TASK_3
Problem_Statement

Statement: We have to create a web portal for our company with all the security as much as possible.
So, we use Wordpress software with dedicated database server.
Database should not be accessible from the outside world for security purposes.
We only need to public the WordPress to clients.
So here are the steps for proper understanding!

Steps:
1) Write a Infrastructure as code using terraform, which automatically create a VPC.

2) In that VPC we have to create 2 subnets:
   a) public subnet [Accessible for Public World!]
   b) private subnet [Restricted for Public World!]

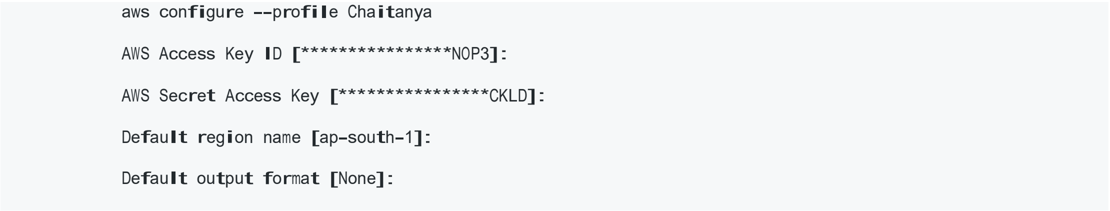
3) Create a public facing internet gateway for connect our VPC/Network to the internet world and attach this gateway to our VPC.

4) Create a routing table for Internet gateway so that instance can connect to outside world, update and associate it with public subnet.

5) Launch an ec2 instance which has Wordpress setup already having the security group allowing port 80 so that our client can connect to our wordpress site.
Also attach the key to instance for further login into it.

6) Launch an ec2 instance which has MYSQL setup already with security group allowing port 3306 in private subnet so that our wordpress vm can connect with the same.
Also attach the key with the same.

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. We will use terraform for this now.

```
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 :

Public Subnet- To provide access to outside world

Private Subnet-To restrict access to outside world

```
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 with the help of Terraform.

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

Step - 5: Next, we will 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 customized security group which will be used while launching Wordpress site with specific requirements .

```
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 = "connect_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"
     }
   }
```

Step : 7 Now, we create another security group with our customizations 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 = "connect_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"
      }
    }
```

Step - 8: We launch our Wordpress and MySQL instances

WORDPRESS

```
resource "aws_instance" "wordpress" {

      ami          = "ami-ff82f990"
      instance_type = "t2.micro"
      key_name      =  "lw_wp_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 = "hybrid_wordpress"
        }
      }
```

MYSQL

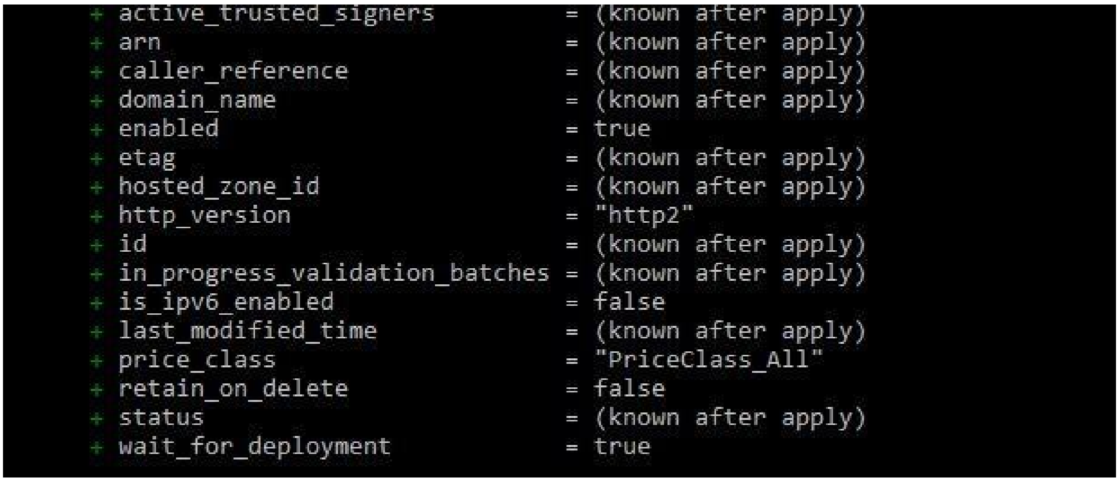
```
resource "aws_instance" "sql" {
                ami            =  "ami-90806cb5f98777d09"
                instance_type   =  "t2.micro"
                key_name        =  "lw_mysql_key"
                subnet_id     = "${aws_subnet.lw_private_subnet.id}"
                availability_zone = "ap-south-1a"
                security_groups = ["${aws_security_group.lw_sg_private.id}"]

                tags = {
                 Name = "hybrid_sql"
                 }
```

```
            }
```

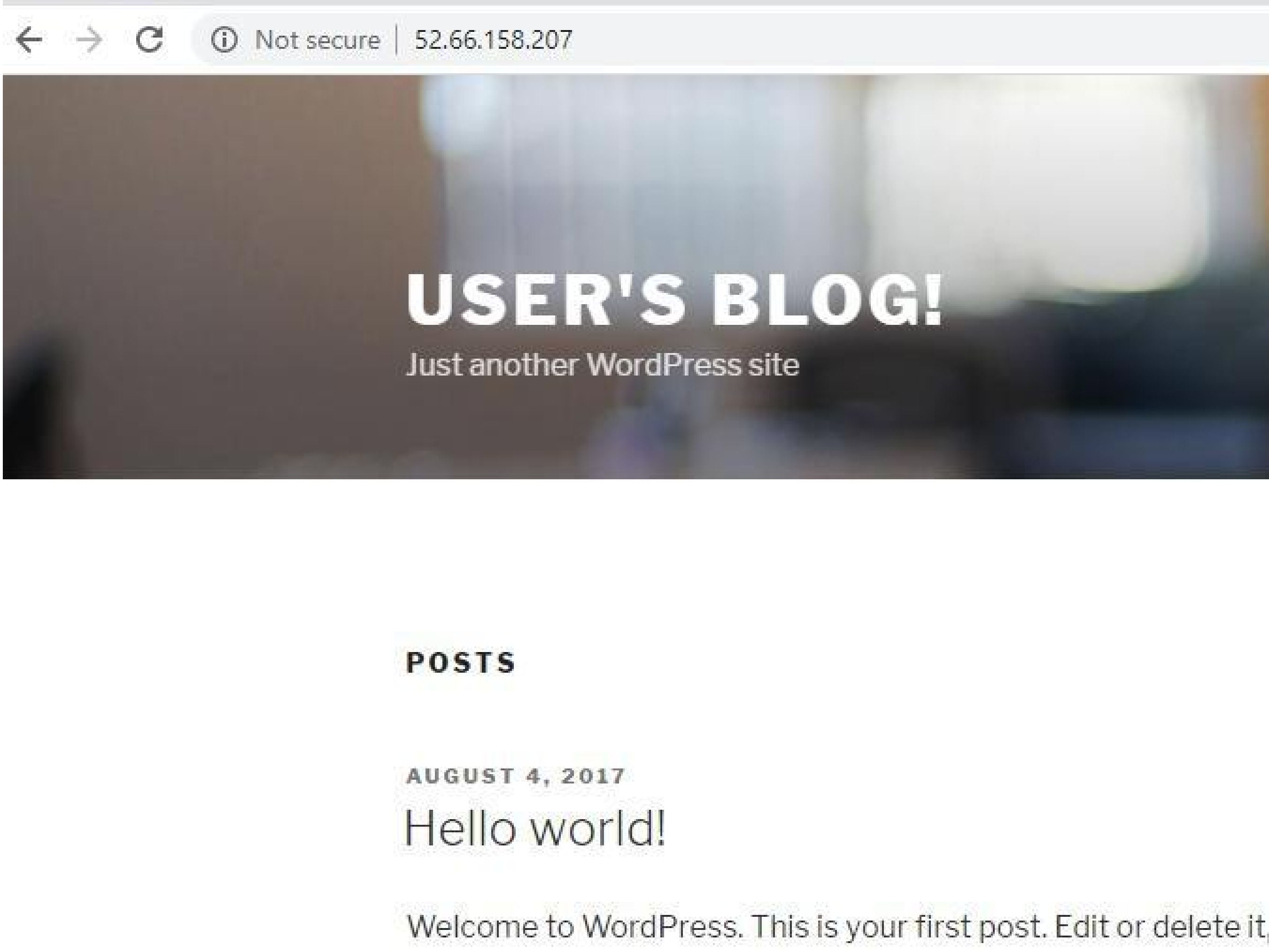
Step - 9: Now, we run our terraform code. For doing so, we first run the command **terraform init**

we run the command **terraform apply --auto-approve**



```
      + active_trusted_signers          = (known after apply)
      + arn                             = (known after apply)
      + caller_reference                = (known after apply)
      + domain_name                     = (known after apply)
      + enabled                         = true
      + etag                            = (known after apply)
      + hosted_zone_id                  = (known after apply)
      + http_version                    = "http2"
      + id                              = (known after apply)
      + in_progress_validation_batches = (known after apply)
      + is_ipv6_enabled                 = false
      + last_modified_time              = (known after apply)
      + price_class                     = "PriceClass_All"
      + retain_on_delete                = false
      + status                          = (known after apply)
      + wait_for_deployment             = true
```

After This code runs successfully our site will be up.



USER'S BLOG!

Just another WordPress site

POSTS

AUGUST 4, 2017

Hello world!

Welcome to WordPress. This is your first post. Edit or delete it, HENCE TASK SUCCESSFULLY COMPLETED !!