```
ubuntu@ip-172-31-27-10:~/proj$ ls
ec2.tf  main.tf  provider.tf  terraform.tfstate
terraform.tfvars  vars.tf

ubuntu@ip-172-31-27-10:~/proj$ cat provider.tf
provider "aws" {
  region="us-east-1"
  access_key="<confidential>"
  secret_key="<confidential>"
}

ubuntu@ip-172-31-27-10:~/proj$ cat vars.tf
variable "region" {}
variable "main_vpc_cidr" {}
variable "public_subnets" {}
variable "private_subnets" {}

ubuntu@ip-172-31-27-10:~/proj$ cat terraform.tfvars
region="us-east-1"
main_vpc_cidr="10.0.0.0/24"
public_subnets="10.0.0.128/26"
private_subnets="10.0.0.192/26"

ubuntu@ip-172-31-27-10:~/proj$ cat main.tf
# create a vpc
resource "aws_vpc" "Main" {
  cidr_block=var.main_vpc_cidr
  instance_tenancy="default"
  tags={
    Name="test-vpc"
  }
}

# create an elastic IP
resource "aws_eip" "nateIP" {
  vpc=true
}

# create an IGW and attach to the vpc
resource "aws_internet_gateway" "IGW" {
  vpc_id=aws_vpc.Main.id
  tags={
    Name="test-IGW"
  }
}
```

```
# create a NATGW
resource "aws_nat_gateway" "NATgw" {
  allocation_id=aws_eip.nateIP.id
  subnet_id=aws_subnet.publicsubnets.id
  tags={
    Name="test-NATGW"
  }
}

# create public subnet
resource "aws_subnet" "publicsubnets" {
  vpc_id=aws_vpc.Main.id
  cidr_block="${var.public_subnets}"
  tags={
    Name="public subnet"
  }
}

# create private subnet
resource "aws_subnet" "privatesubnet" {
  vpc_id=aws_vpc.Main.id
  cidr_block="${var.private_subnets}"
  tags={
    Name="private subnet"
  }
}

# create public route table
resource "aws_route_table" "publicRT" {
  vpc_id=aws_vpc.Main.id
  route {
    cidr_block="0.0.0.0/0"
    gateway_id=aws_internet_gateway.IGW.id
  }
  tags={
    Name="publicRT"
  }
}

# create private route table
resource "aws_route_table" "privateRT" {
  vpc_id=aws_vpc.Main.id
  route {
    cidr_block="0.0.0.0/0"
```

```
    nat_gateway_id=aws_nat_gateway.NATgw.id
  }
  tags={
    Name="privateRT"
  }
}

# associate public subnet with public route table
resource "aws_route_table_association"
"publicRTassociation" {
  subnet_id=aws_subnet.publicsubnets.id
  route_table_id=aws_route_table.publicRT.id
}

# associate private subnet with private route table
resource "aws_route_table_association"
"privateRTassociation" {
  subnet_id=aws_subnet.privatesubnet.id
  route_table_id=aws_route_table.privateRT.id
}

# create or choose AWS AMI
data "aws_ami" "ubuntu-linux-1804" {
  most_recent = true
  owners      = ["679593333241"]
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-
amd64-server-*"]
  }
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}

# use existing key from aws console
variable "key_name" {
  description="tf"
  default="tf"
}
```

ubuntu@ip-172-31-27-10:~/proj$ cat ec2.tf

```
# launch public ec2
resource "aws_instance" "public-ec2" {
  ami= data.aws_ami.ubuntu-linux-1804.id
  instance_type="t2.micro"
  associate_public_ip_address=true
  subnet_id=aws_subnet.publicsubnets.id
  key_name=var.key_name
  tags={
    Name="public ec2"
  }
}

# launch private ec2
resource "aws_instance" "private-ec2" {
  ami= data.aws_ami.ubuntu-linux-1804.id
  instance_type="t2.micro"
  associate_public_ip_address=true
  subnet_id=aws_subnet.privatesubnet.id
  key_name=var.key_name
  tags={
    Name="private ec2"
  }
}

$ terraform init
$ terraform plan
$ terraform apply
$ terraform destroy
```

==========

Note:

choose any AMI/ EC2 OS:

Note: owners = ["679593333241"] had to subscribe the The AMI from https://aws.amazon.com/marketplace/pp/prodview-pkjqrkcfgcaog ,

After successful subscrbtion, copy-paste the AMI-ID of the subscription in

then at EC2 -> AMI -> public Image – seach box , thus we get the owner id of the AMI.