**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.

**Problem 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.

**TASK:-**

1) Create VPC using Terraform.

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 to 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 the outside world, update and associate it with the public subnet.

5) Launch an ec2 instance that 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 the instance for further login into it.

6) Launch an ec2 instance that has MYSQL setup already with security group allowing port 3306 in a private subnet so that our WordPress VM can connect with the same.

Also, attach the key with the same.

**To access our AWS account :-**

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

**# Creating a VPC**

resource “aws\_vpc” “myvpc” {  
cidr\_block = “192.168.0.0/16”  
instance\_tenancy = “default”  
enable\_dns\_hostnames = “true”  
tags = {  
Name = “myvpc”  
}  
}  
**# Creating Public subnet.**

resource “aws\_subnet” “subnet-1a” {  
vpc\_id = aws\_vpc.myvpc.id  
cidr\_block = “192.168.0.0/24”  
availability\_zone = “ap-south-1a”  
map\_public\_ip\_on\_launch = “true”  
depends\_on = [  
aws\_vpc.myvpc,  
]  
tags = {  
Name = “subnet-1a”  
}  
}

**# Creating a private subnet.**

resource “aws\_subnet” “subnet-1b” {  
vpc\_id = aws\_vpc.myvpc.id  
cidr\_block = “192.168.1.0/24”  
availability\_zone = “ap-south-1b”  
depends\_on = [  
aws\_vpc.myvpc,  
]  
tags = {  
Name = “subnet-1b”  
}  
}

**# Creating an internet gateway**

resource “aws\_internet\_gateway” “myigw” {  
vpc\_id = aws\_vpc.myvpc.id  
depends\_on = [  
aws\_vpc.myvpc,  
]  
tags = {  
Name= “myigw”  
}  
}

**# Creating a routing table**

resource “aws\_route\_table” “myrt” {  
vpc\_id = aws\_vpc.myvpc.id  
  
route {  
cidr\_block = “0.0.0.0/0”  
gateway\_id = aws\_internet\_gateway.myigw.id  
}

depends\_on = [  
aws\_vpc.myvpc,  
]

tags = {  
Name = “myrt”  
}  
}

**# Associating the routing table to subnet1**

resource “aws\_route\_table\_association” “assoc” {  
subnet\_id = aws\_subnet.subnet-1a.id  
route\_table\_id = aws\_route\_table.myrt.id

depends\_on = [  
aws\_subnet.subnet-1a,  
]  
}

**# creating a security group for WordPress**

resource “aws\_security\_group” “wp-sg” {  
name = “wp-sg”  
description = “allows ssh and http protocol”  
vpc\_id = aws\_vpc.myvpc.id

ingress {  
description = “Allow ssh”  
from\_port = 22  
to\_port = 22  
protocol = “tcp”  
cidr\_blocks = [“0.0.0.0/0”]  
}

ingress {  
description = “Allow http”  
from\_port = 80  
to\_port = 80  
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”]  
}

depends\_on = [  
aws\_vpc.myvpc,  
]

tags = {  
Name = “wp-sg”  
}  
}

**# Creating security group for MySQL**

resource “aws\_security\_group” “mysql-sg” {  
name = “mysql-sg”  
description = “allows 3306 port”  
vpc\_id = aws\_vpc.myvpc.id

ingress {  
description = “Allow 3306”  
from\_port = 3306  
to\_port = 3306  
protocol = “tcp”  
security\_groups = [aws\_security\_group.wp-sg.id]  
}

egress {  
from\_port = 0  
to\_port = 0  
protocol = “-1”  
cidr\_blocks = [“0.0.0.0/0”]  
}

depends\_on = [  
aws\_vpc.myvpc,  
aws\_security\_group.wp-sg,  
]

tags = {  
Name = “mysql-sg”  
}  
}

**# Launching WordPress in the public subnet**

resource “aws\_instance” “wordpress” {  
ami = “ami-000cbce3e1b899ebd”  
instance\_type = “t2.micro”  
subnet\_id = aws\_subnet.subnet-1a.id  
key\_name = “my-key”  
vpc\_security\_group\_ids = [aws\_security\_group.wp-sg.id]

depends\_on = [  
aws\_subnet.subnet-1a,  
aws\_security\_group.wp-sg,  
]

tags = {  
Name = “wordpress”  
}  
}

**# Launching MySQL in a private subnet**

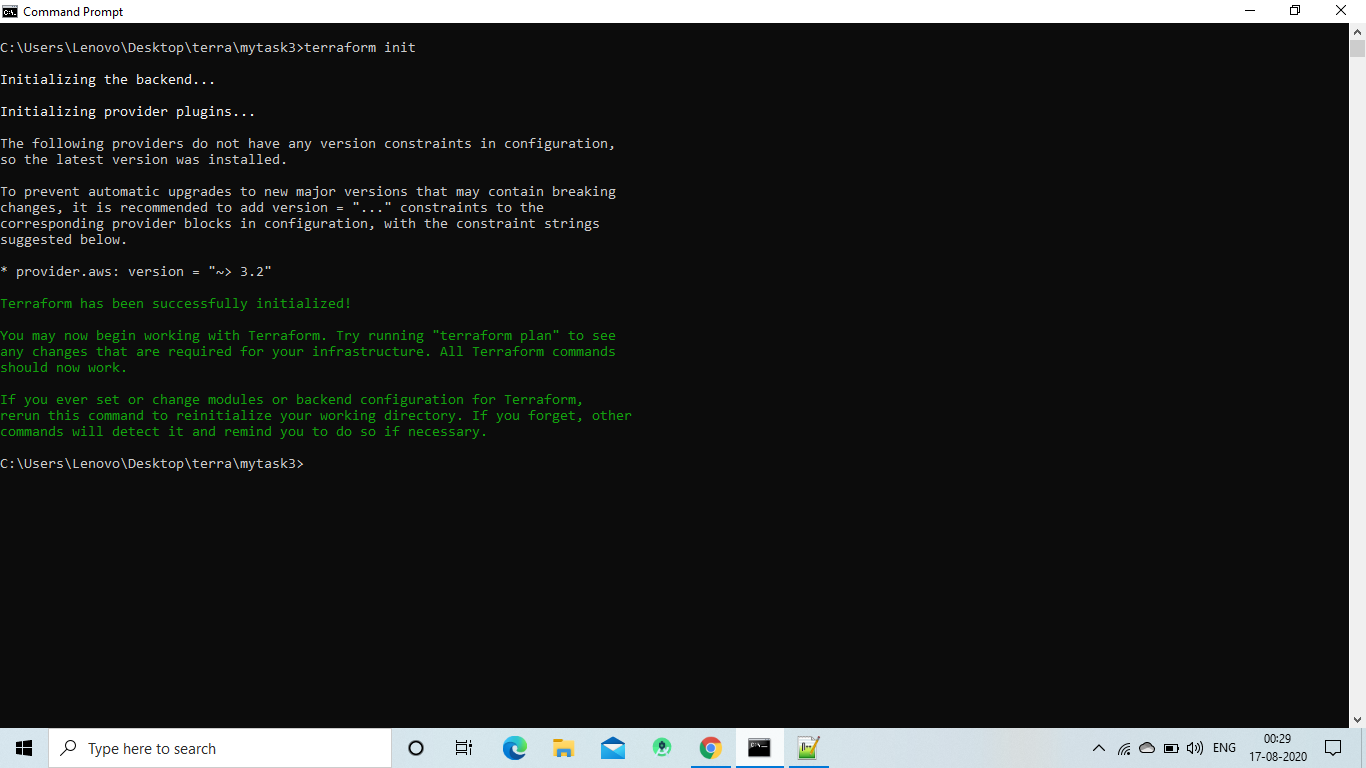
resource “aws\_instance” “my-sql” {  
ami = “ami-000cbce3e1b899ebd”  
instance\_type = “t2.micro”  
subnet\_id = aws\_subnet.subnet-1a.id  
key\_name = “my-key”  
vpc\_security\_group\_ids = [aws\_security\_group.wp-sg.id]

depends\_on = [  
aws\_subnet.subnet-1a,  
aws\_security\_group.wp-sg,  
]

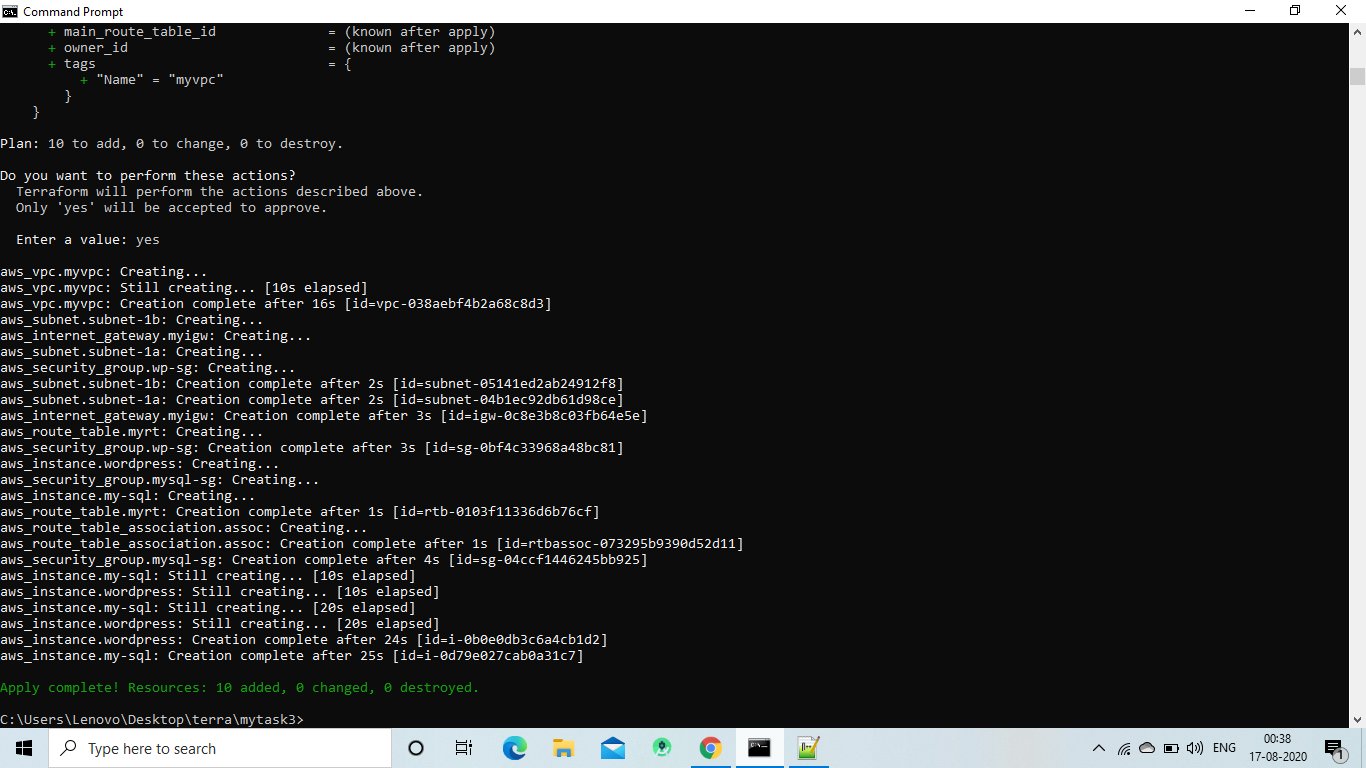
tags = {  
Name = “my-sql”  
}  
}

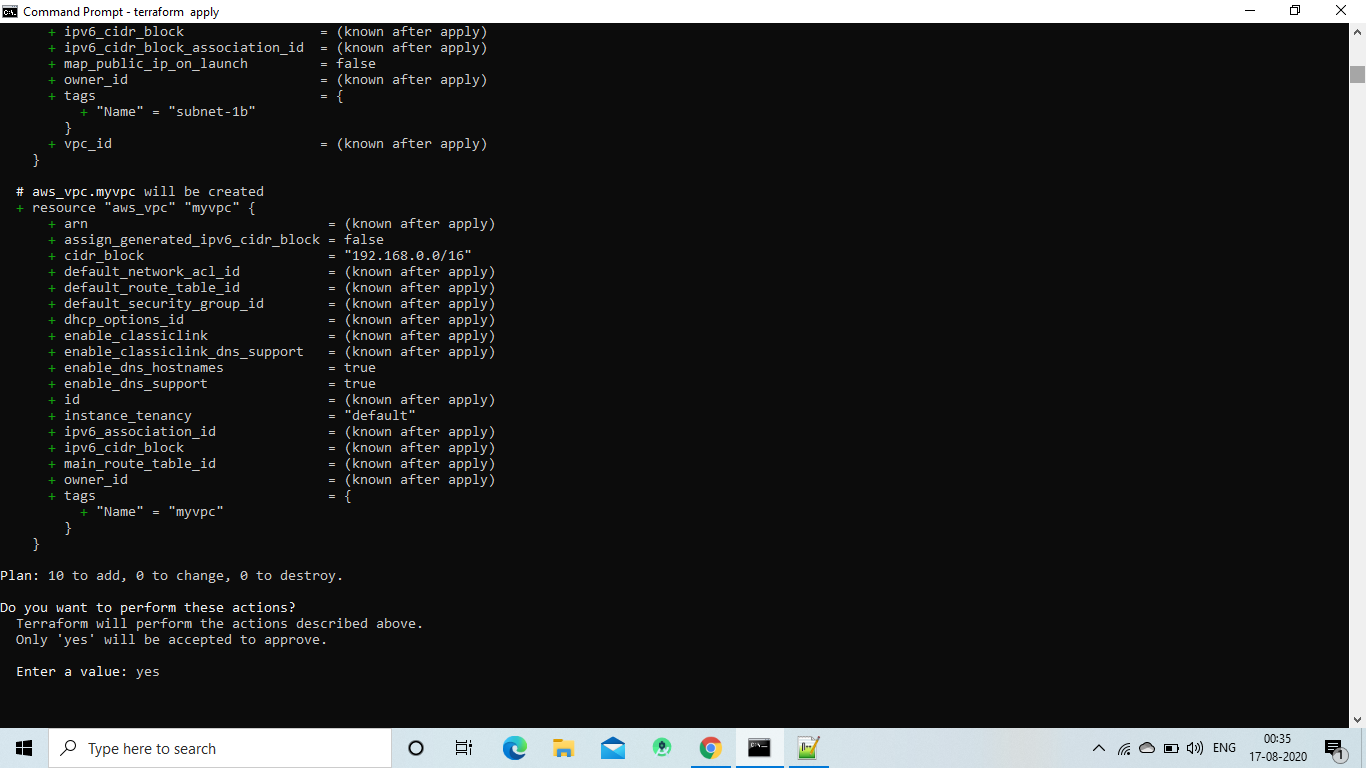
**Now before running these files we first need to initialize the directory and then we will apply or run it.**

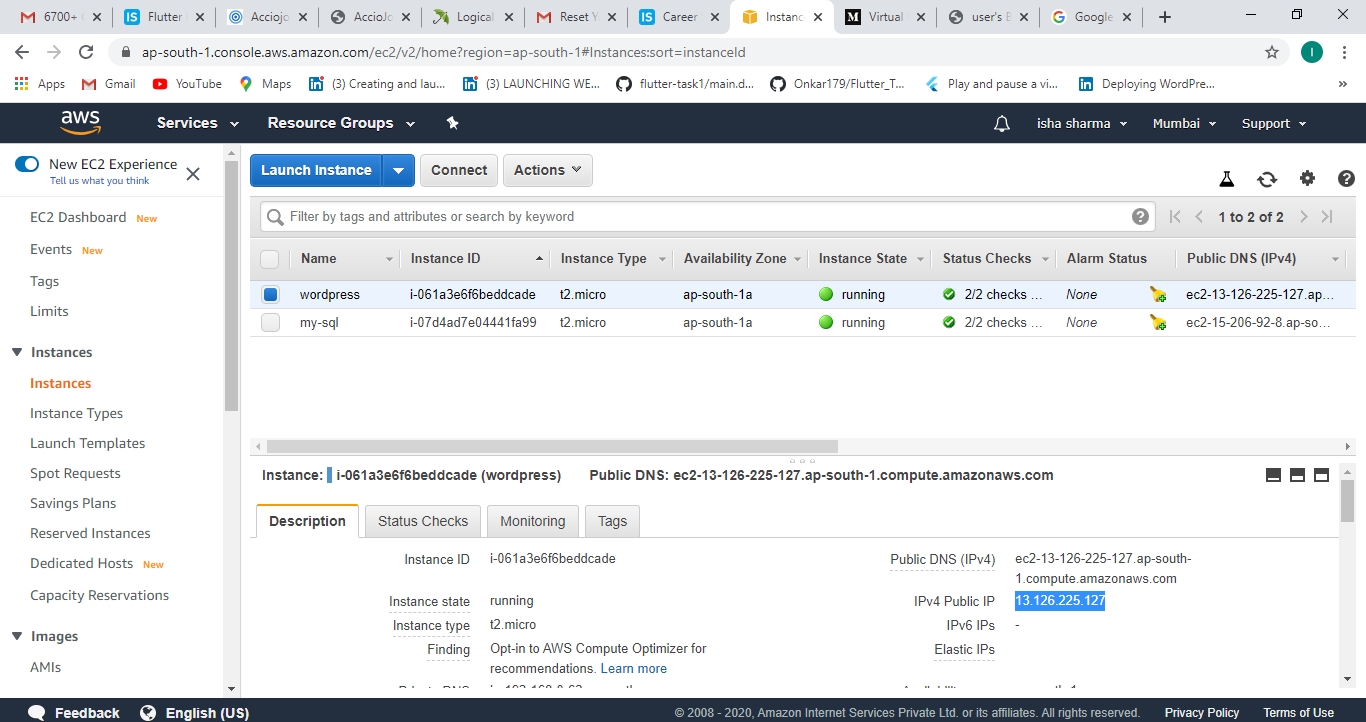
terraform init

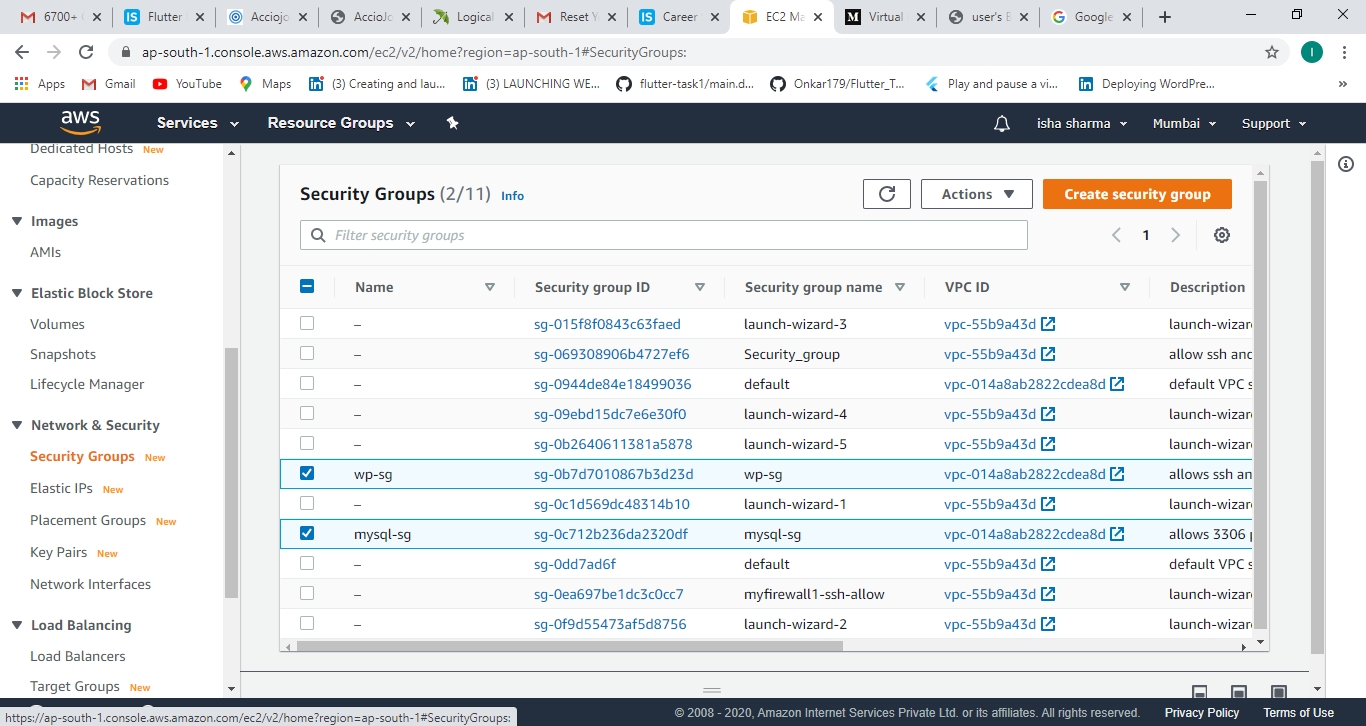
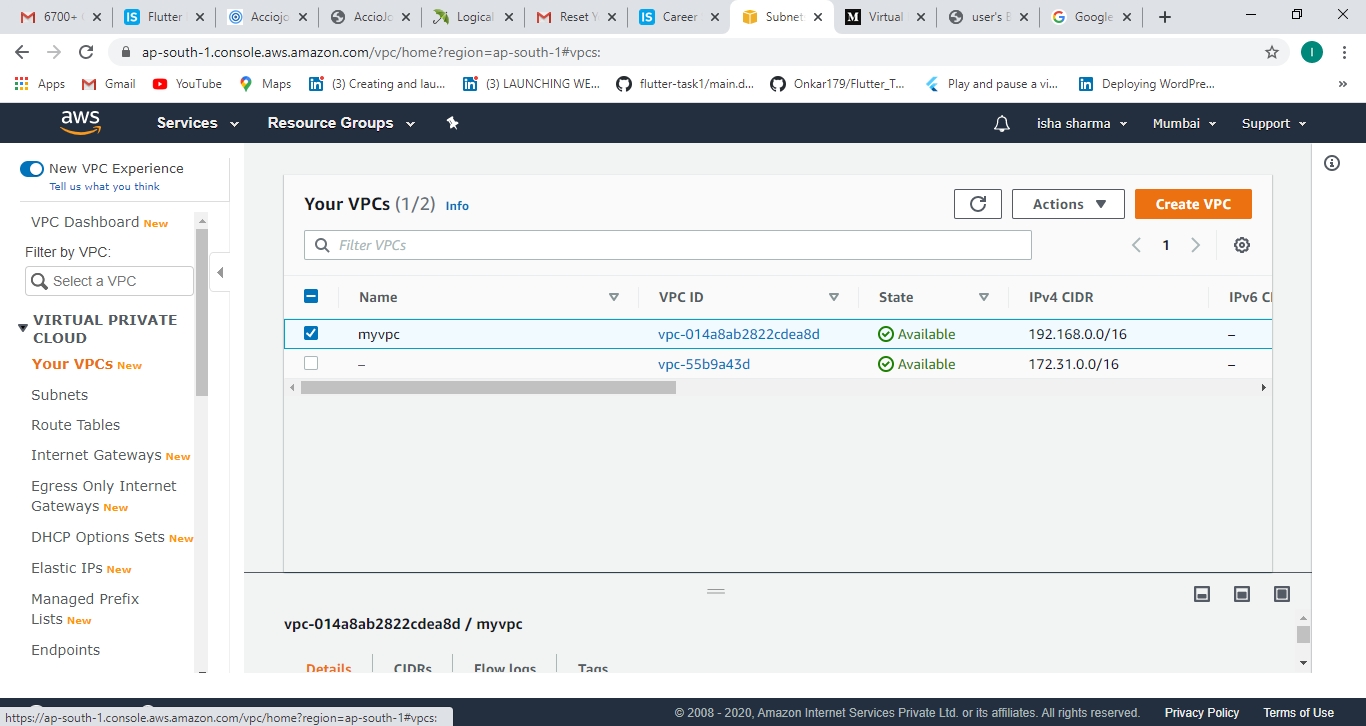


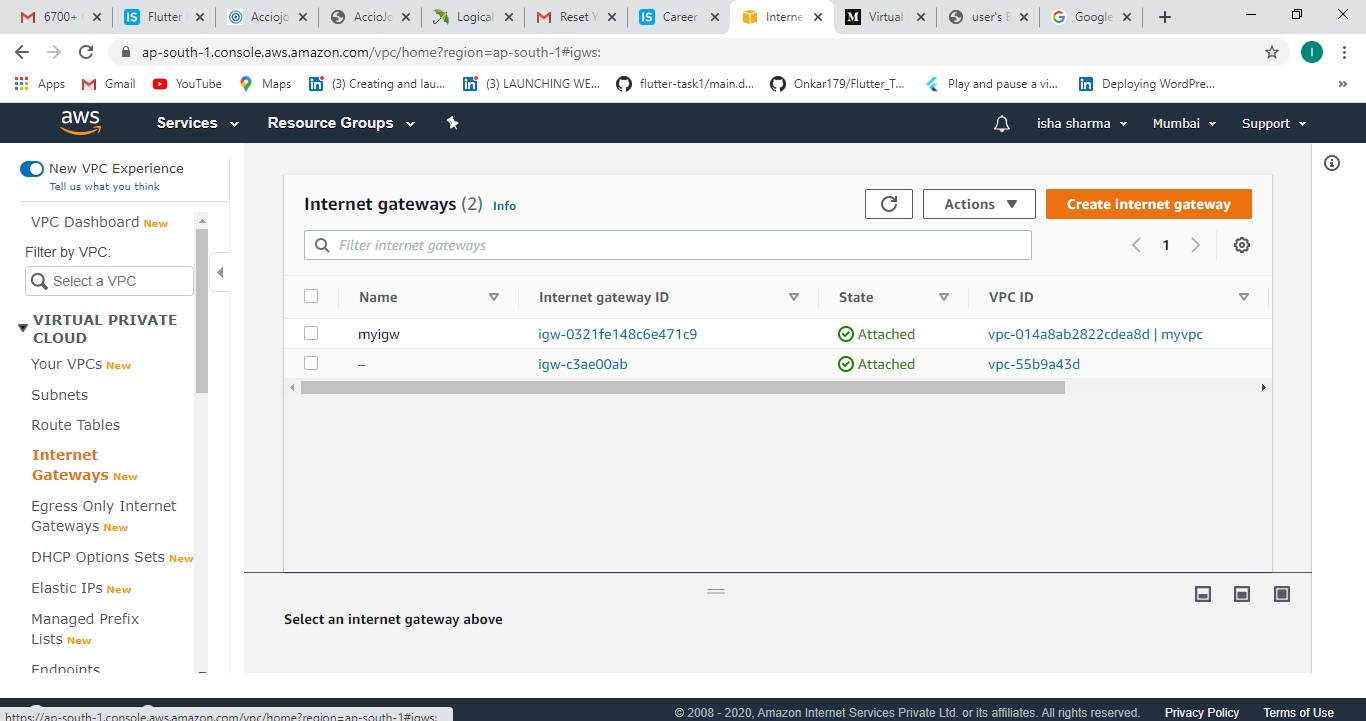
terraform apply

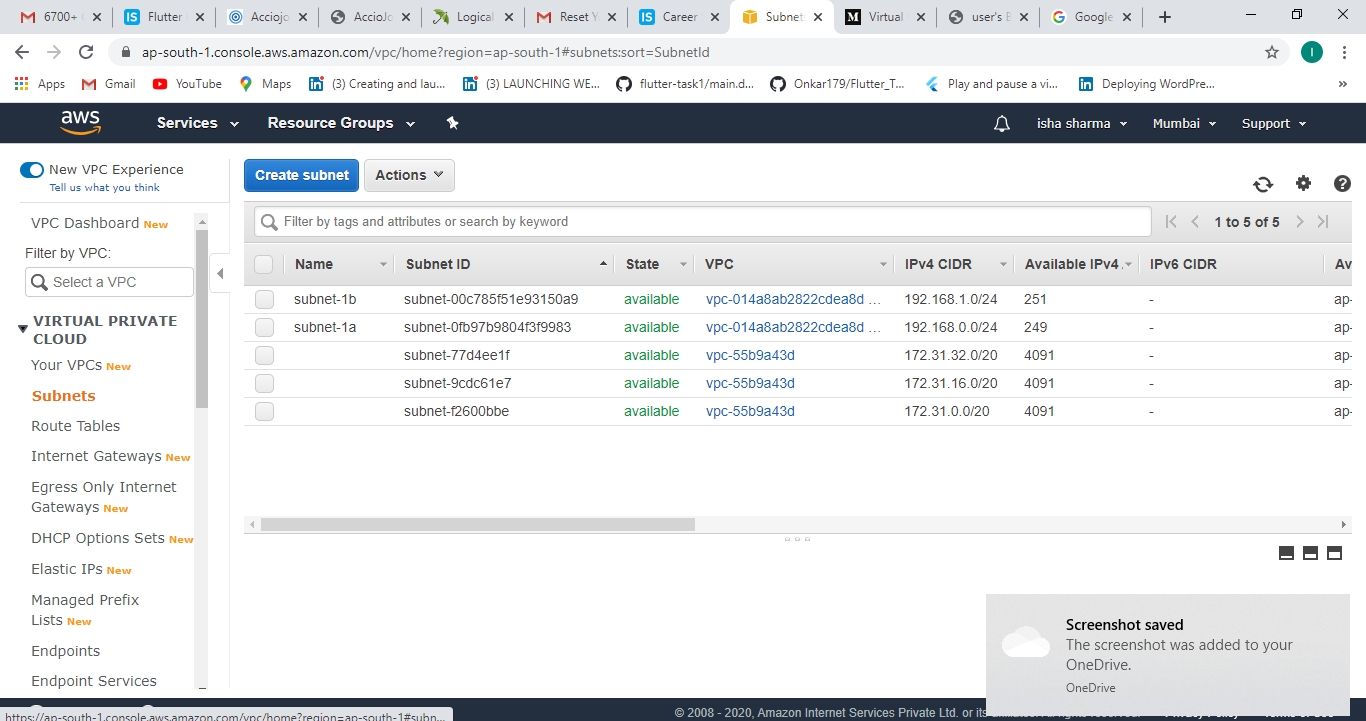


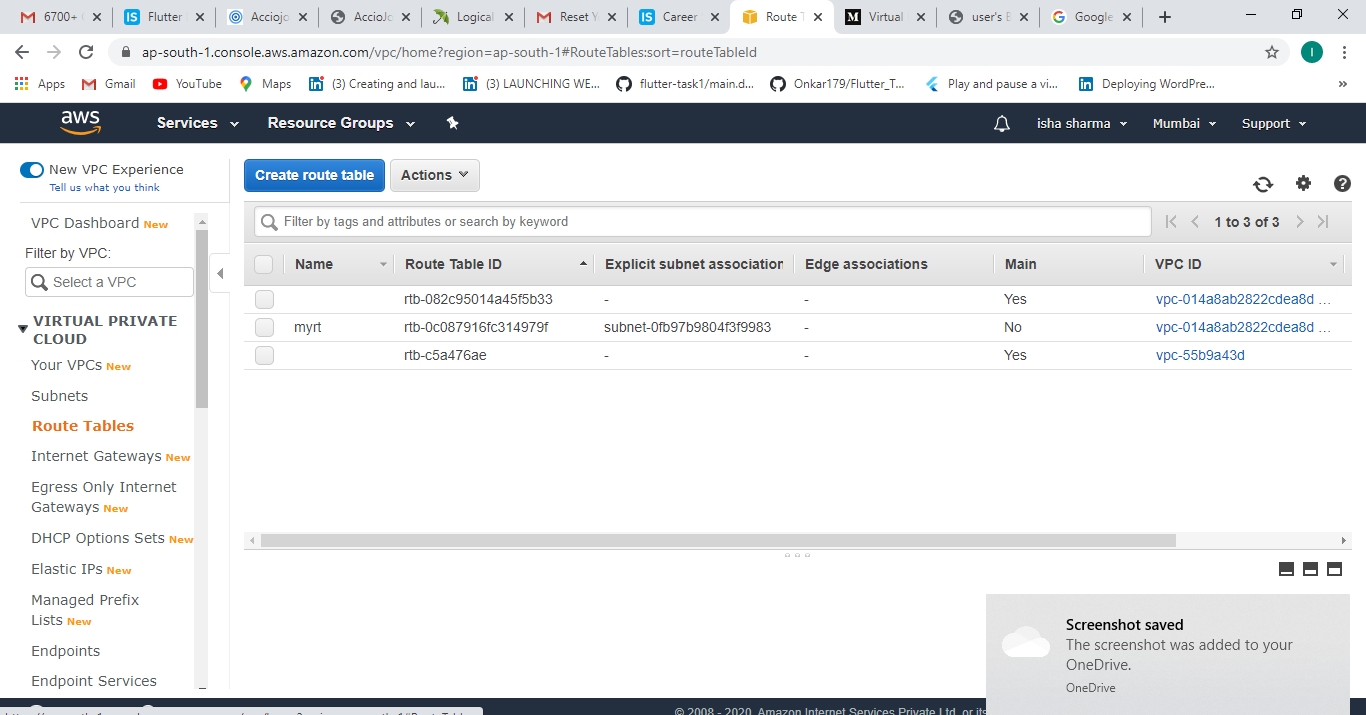
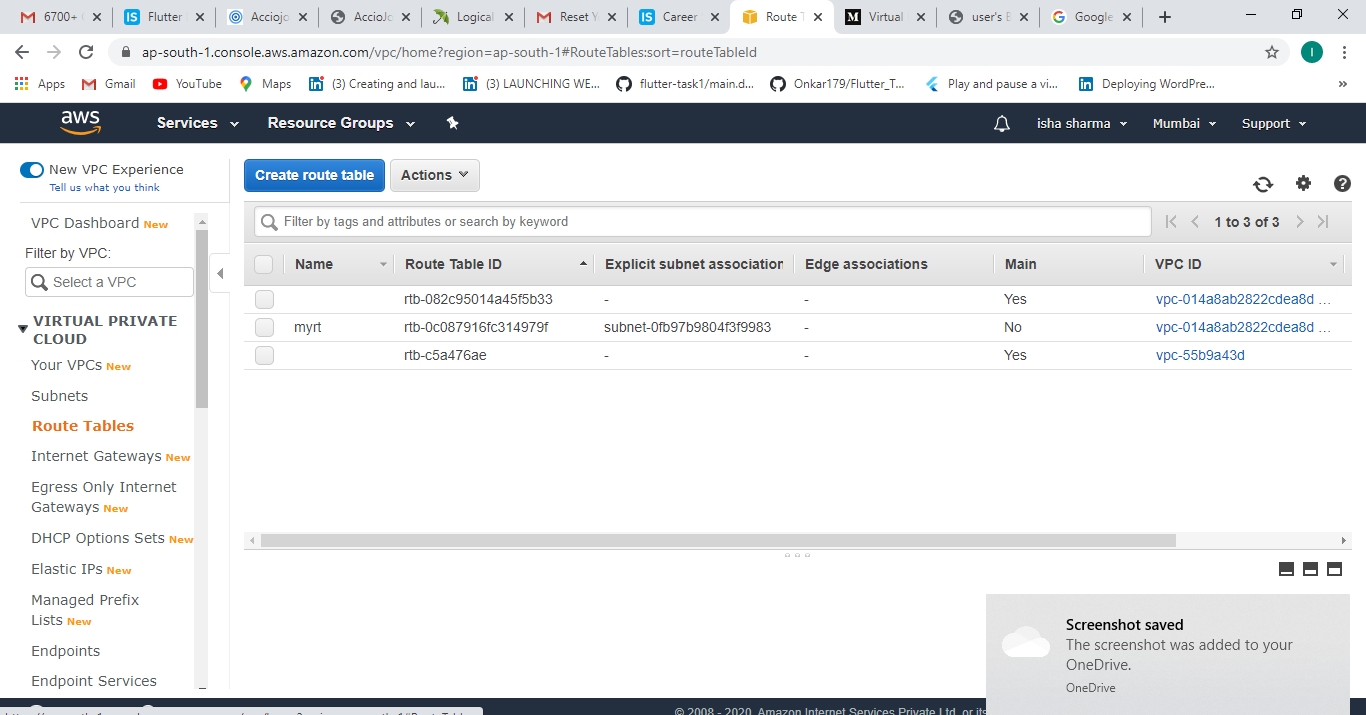
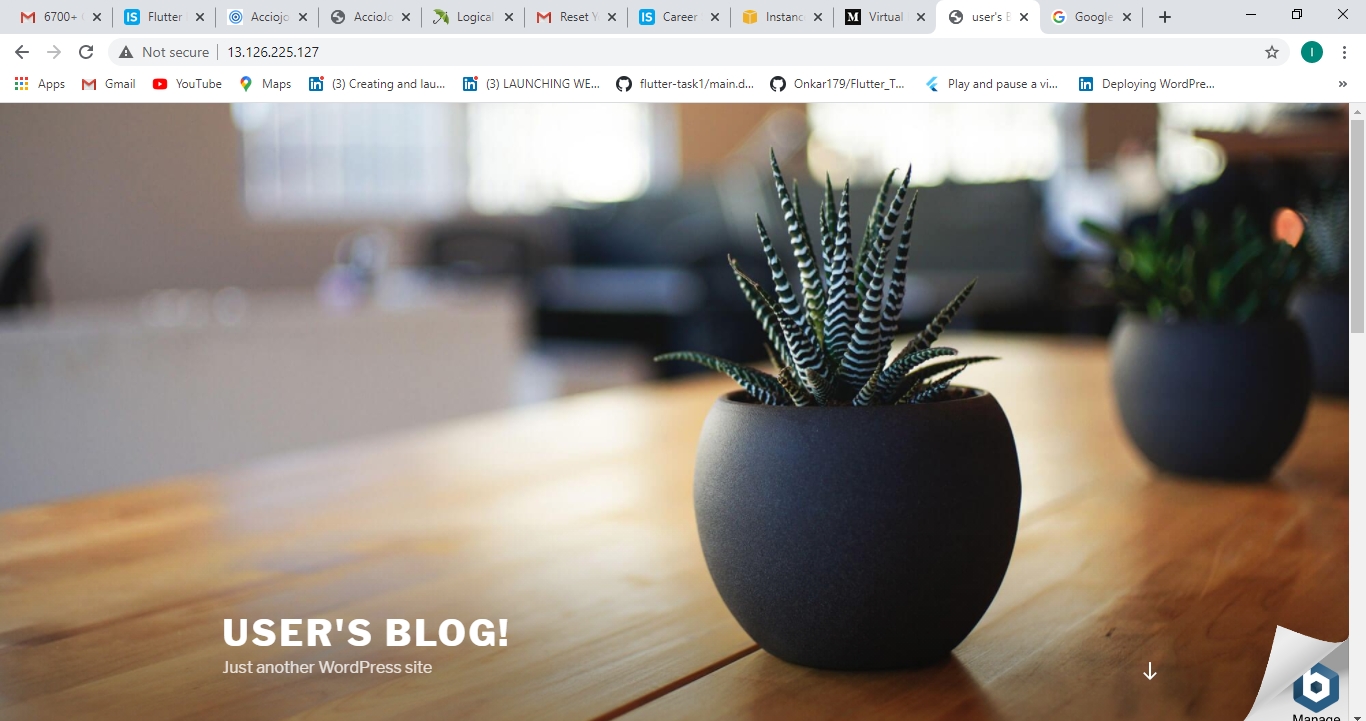


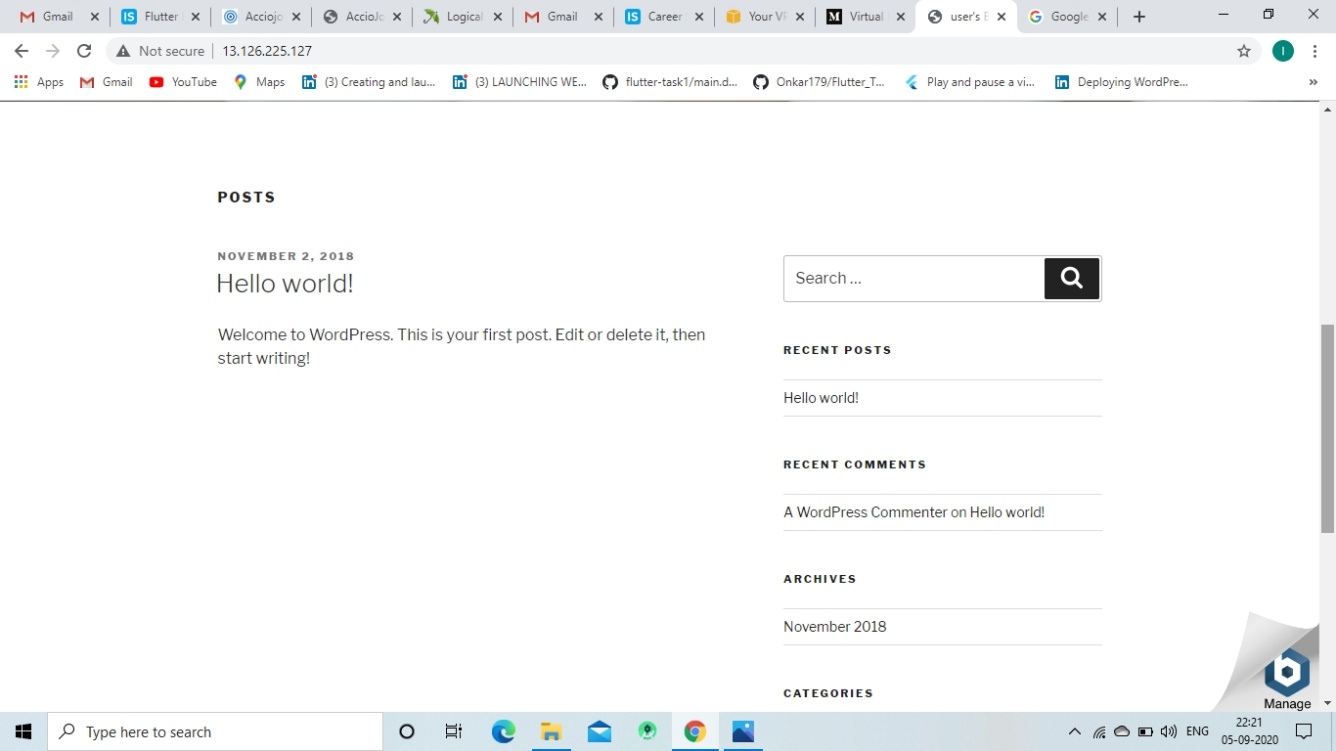


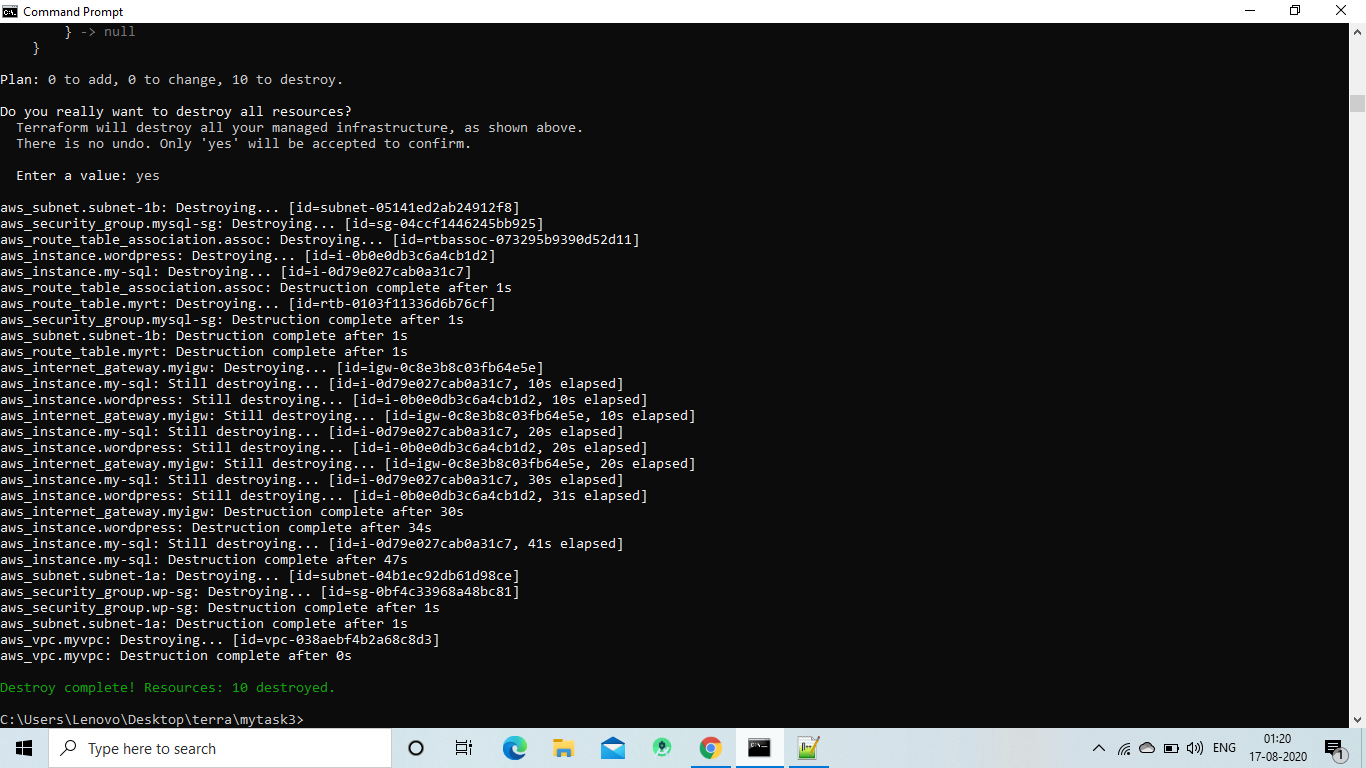
****

****

****

****

****terraform destroy



**Task3 Completed**

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

**Thank You**