

Task-4 Launching instance in public and private subnet in addition with NAT

**Virtual Private Cloud (VPC)**

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you’ve defined. This virtual network closely resembles a traditional network that you’d operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

The following are the key concepts for VPCs:

* **Virtual private cloud (VPC)** — A virtual network dedicated to your AWS account.
* **Subnet** — A range of IP addresses in your VPC. *Subnets are like labs in vpc that have internal connectivity in every availability zones.*
* **Route table** — A set of rules, called routes, that are used to determine where network traffic is directed.
* **Internet gateway** — A gateway that you attach to your VPC to enable communication between resources in your VPC and the internet.
* **VPC endpoint** — Enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

*What is NAT ?*

You can use a network address translation (NAT) instance in a public subnet in your VPC to enable instances in the private subnet to initiate outbound IPv4 traffic to the internet or other AWS services, but prevent the instances from receiving inbound traffic initiated by someone on the internet.

**TASK:-**

Performing the following steps:

1. Write an Infrastructure as code using terraform, which automatically create a VPC.

2. In that VPC we have to create 2 subnets:

1. public subnet [ Accessible for Public World! ]

2. 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. Create a NAT gateway for connect our VPC/Network to the internet world and attach this gateway to our VPC in the public network

6. Update the routing table of the private subnet, so that to access the internet it uses the nat gateway created in the public subnet

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

8. 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.

*Note: Wordpress instance has to be part of public subnet so that our client can connect our site. mysql instance has to be part of private subnet so that outside world can’t connect to it. Don’t forgot to add auto ip assign and auto dns name assignment option to be enabled.*

**To access our AWS account :-**

provider “aws” {  
region = “ap-south-1”  
profile = “my\_user”

}

**######Create a customized VPC , for both the subnets (PUBLIC & PRIVATE)######**  
resource “aws\_vpc” “mytask4\_vpc” {  
cidr\_block = “192.168.0.0/16”  
instance\_tenancy = “default”  
enable\_dns\_hostnames = true  
tags = {  
Name = “mytask4\_vpc”  
}  
}

**#######Creating two subnets (PUBLIC & PRIVATE) for two different instances #######**  
resource “aws\_subnet” “public\_subnet” {  
vpc\_id = “${aws\_vpc.mytask4\_vpc.id}”  
cidr\_block = “192.168.10.0/24”  
availability\_zone = “ap-south-1a”  
map\_public\_ip\_on\_launch = “true”  
tags = {  
Name = “public\_subnet”  
}  
}

resource “aws\_subnet” “private\_subnet” {  
vpc\_id = “${aws\_vpc.mytask4\_vpc.id}”  
cidr\_block = “192.168.20.0/24”  
availability\_zone = “ap-south-1b”  
map\_public\_ip\_on\_launch = “true”  
tags = {  
Name = “private\_subnet”  
}  
}

**#######Creating a Internet Gateway to connect the VPC to the internet world #######**

resource “aws\_internet\_gateway” “mytask4\_gateway” {  
vpc\_id = “${aws\_vpc.mytask4\_vpc.id}”  
tags = {  
Name = “mytask4\_gateway”  
}  
}

**#######Creating a route table for the internet gateway , so that the instances can connect to the public world. Here , we only have to associate the route table to the public subnet so that the instance in the public subnet can connect to internet ########**

resource “aws\_route\_table” “mytask4\_route\_public” {  
vpc\_id = “${aws\_vpc.mytask4\_vpc.id}”  
route {  
cidr\_block = “0.0.0.0/0”  
gateway\_id = aws\_internet\_gateway.mytask4\_gateway.id  
}  
tags = {  
Name = “mytask4\_route\_public”  
}  
}

#Associating routing table to subnet we want to make public

resource “aws\_route\_table\_association” “a” {  
subnet\_id = aws\_subnet.public\_subnet.id  
route\_table\_id = aws\_route\_table.mytask4\_route\_public.id  
}

**######## Now, we are heading towards to create a Network Address Translation(NAT)  
which will connect the VPC to the public network and assign the subnet has public #######**

resource “aws\_eip” “task4\_eip” {  
vpc=true  
}

resource “aws\_nat\_gateway” “mytask4\_nat” {  
allocation\_id = aws\_eip.task4\_eip.id  
subnet\_id = aws\_subnet.public\_subnet.id

tags = {  
Name = “mytask4\_nat”  
}  
}

**#######Now we have to update the route table for the private subnet so that the pivate subnet  
uses NAT to access the internet which has been created in public subnet #######**

resource “aws\_route\_table” “mytask4\_route\_private” {  
vpc\_id = “${aws\_vpc.mytask4\_vpc.id}”  
route {  
cidr\_block = “0.0.0.0/0”  
gateway\_id = aws\_internet\_gateway.mytask4\_gateway.id  
}  
tags = {  
Name = “mytask4\_route\_private”  
}  
}

#Associating the routing table to private subnet

resource “aws\_route\_table\_association” “b” {  
subnet\_id = aws\_subnet.private\_subnet.id  
route\_table\_id = aws\_route\_table.mytask4\_route\_private.id  
}

**###### Now we will create security groups for the instances in the public and private subnet and for the BASTION HOST #######**  
resource “aws\_security\_group” “wp\_sg” {  
name = “task4\_sg”  
description = “Allow ssh-22,http-80 protocols and NFS inbound traffic”  
vpc\_id = “${aws\_vpc.mytask4\_vpc.id}”  
ingress {  
description = “HTTP”  
from\_port = 80  
to\_port = 80  
protocol = “tcp”  
cidr\_blocks = [“0.0.0.0/0”]  
}  
ingress {  
description = “SSH”  
from\_port = 22  
to\_port = 22  
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 = “wp\_sg”  
}  
}

resource “aws\_security\_group” “bastion” {  
name = “bastion”  
description = “Bastion host”  
vpc\_id = aws\_vpc.mytask4\_vpc.id

ingress {  
description = “ssh”  
from\_port = 22  
to\_port = 22  
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 =”bastion”  
}  
}

resource “aws\_security\_group” “sql\_sg” {  
name = “sg\_mysql”  
vpc\_id = “${aws\_vpc.mytask4\_vpc.id}”  
ingress {  
protocol = “tcp”  
from\_port = 3306  
to\_port = 3306  
security\_groups = [“${aws\_security\_group.wp\_sg.id}”]  
}  
egress {  
from\_port = 0  
to\_port = 0  
protocol = “-1”  
cidr\_blocks = [“0.0.0.0/0”]  
}  
tags ={  
Name= “sql\_sg”  
}  
}  
resource “aws\_security\_group” “bashion\_allow” {  
name = “bashion\_allow”  
description = “Allow bashion”  
vpc\_id = aws\_vpc.mytask4\_vpc.id

ingress {  
description = “SSH”  
from\_port = 22  
to\_port = 22  
protocol = “tcp”  
security\_groups = [aws\_security\_group.bastion.id]  
}  
  
  
egress {  
from\_port = 0  
to\_port = 0  
protocol = “-1”  
cidr\_blocks = [“0.0.0.0/0”]  
}

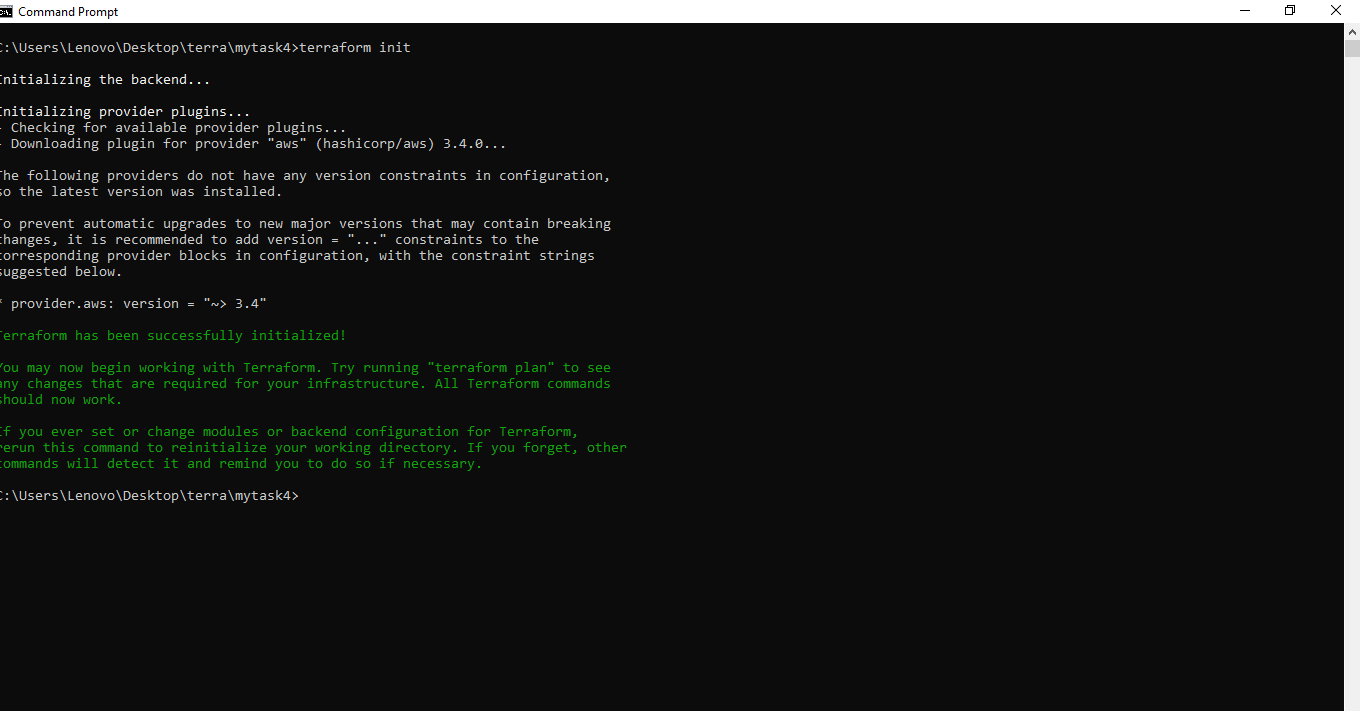
tags = {  
Name =”bashion\_allow”  
}  
}

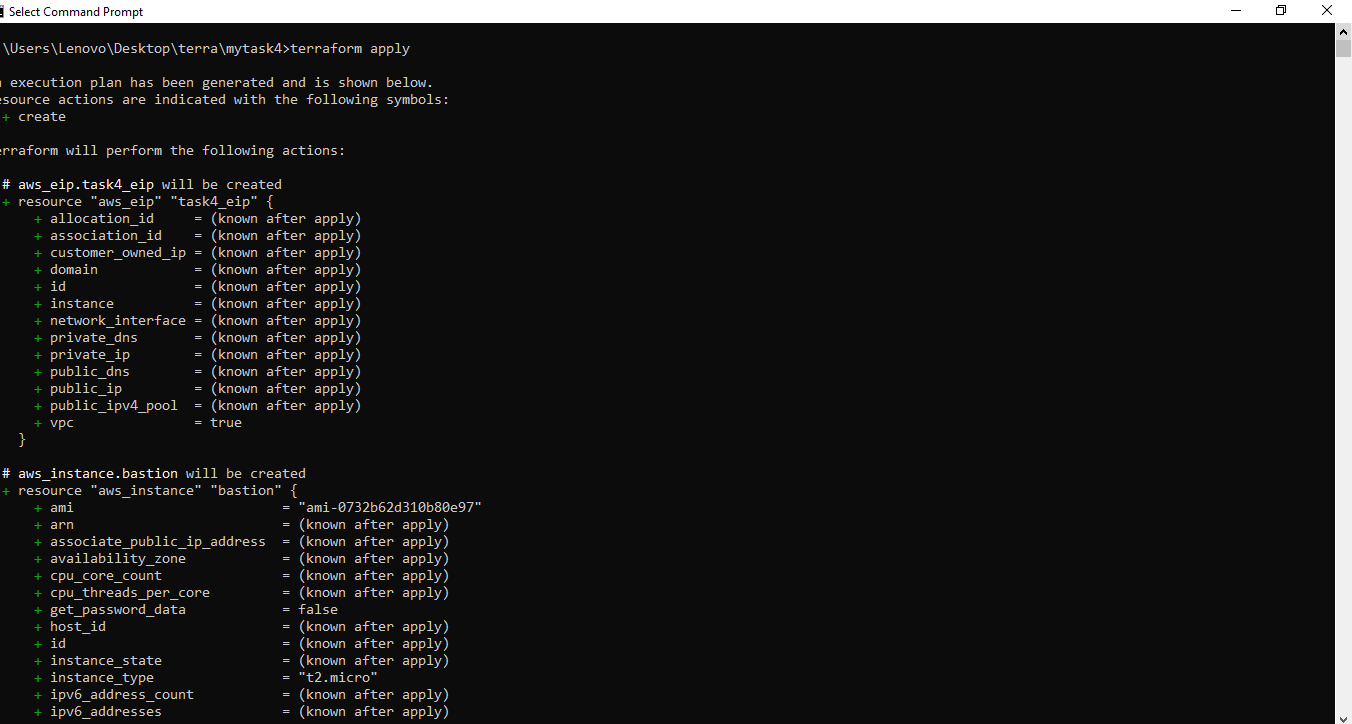
**######Launch instances in public and private subnets and Bastion Hosts #######**  
resource “aws\_instance” “mytask4\_wp” {  
ami = “ami-000cbce3e1b899ebd”  
instance\_type = “t2.micro”  
associate\_public\_ip\_address = true  
key\_name = “my-key”  
vpc\_security\_group\_ids = [aws\_security\_group.wp\_sg.id]  
subnet\_id = aws\_subnet.public\_subnet.id  
tags = {  
Name = “mytask4\_wp”  
}  
}

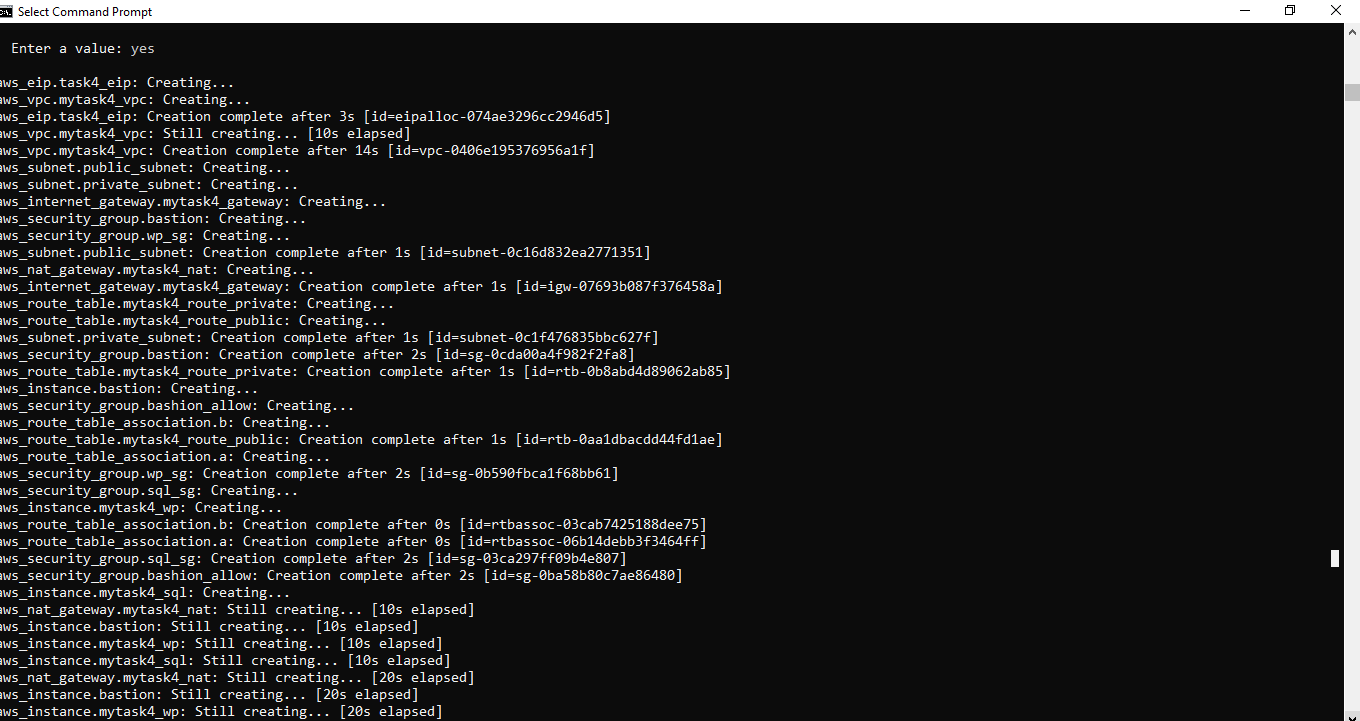
resource “aws\_instance” “bastion” {  
ami = “ami-0732b62d310b80e97”  
instance\_type = “t2.micro”  
key\_name = “my-key”  
vpc\_security\_group\_ids =[aws\_security\_group.bastion.id]  
subnet\_id = aws\_subnet.public\_subnet.id  
  
tags = {  
Name = “bastion”  
}  
}

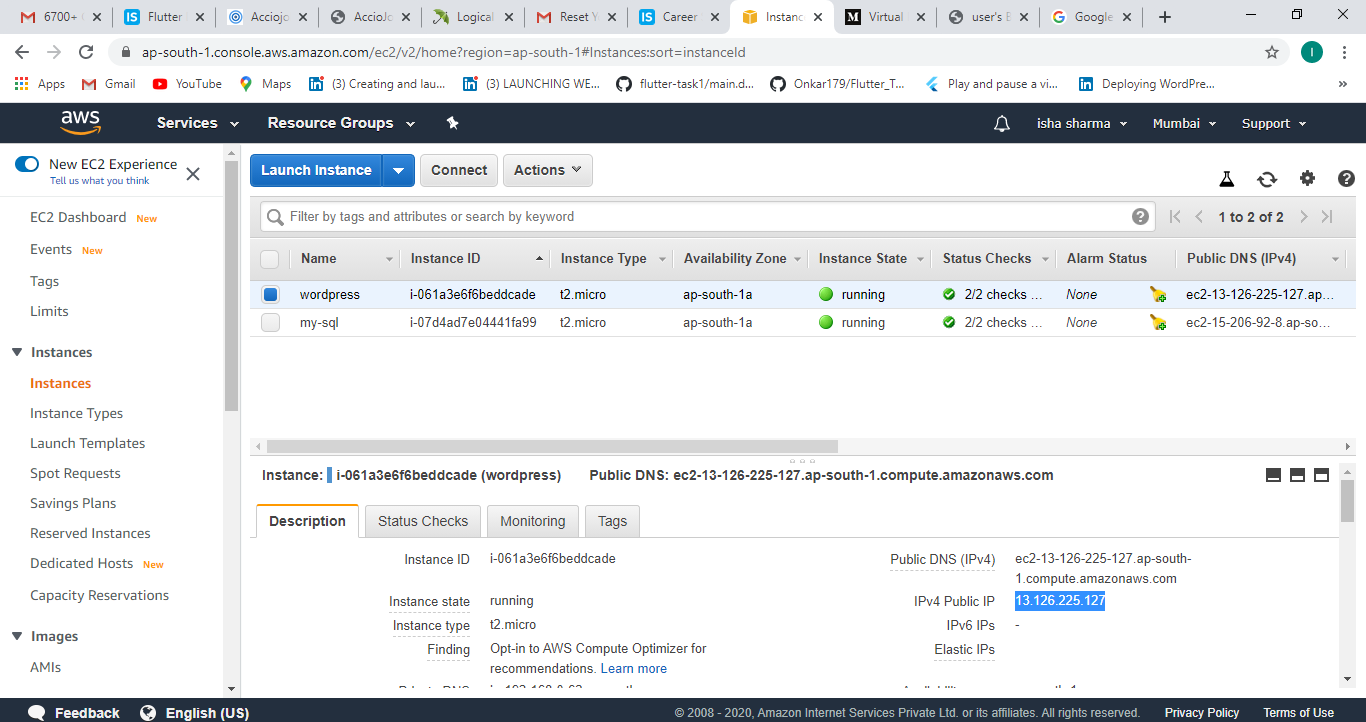
resource “aws\_instance” “mytask4\_sql” {  
ami = “ami-08706cb5f68222d09”  
instance\_type = “t2.micro”  
key\_name = “my-key”  
vpc\_security\_group\_ids = [aws\_security\_group.sql\_sg.id , aws\_security\_group.bashion\_allow.id]  
subnet\_id = aws\_subnet.private\_subnet.id  
tags = {  
Name = “mytask4\_sql”  
}  
}

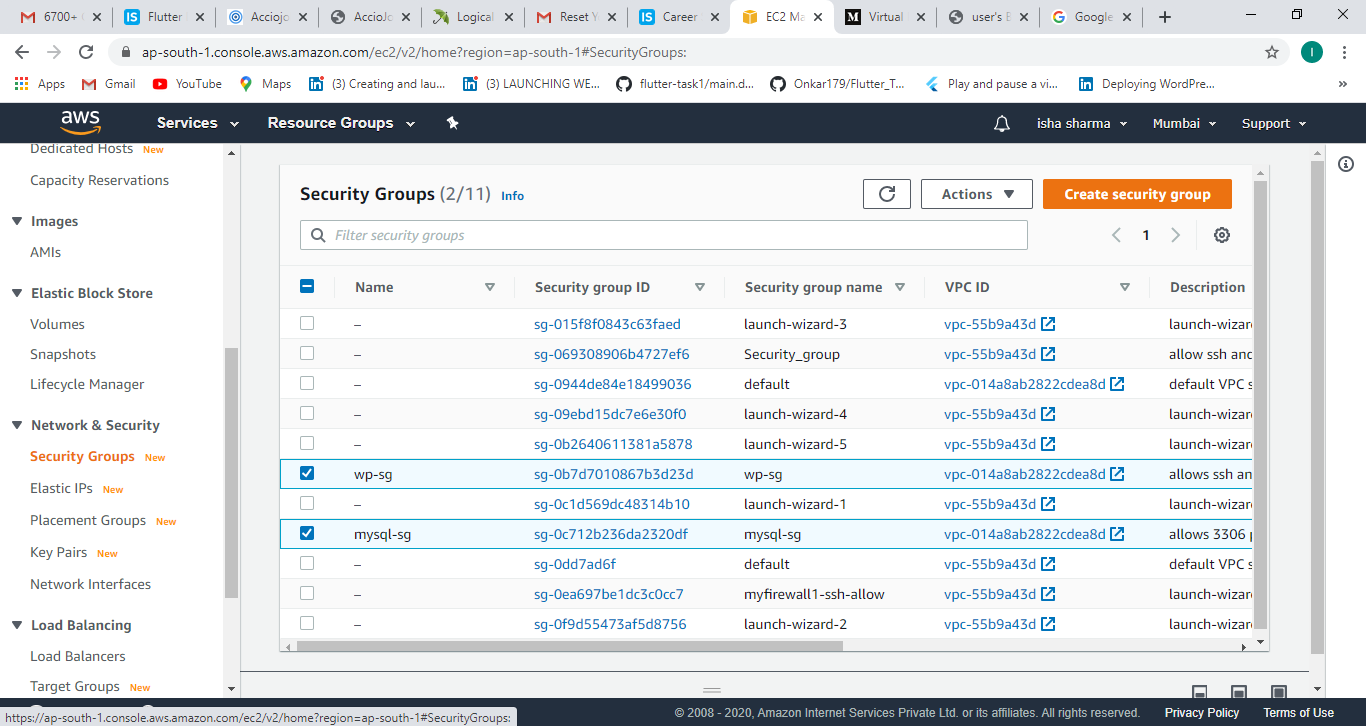
# terraform init

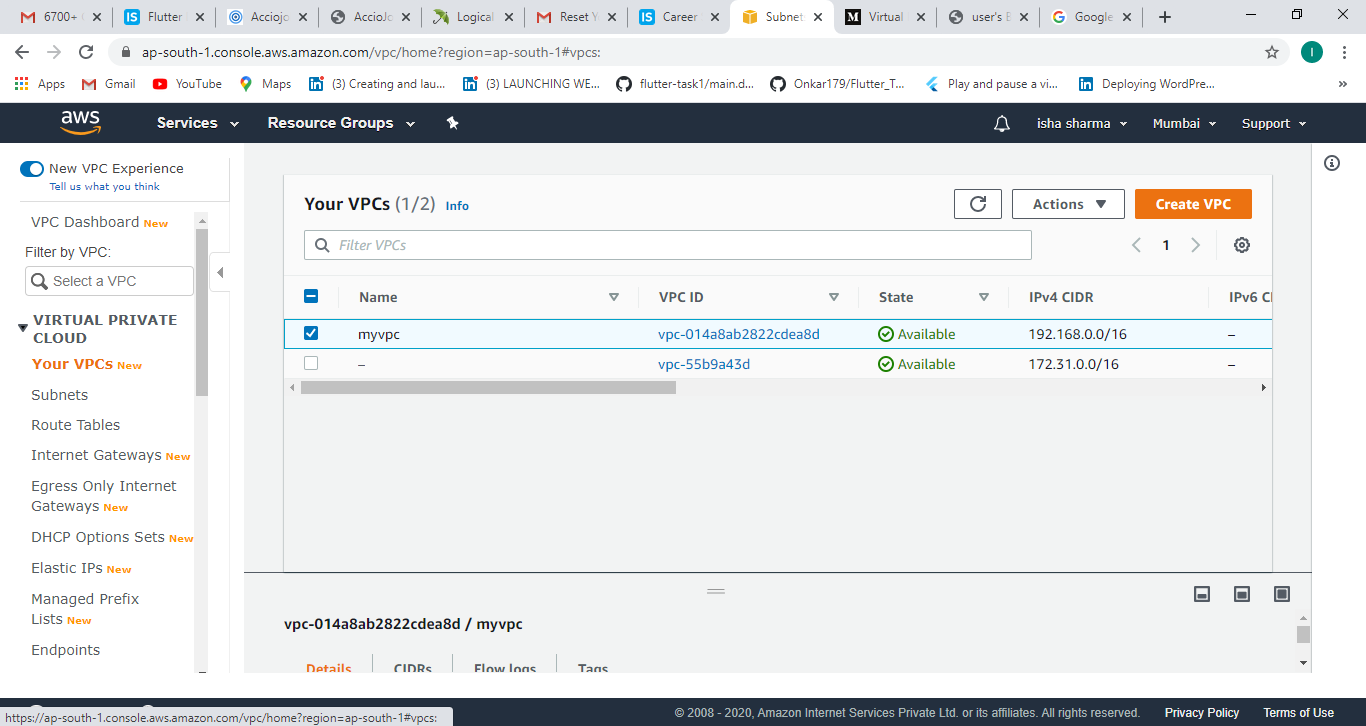


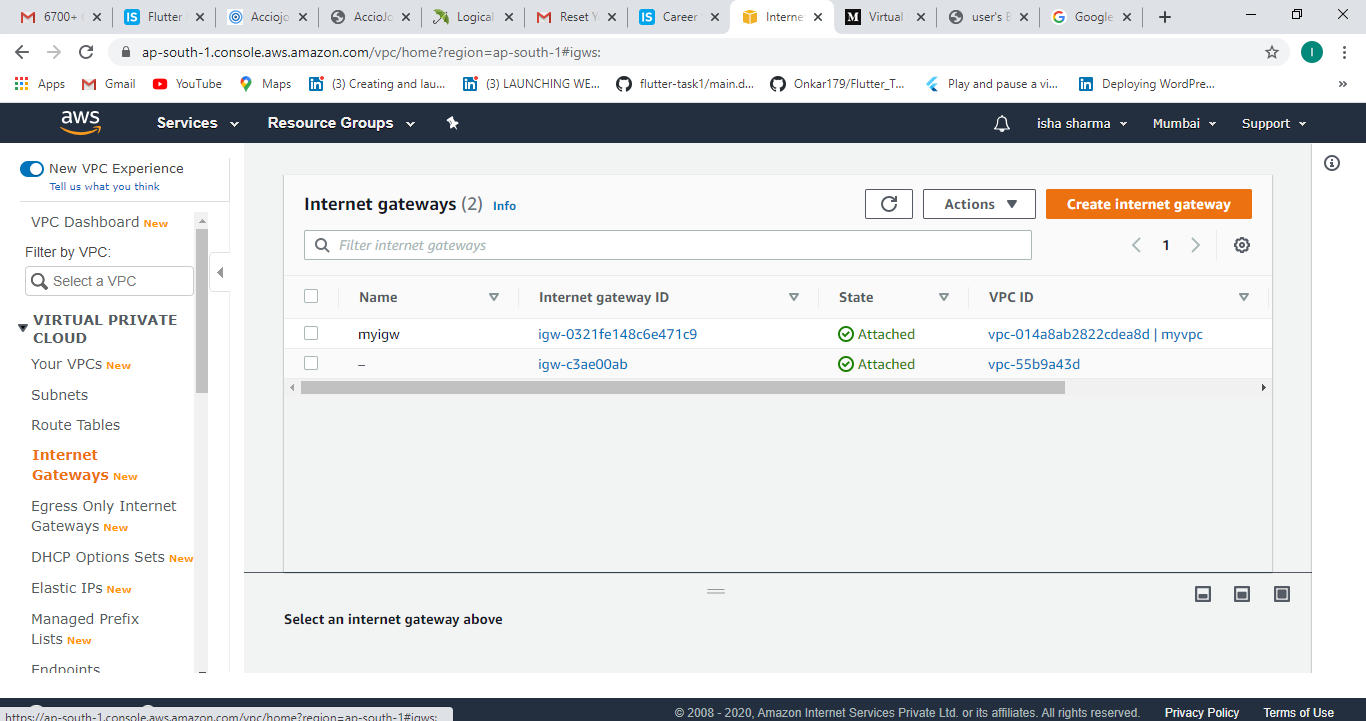


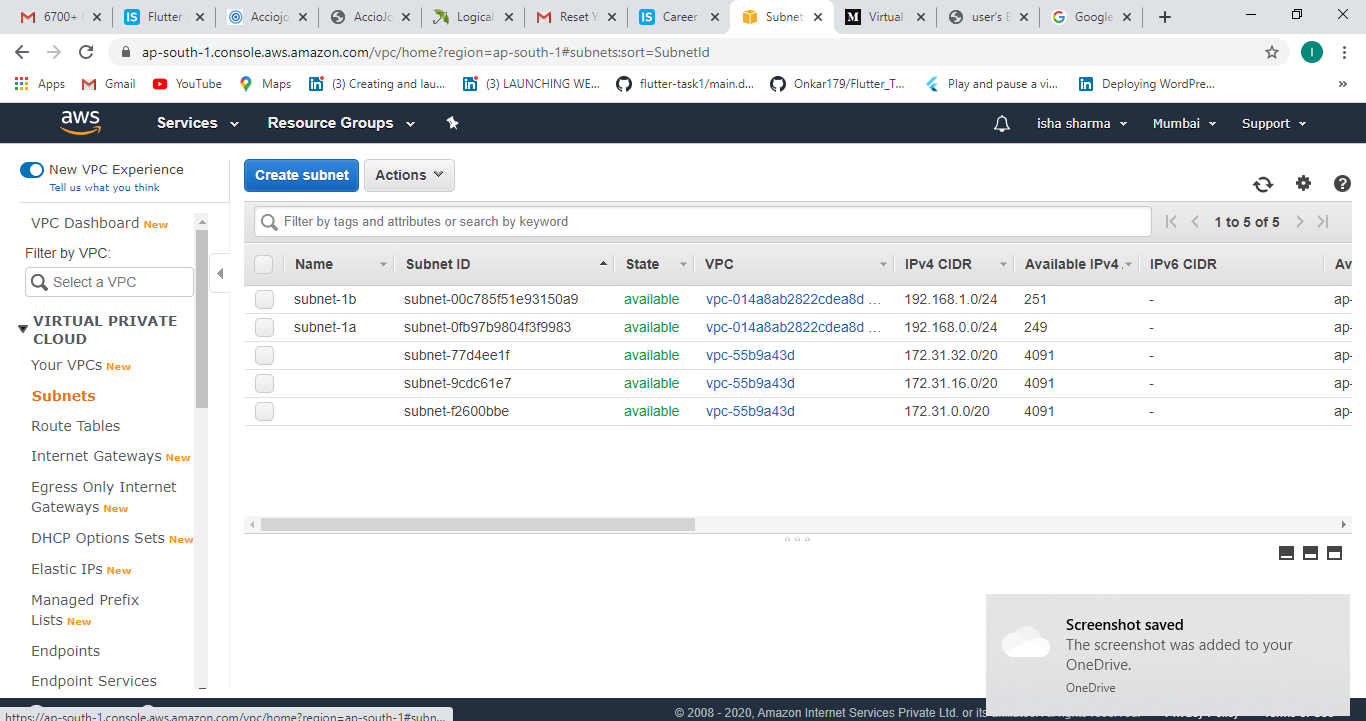


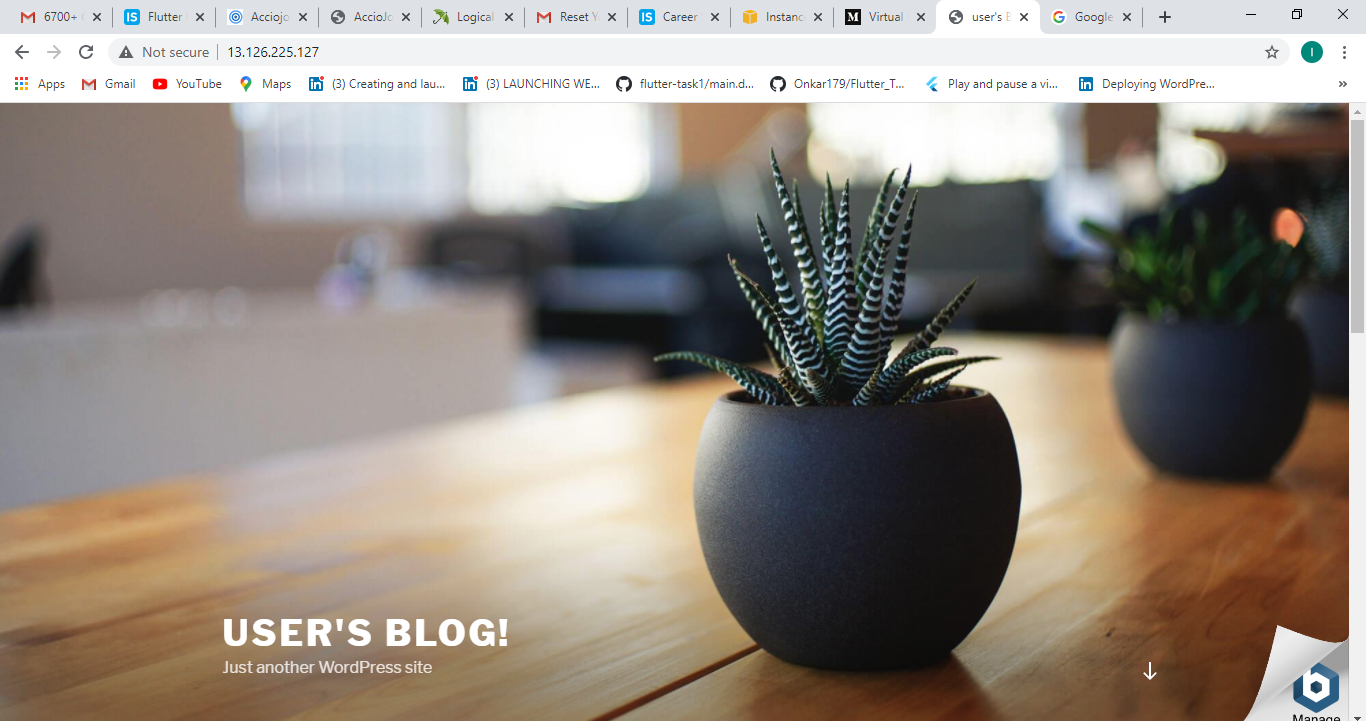


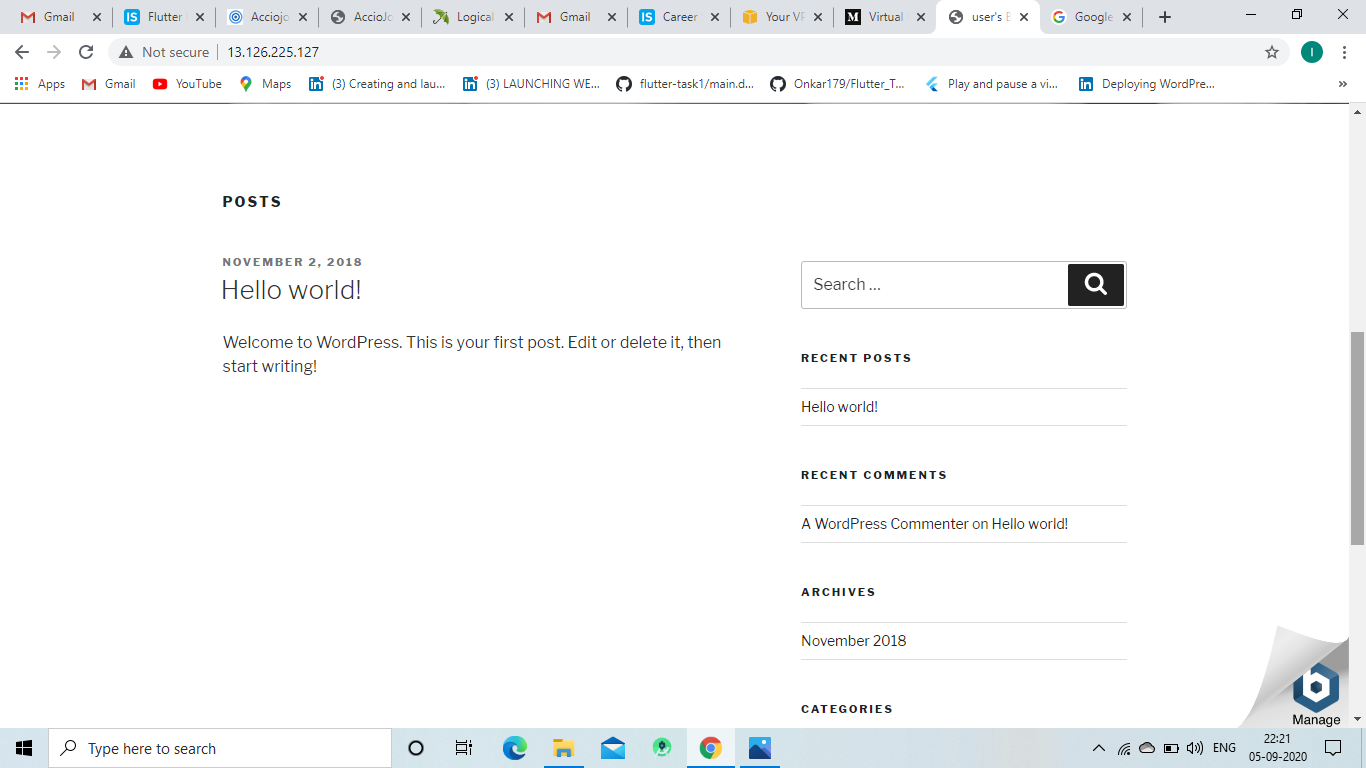












# **Task4 Completed**

**!!! Thank you vimal sir for your guidence and sharing your knowledge!!!**

**Thank You**