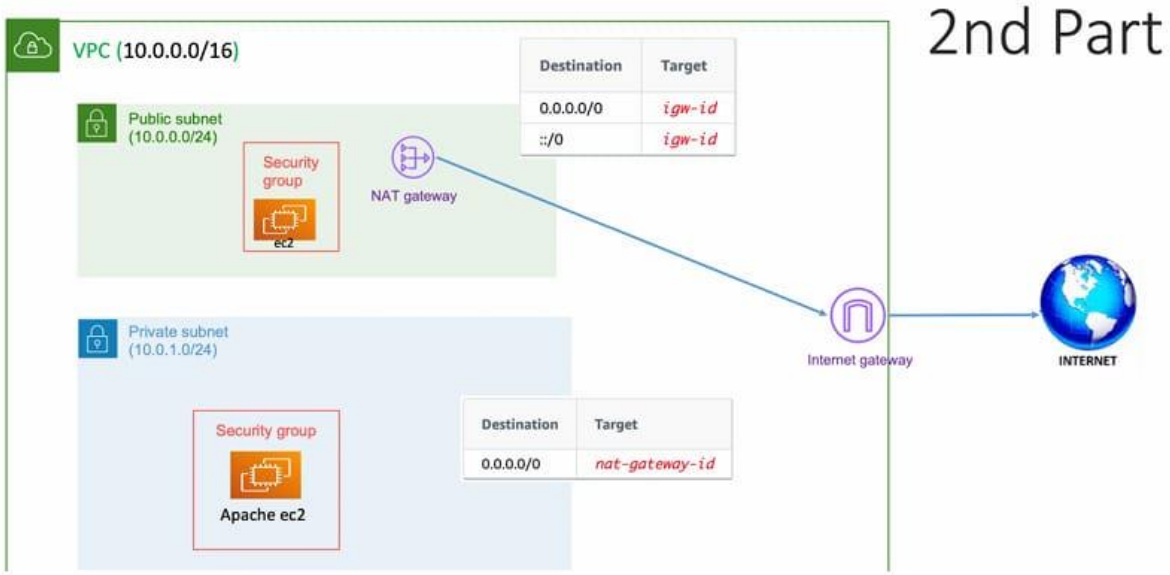
Bola Wisa

Task 2

Lab2:
Use variables for all arguments, output public_ip , private_ip , screenshot of the public ip from

the browser, private ip logs also use count for creating subnets.

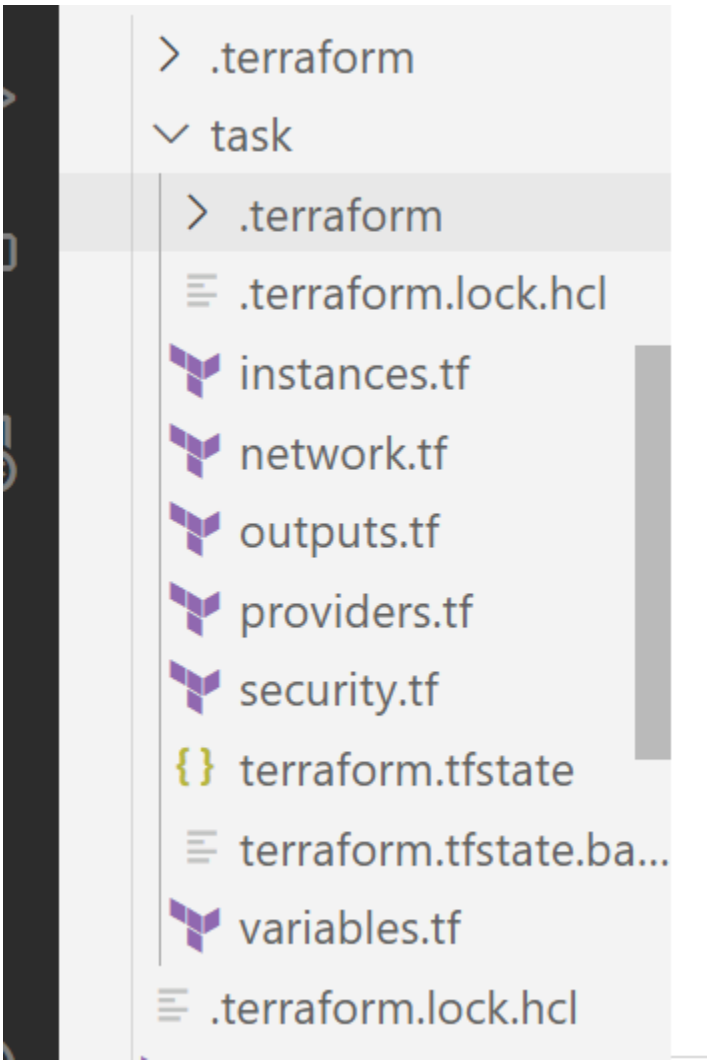


Search

Name	VPC ID	State	Block Public...	IPv4
VPC-Sohag	vpc-04a57d1693c502559	Available	Off	10.0

Find Instance by attribute or tag (case-sensitive) | All states | 1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
Apache-Server	i-04cd299c7fc675abb	Running	t2.micro	...	View alarms +	us-east-1d	-
WebServer	i-0dbad51c4740cb6d0	Running	t2.micro	...	View alarms +	us-east-1e	ec2-100-25-15
Jump-Box	i-04625ed17a47de590	Running	t2.micro	...	View alarms +	us-east-1e	-



instances.tf

terraform > task > instances.tf

```hcl
resource "aws_instance" "jump_box" {
  ami                    = var.ami_id
  instance_type          = var.instance_type
  subnet_id              = aws_subnet.public_subnet.id
  key_name               = var.key_name
  vpc_security_group_ids = [aws_security_group.web_sg.id]

  tags = {
    Name = "Jump-Box"
  }
}

resource "aws_instance" "apache_server" {
  ami                    = var.ami_id
  instance_type          = var.instance_type
  subnet_id              = aws_subnet.private_subnet.id
  key_name               = var.key_name
  vpc_security_group_ids = [aws_security_group.private_sg.id]

  user_data = <<-EOF
              #!/bin/bash
              sudo dnf update -y
              sudo dnf install -y httpd
              sudo systemctl start httpd
              sudo systemctl enable httpd
              echo "<h1>Welcome Bola, Today is Wednesday 26-3-2025</h1>" | sudo tee /var/www/html/index.html
              EOF

  tags = {
    Name = "Apache-Server"
  }
}


```

network.tf

terraform > task > network.tf

```hcl
resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr

  tags = {
    Name = "VPC-Sohag"
  }
}

resource "aws_subnet" "public_subnet" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = var.subnet_cidrs[0]
  map_public_ip_on_launch = true

  tags = {
    Name = "Public-Subnet"
  }
}

resource "aws_subnet" "private_subnet" {
  vpc_id     = aws_vpc.main.id
  cidr_block = var.subnet_cidrs[1]

  tags = {
    Name = "Private-Subnet"
  }
}

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "Internet-Gateway"
  }
}

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

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

  tags = {
    Name = "Public-RT"
```

terraform > task > network.tf

```
resource "aws_route_table" "public_rt" {
  tags = {
    Name = "Public-RT"
  }
}

resource "aws_route_table_association" "public_assoc" {
  subnet_id      = aws_subnet.public_subnet.id
  route_table_id = aws_route_table.public_rt.id
}

resource "aws_eip" "nat_eip" {
  domain = "vpc"
}

resource "aws_nat_gateway" "nat" {
  allocation_id = aws_eip.nat_eip.id
  subnet_id     = aws_subnet.public_subnet.id

  tags = {
    Name = "NAT-Gateway"
  }
}

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

  route {
    cidr_block     = "0.0.0.0/0"
    nat_gateway_id = aws_nat_gateway.nat.id
  }

  tags = {
    Name = "Private-RT"
  }
}

resource "aws_route_table_association" "private_assoc" {
  subnet_id      = aws_subnet.private_subnet.id
  route_table_id = aws_route_table.private_rt.id
}

```

outputs.tf

terraform > task > outputs.tf

```
output "jump_box_public_ip" {
  description = "Public IP of the Jump Box"
  value       = aws_instance.jump_box.public_ip
}

output "apache_server_private_ip" {
  description = "Private IP of the Apache Server"
  value       = aws_instance.apache_server.private_ip
}

```

providers.tf

terraform > task > providers.tf

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

```

security.tf

terraform > task > security.tf

```hcl
# Security Group for Public Instance (Web-SG)
resource "aws_security_group" "web_sg" {
  vpc_id = aws_vpc.main.id
  name   = "Web-SG"

  tags = {
    Name = "Web-SG"
  }
}

# Allow HTTP from anywhere
resource "aws_vpc_security_group_ingress_rule" "allow_http" {
  security_group_id = aws_security_group.web_sg.id
  cidr_ipv4         = "0.0.0.0/0"
  from_port         = 80
  to_port           = 80
  ip_protocol       = "tcp"
}

# Allow SSH from anywhere (For public instance)
resource "aws_vpc_security_group_ingress_rule" "allow_ssh_public" {
  security_group_id = aws_security_group.web_sg.id
  cidr_ipv4         = "0.0.0.0/0"
  from_port         = 22
  to_port           = 22
  ip_protocol       = "tcp"
}

# Allow all outbound traffic (for Web-SG)
resource "aws_vpc_security_group_egress_rule" "allow_all_out_web" {
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1" # All protocols
}

# Security Group for Private Instance (Private-SG)
resource "aws_security_group" "private_sg" {
  vpc_id = aws_vpc.main.id
  name   = "Private-SG"

  tags = {
    Name = "Private-SG"
  }
}

# Allow SSH only from the Public Subnet (Public Instance)
resource "aws_vpc_security_group_ingress_rule" "allow_ssh_private" {
  security_group_id = aws_security_group.private_sg.id
  referenced_security_group_id = aws_security_group.web_sg.id
  from_port         = 22
  to_port           = 22
  ip_protocol       = "tcp"
}

# Allow all outbound traffic (for Private-SG)
resource "aws_vpc_security_group_egress_rule" "allow_all_out_private" {
  security_group_id = aws_security_group.private_sg.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1" # All protocols
}
```

security.tf | variables.tf

terraform > task > variables.tf

```hcl
variable "aws_region" {
  description = "AWS region to deploy resources"
  type        = string
  default     = "us-east-1"
}

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

variable "subnet_cidrs" {
  description = "List of CIDR blocks for public and private subnets"
  type        = list(string)
  default     = ["10.0.0.0/24", "10.0.1.0/24"]
}

variable "key_name" {
  description = "The name of the SSH key pair"
  type        = string
  default     = "vockey"
}

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

variable "ami_id" {
  description = "AMI ID for the instances"
  type        = string
  default     = "ami-08b5b3a93ed654d19"
}

```