance" "example" {
  ami                         = var.ami
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.main.id
  vpc_security_group_ids      = [aws_security_group.allow_ssh_http.id]
  key_name                    = aws_key_pair.main.key_name
  associate_public_ip_address = true

  tags = {
    Name = "MyExampleInstance"
  }
}
```

3. `variables.tf` File

variables.tf

```
variable "region" {
  description = "The AWS region to create resources in"
  default     = "us-west-2"
}

variable "availability_zone" {
  description = "The Availability Zone to create resources in"
  default     = "us-west-2a"
}

variable "vpc_cidr" {
  description = "The CIDR block for the VPC"
  default     = "10.0.0.0/16"
}

variable "subnet_cidr" {
  description = "The CIDR block for the subnet"
  default     = "10.0.1.0/24"
}
```

```
variable "instance_type" {
  description = "The EC2 instance type"
  default     = "t2.micro"
}

variable "ami" {
  description = "The AMI to use for the instance"
  default     = "ami-0c55b159cbfafe1f0"  # Update with your preferred AMI
}

variable "key_name" {
  description = "The name of the SSH key pair"
  default     = "my-key-pair"
}

variable "public_key_path" {
  description = "The path to the SSH public key file"
  default     = "~/.ssh/id_rsa.pub"  # Update with your
public key path

}
```

`outputs.tf` File

outputs.tf

```
output "vpc_id" {

  description = "The ID of the VPC"

  value       = aws_vpc.main.id

}


output "subnet_id" {

  description = "The ID of the subnet"

  value       = aws_subnet.main.id

}


output "instance_ip" {

  description = "The public IP of the EC2 instance"

  value       = aws_instance.example.public_ip

}
```

```
output "instance_id" {
  description = "The ID of the EC2 instance"
  value       = aws_instance.example.id
}
```

`terraform.tfvars` File (Optional)

terraform.tfvars

```
region           = "us-west-2"
availability_zone = "us-west-2a"
vpc_cidr         = "10.0.0.0/16"
subnet_cidr      = "10.0.1.0/24"
instance_type    = "t2.micro"
ami              = "ami-0c55b159cbfafe1f0"
key_name         = "my-key-pair"
public_key_path   = "~/.ssh/id_rsa.pub"
```

Initialize, Plan, and Apply

1. Initialize the project : Download necessary providers and initialize the working directory.

```
terraform init
```

2. Plan the infrastructure

```
terraform plan
```

3. Apply the configuration

```
terraform apply
```

7. Cleanup

When you're done with the infrastructure, destroy it:

```
terraform destroy
```

Terraform project now provisions a full network setup, including a VPC, Subnet, Internet Gateway, Security Group, and an EC2 instance with SSH access.

This project provides a more complete and realistic cloud infrastructure setup, which you can further customize or extend with additional services such as S3, RDS, Load Balancers, etc.