Packer & Terraform

This is a sample project where we use packer and Terraform to create and provision custom AMI, VPC, subnets and EC2 instances.

A. Create a custom AWS AMI using Packer that contains the following:

- Amazon Linux
- Docker
- Your SSH public key is set so you can login using your private key

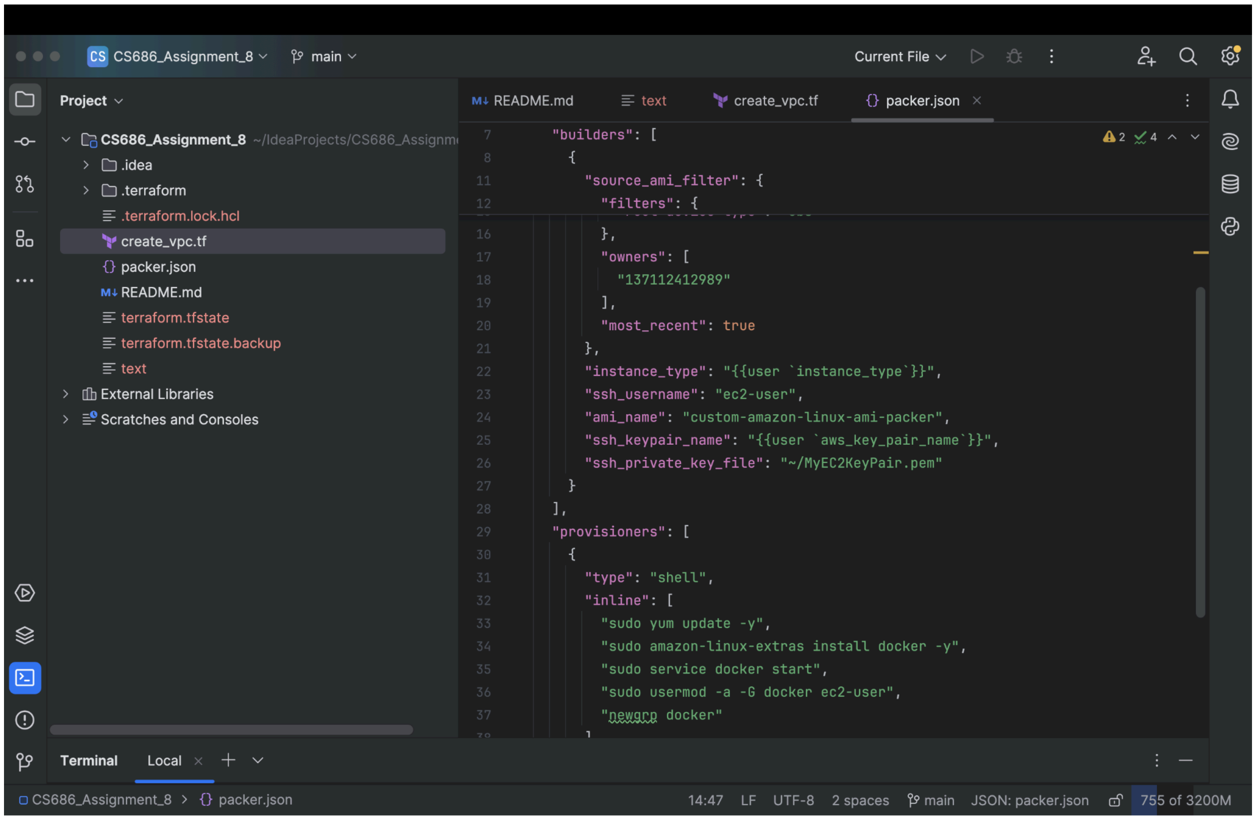
To create a custom AWS AMI with the above specifications using packer,

We need to build the packer.json file.

Make the below changes in the file to run locally.

- Update region, instance_type and aws_key_pair_name inside variables.

  Aws_key_pair_name will be the name of the KeyPair we create on AWS console and whose private key we use to access instances.
- Update the path to private SSH key (.pem file) inside the builders.

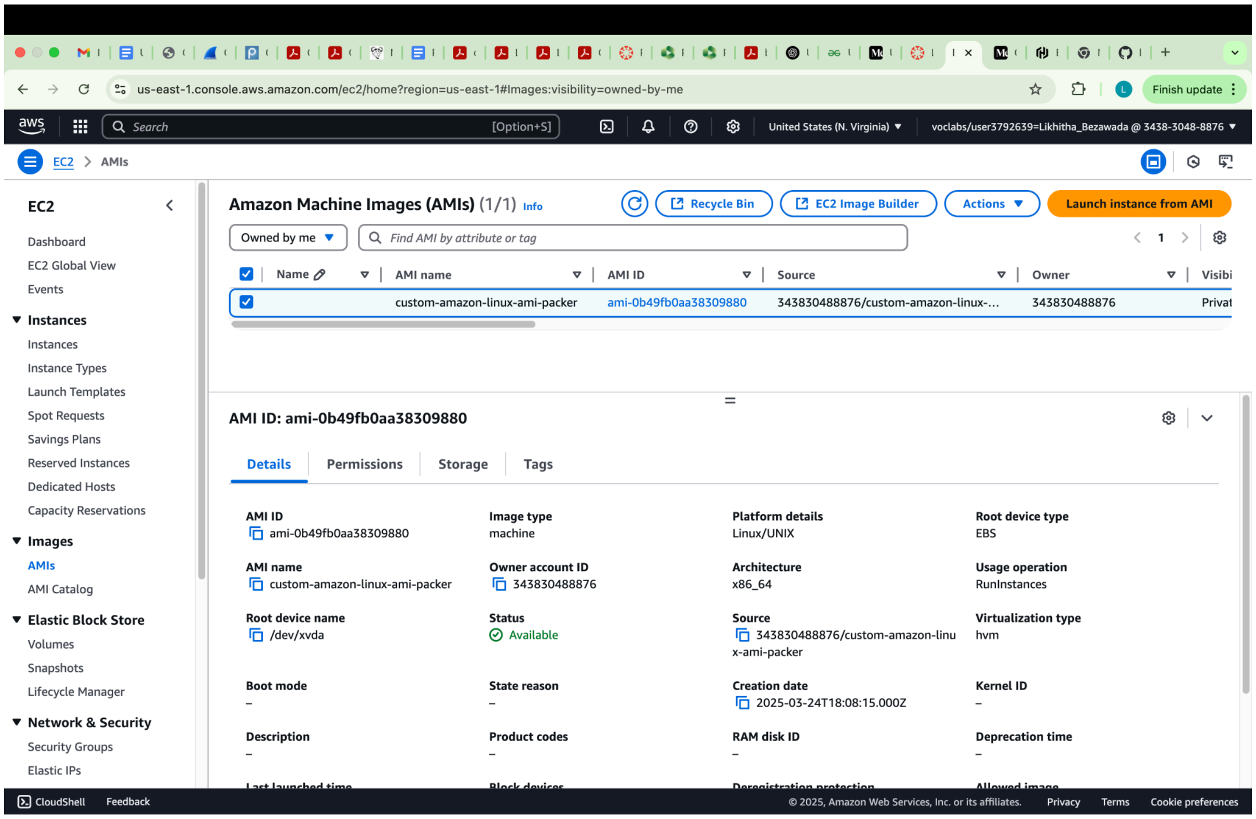


```
  "builders": [
    {
      "source_ami_filter": {
        "filters": {
        },
        "owners": [
          "137112412989"
        ],
        "most_recent": true
      },
      "instance_type": "{{user `instance_type`}}",
      "ssh_username": "ec2-user",
      "ami_name": "custom-amazon-linux-ami-packer",
      "ssh_keypair_name": "{{user `aws_key_pair_name`}}",
      "ssh_private_key_file": "~/MyEC2KeyPair.pem"
    }
  ],
  "provisioners": [
    {
      "type": "shell",
      "inline": [
        "sudo yum update -y",
        "sudo amazon-linux-extras install docker -y",
        "sudo service docker start",
        "sudo usermod -a -G docker ec2-user",
        "newgrp docker"
```

Then save the file and use the below command for build.

packer build packer.json

This will create a custom AMI with the above requirements.


us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Images:visibility=owned-by-me

United States (N. Virginia) ▼ voclabs/user3792639=Likhitha_Bezawada @ 3438-3048-8876 ▼

EC2 > AMIs

Amazon Machine Images (AMIs) (1/1) Info

Name	AMI name	AMI ID	Source	Owner	Visibi
	custom-amazon-linux-ami-packer	ami-0b49fb0aa38309880	343830488876/custom-amazon-linux-...	343830488876	Privat

AMI ID: ami-0b49fb0aa38309880

Details | Permissions | Storage | Tags

AMI ID ami-0b49fb0aa38309880	**Image type** machine	**Platform details** Linux/UNIX	**Root device type** EBS
AMI name custom-amazon-linux-ami-packer	**Owner account ID** 343830488876	**Architecture** x86_64	**Usage operation** RunInstances
Root device name /dev/xvda	**Status** Available	**Source** 343830488876/custom-amazon-linux-ami-packer	**Virtualization type** hvm
Boot mode –	**State reason** –	**Creation date** 2025-03-24T18:08:15.000Z	**Kernel ID** –
Description –	**Product codes** –	**RAM disk ID** –	**Deprecation time** –

© 2025, Amazon Web Services, Inc. or its affiliates.

B. Terraform scripts to provision AWS resources:

VPC, private subnets, public subnets, all necessary routes (use modules)

1 bastion host in the public subnet (accept only your IP on port 22)

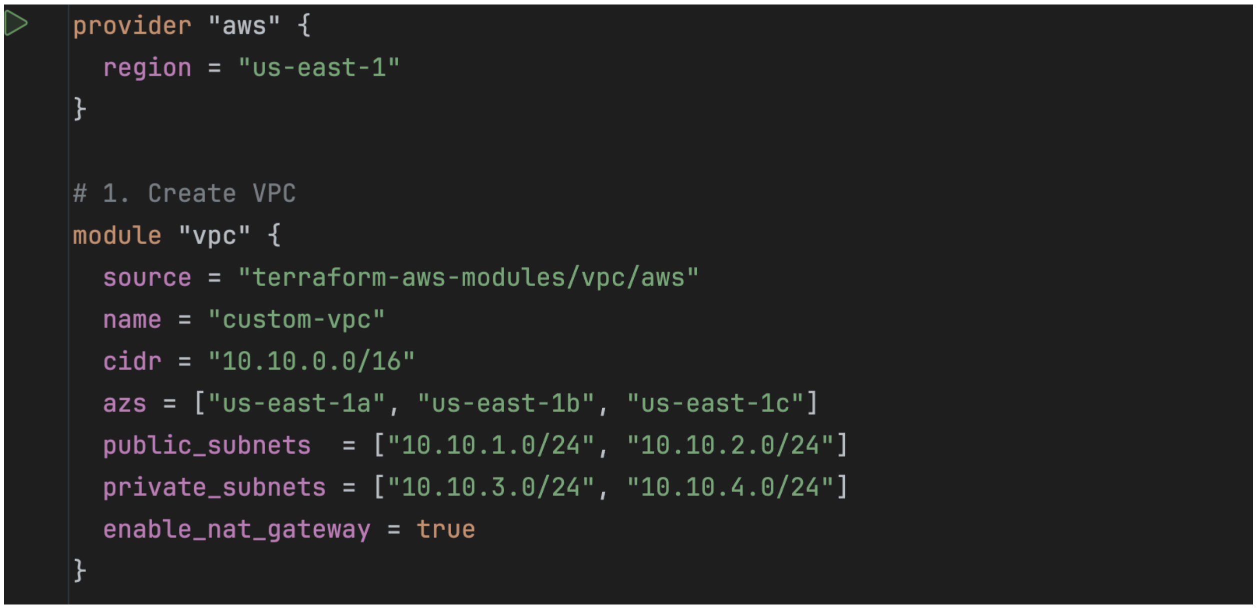
6 EC2 instances in the private subnet using your new AMI created from Packer

To implement part B, install terraform in your system locally from the below page

https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

The file create_vpc.tf contains code to provision all the above instances.

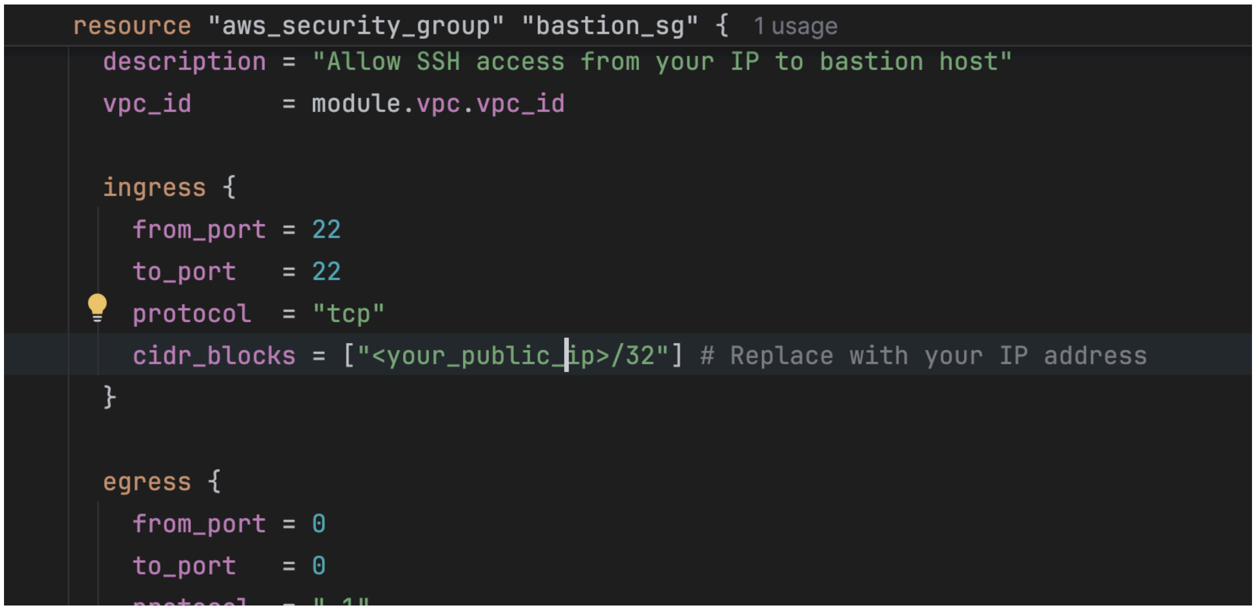
- Replace the region in the first two blocks with the appropriate region name.



```
provider "aws" {
  region = "us-east-1"
}

# 1. Create VPC
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"
  name = "custom-vpc"
  cidr = "10.10.0.0/16"
  azs = ["us-east-1a", "us-east-1b", "us-east-1c"]
  public_subnets  = ["10.10.1.0/24", "10.10.2.0/24"]
  private_subnets = ["10.10.3.0/24", "10.10.4.0/24"]
  enable_nat_gateway = true
}
```

- Inside the block bastion_sg, replace the ip address with public ip of device you want to access bastion host from.

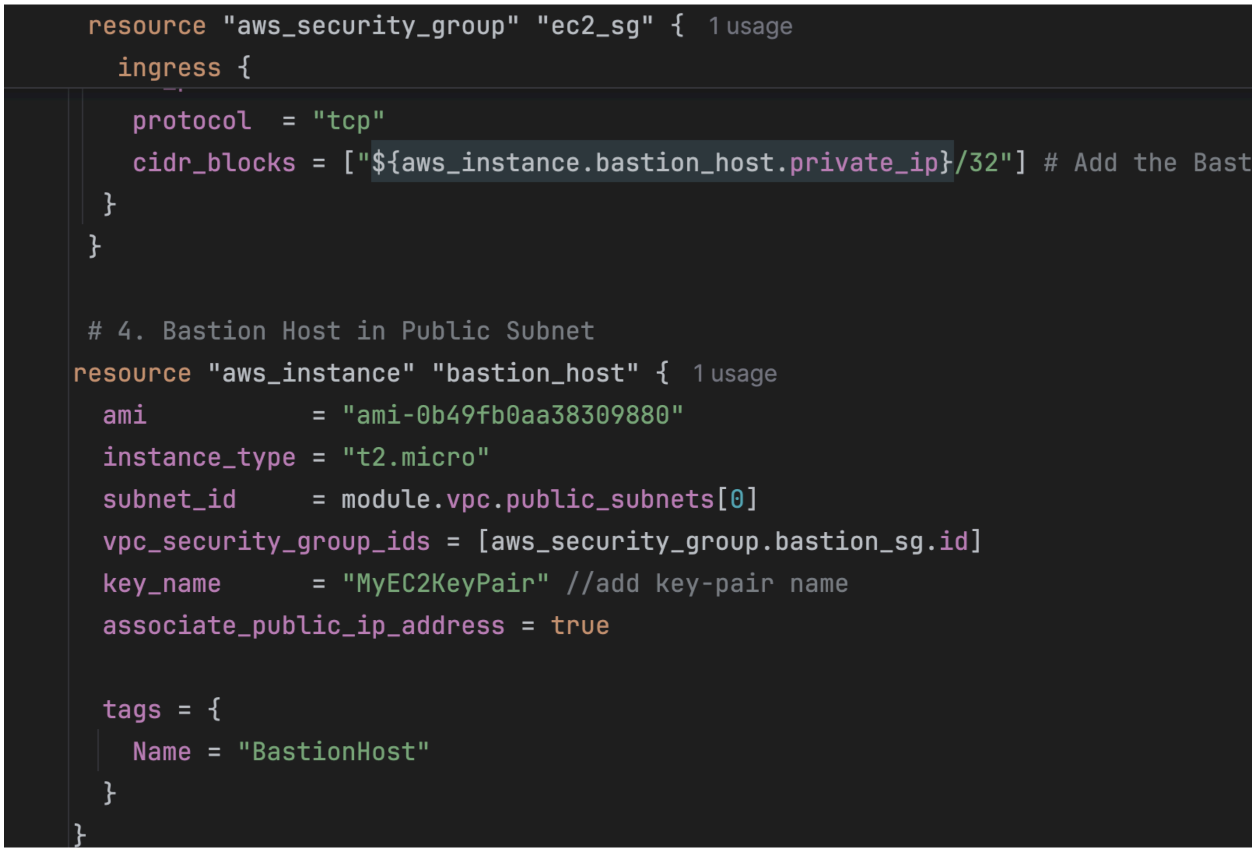


```
resource "aws_security_group" "bastion_sg" {  1 usage
  description = "Allow SSH access from your IP to bastion host"
  vpc_id      = module.vpc.vpc_id

  ingress {
    from_port = 22
    to_port   = 22
    protocol  = "tcp"
    cidr_blocks = ["<your_public_ip>/32"] # Replace with your IP address
  }

  egress {
    from_port = 0
    to_port   = 0
```

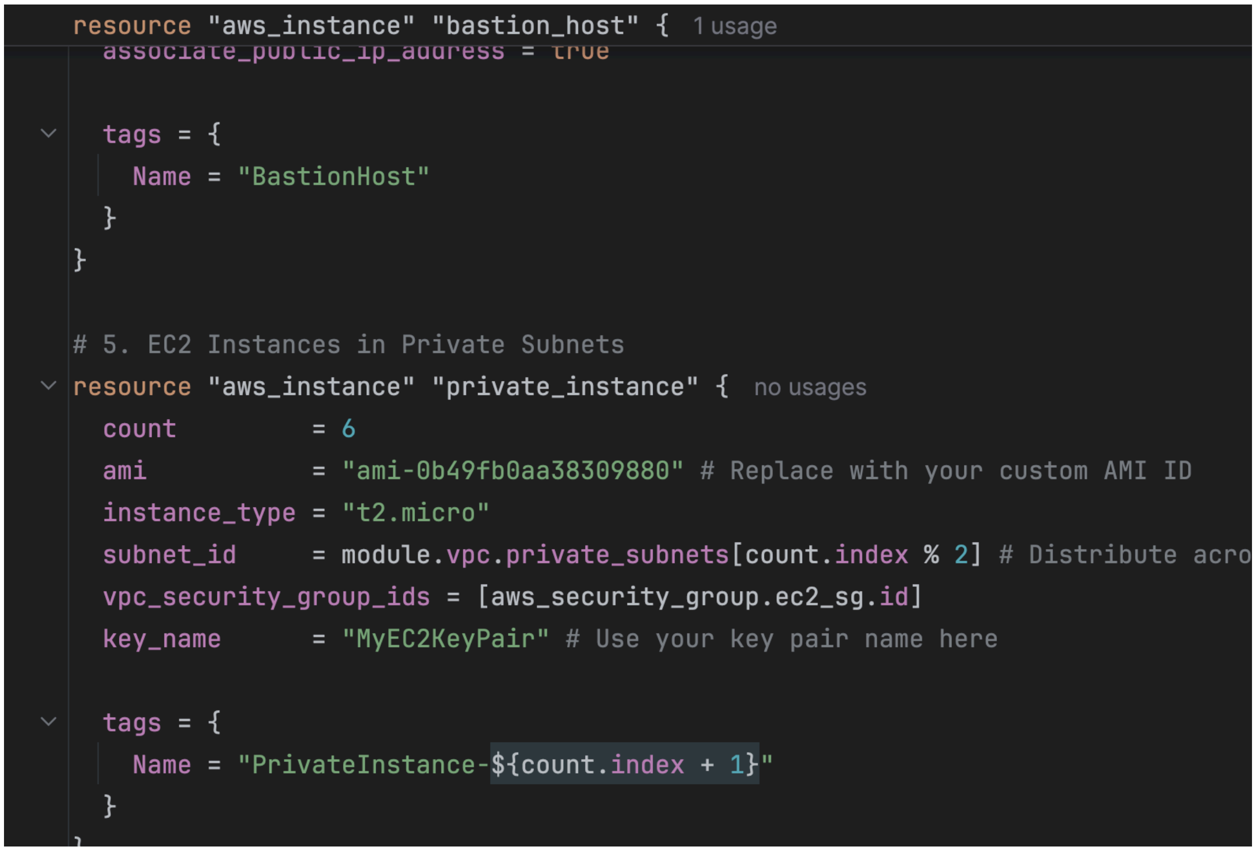
- Replace the ami id with ami id you created with packer.json in part A inside the blocks bastion_host and private_instance.



```
resource "aws_security_group" "ec2_sg" {  1 usage
  ingress {
    protocol  = "tcp"
    cidr_blocks = ["${aws_instance.bastion_host.private_ip}/32"] # Add the Bast
  }
}

# 4. Bastion Host in Public Subnet
resource "aws_instance" "bastion_host" {  1 usage
  ami           = "ami-0b49fb0aa38309880"
  instance_type = "t2.micro"
  subnet_id     = module.vpc.public_subnets[0]
  vpc_security_group_ids = [aws_security_group.bastion_sg.id]
  key_name      = "MyEC2KeyPair" //add key-pair name
  associate_public_ip_address = true

  tags = {
    Name = "BastionHost"
  }
}
```



```
resource "aws_instance" "bastion_host" {  1 usage
  associate_public_ip_address = true

  tags = {
    Name = "BastionHost"
  }
}

# 5. EC2 Instances in Private Subnets
resource "aws_instance" "private_instance" {  no usages
  count         = 6
  ami           = "ami-0b49fb0aa38309880" # Replace with your custom AMI ID
  instance_type = "t2.micro"
  subnet_id     = module.vpc.private_subnets[count.index % 2] # Distribute acro
  vpc_security_group_ids = [aws_security_group.ec2_sg.id]
  key_name      = "MyEC2KeyPair" # Use your key pair name here

  tags = {
    Name = "PrivateInstance-${count.index + 1}"
  }
}
```

Now run the below commands to provision instances.

terraform init

terraform plan

terraform apply

To access private ec2 instances from bastion host , first copy your private key to bastion host and then use it to ssh it to private instances.