TASK_4
Problem_Statement

Step - 1: First of all, configure your AWS profile.

```
aws configure --profile Chaitanya
AWS Access Key ID [****************NOP3]:
AWS Secret Access Key [****************CKLD]:
Default region name [ap-south-1]:
Default output format [None]:
```

Step - 2: Next, we need to create a VPC.

The terrfaorm code to create a VPC is as follows :-

```
resource "aws_vpc" "lw_vpc" {
cidr_block = "192.168.0.0/16"
instance_tenancy = "default"
enable_dns_hostnames = true
tags = {
  Name = "lw_vpc"
}
}
```

Step - 3: Now, we need to create two subnets in this VPC :

a) public subnet- to allow outside access

b) private subnet- to restrict public access

```
resource "aws_subnet" "lw_public_subnet" {
vpc_id = "${aws_vpc.lw_vpc.id}"
cidr_block = "192.168.0.0/24"
availability_zone = "ap-south-1a"
map_public_ip_on_launch = "true"
tags = {
  Name = "lw_public_subnet"
}
}




resource "aws_subnet" "lw_private_subnet" {
  vpc_id = "${aws_vpc.lw_vpc.id}"
  cidr_block = "192.168.1.0/24"
  availability_zone = "ap-south-1a"
  tags = {
    Name = "lw_private_subnet"
  }
}
```

Step - 4: Next, we create a Public facing Internet Gateway.

The terraform code to create the Gateway is as follows :

```
resource "aws_internet_gateway" "lw_gw" {
vpc_id = "${aws_vpc.lw_vpc.id}"
tags = {
  Name = "lw_gw"
}
}
```

Step - 5: Next, we create a Routing Table & associate it with the Public Subnet.

```
resource "aws_route_table" "lw_rt" {
vpc_id = "${aws_vpc.lw_vpc.id}"

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

tags = {
  Name = "lw_rt"
}
}

resource "aws_route_table_association" "lw_rta" {
  subnet_id = "${aws_subnet.lw_public_subnet.id}"
  route_table_id = "${aws_route_table.lw_rt.id}"
}
```

Step - 6: Now, we create our security group which will be used while launching Wordpress.

```
resource "aws_security_group" "lw_sg" {

  name        = "lw_sg"
  vpc_id      = "${aws_vpc.lw_vpc.id}"

  ingress {

    description = "allow_http"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = [ "0.0.0.0/0"]

  }
```

```
    ingress {

      description = "allow_ssh"
      from_port   = 22
      to_port     = 22
      protocol    = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
    }

    ingress {

      description = "allow_icmp"
      from_port   = 0
      to_port     = 0
      protocol    = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
    }

    ingress {

      description = "allow_mysql"
      from_port   = 3306
      to_port     = 3306
      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 = "lw_sg"
     }
   }
```

Bastion Host for this security group-

```
resource "aws_security_group" "bastion_ssh_only" {
   depends_on=[aws_subnet.lw_public_subnet]
   name        = "bastion_ssh_only"
```

```
  description = "It allows bastion ssh inbound traffic"
  vpc_id      =  aws_vpc.lw_vpc.id




ingress {
    description = "allow bastion with ssh only"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
    ipv6_cidr_blocks =  ["::/0"]
  }



egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
    ipv6_cidr_blocks =  ["::/0"]
  }



  tags = {
    Name = "allow_bastion_ssh_only"
  }
}
```

Step : 7 Now, we create another security group which will be used to launch MySQL.

```
    resource "aws_security_group" "lw_sg_private" {


    name        = "lw_sg_private"
    vpc_id      = "${aws_vpc.lw_vpc.id}"


    ingress {


     description = "allow_mysql"
     from_port   = 3306
     to_port     = 3306
     protocol    = "tcp"
     security_groups = [aws_security_group.lw_sg.id]


    }
```

```
 ingress {


   description = "allow_icmp"
   from_port   = -1
   to_port     = -1
   protocol    = "icmp"
   security_groups = [aws_security_group.lw_sg.id]


 }


 egress {


  from_port   = 0
  to_port     = 0
  protocol    = "-1"
  cidr_blocks = ["0.0.0.0/0"]
  ipv6_cidr_blocks =  ["::/0"]
}



tags = {


    Name = "lw_sg_private"
  }
}
```

Bastion Host for this security group-

```
resource "aws_security_group" "bastion_host_sql" {
   depends_on=[aws_subnet.lw_public_subnet]
   name        = "bastion_with_ssh"
   vpc_id      =  aws_vpc.lw_vpc.id




ingress {
    description = "bastion host ssh only "
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    security_groups=[aws_security_group.bastion_ssh_only.id]


}



egress {
    from_port   = 0
```

```
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }


  tags = {
    Name = "bastion_with_ssh_only"
  }
 }
```

Step - 8: Next, we create a NAT gateway to connect our VPC/Network to the internet world.

```
resource "aws_eip" "lw-ip" {
  vpc             = true
  public_ipv4_pool = "amazon"
}
output "new_output" {
    value=  aws_eip.lw-ip
}



resource "aws_nat_gateway" "lw_nat_gw" {
  depends_on    = [aws_eip.lw-ip]
  allocation_id = aws_eip.lw-ip.id
  subnet_id     = aws_subnet.lw_public_subnet.id



  tags = {
    Name = "lw_nat_gw"
  }
}



resource "aws_route_table" "vp_private_subnet_for_rt" {
  depends_on = [aws_nat_gateway.lw_nat_gw]
  vpc_id = aws_vpc.sparsh_vpc.id
  route {
    cidr_block = "0.0.0.0/0"


    gateway_id = aws_nat_gateway.lw_nat_gw.id
  }
  tags = {
    Name = "vp_private_subnet_for_rt"
  }
}
```

```
resource "aws_route_table_association" "lw_private_subnet" {
  depends_on = [aws_route_table.lw_private_subnet]
  subnet_id      = aws_subnet.lw_private_subnet.id
  route_table_id = aws_route_table.lw_private_subnet.id
}
```

Step - 8: We launch our Wordpress and MySQL instances..

```
WORDPRESS
resource "aws_instance" "wordpress" {

  ami          = "ami-gg73h918"
  instance_type = "t2.micro"
  key_name      =  "lw_key"
  subnet_id     = "${aws_subnet.lw_public_subnet.id}"
  security_groups = ["${aws_security_group.lw_sg.id}"]
  associate_public_ip_address = true
  availability_zone = "ap-south-1a"


  tags = {
    Name = "lw_wordpress"
    }
  }
```

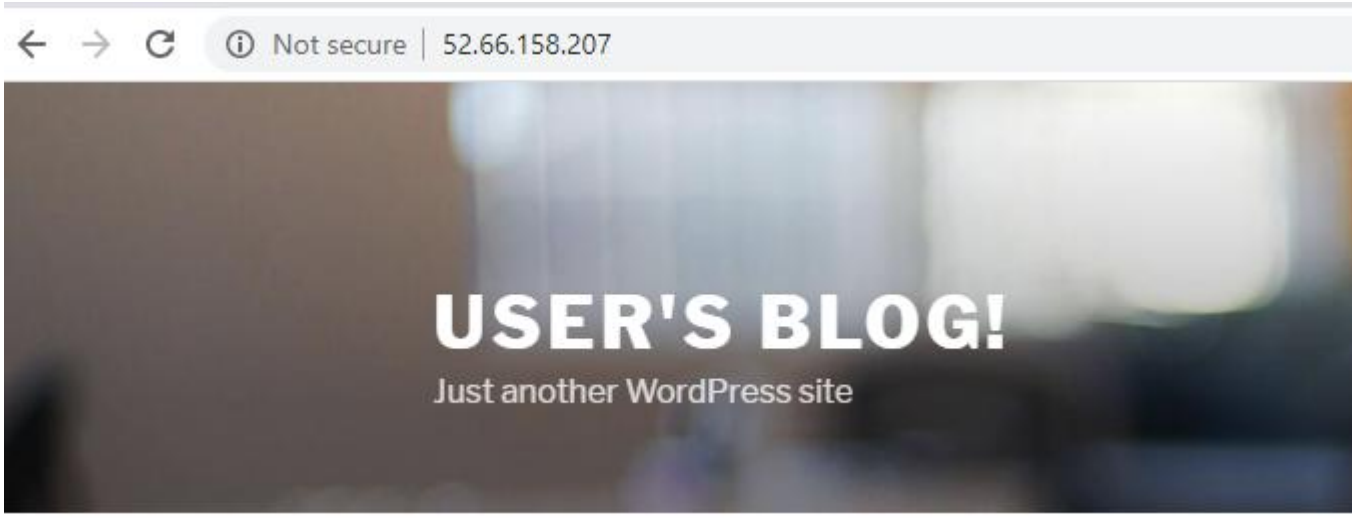
```
MYSQL
                resource "aws_instance" "sql" {
                ami            =  "ami-18696av5f68432d09"
                instance_type   =  "t2.micro"
                key_name       =  "lw_key"
                subnet_id     = "${aws_subnet.lw_private_subnet.id}"
                availability_zone = "ap-south-1a"
                security_groups = ["${aws_security_group.lw_sg_private.id}"]
                depends_on     = [aws_security_group.bastion_host_sql,aws_security_group.bastion_ssh]

                tags = {
                 Name = "lw_sql"
                 }
               }
```

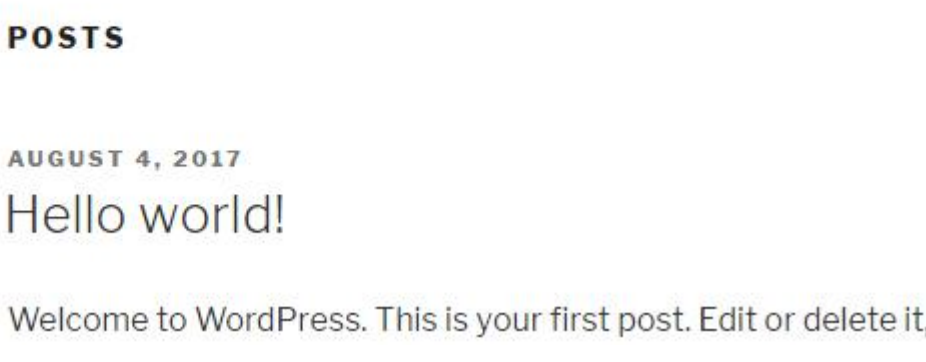
Bastion Host

```
        resource "aws_instance" "bastion_host" {
                    depends_on=[aws_security_group.bastion_ssh]
                    ami            =  "ami-08706cb5f68222d09"
                    instance_type   =  "t2.micro"
                    key_name       =  "task4"
                    subnet_id= aws_subnet.lw_public_subnet.id
```

```
  vpc_security_group_ids=[aws_security_group.bastion_ssh.id]

  tags = {

    Name = "bastion_host"

  }

}
```

Now, we run our terraform code. For doing so, we first run the command **terraform init**
we run the command **terraform apply --auto-approve**





TASK COMPLETED SUCCESSFULLYYYYYYY!!!!!!!!!!!!!!!!!!!!!!!!!!!!