***Set up VPC with Public & Private subnets using Terraform***

VPC - It's a virtual private cloud which is logically isolated from other networks.

VPC is a Private Network that you define on AWS platform and create resources in it. It is logically isolated from other network & is most secured network. Typically one need to create resources in Public subnet which are exposed to public world. E.g. web server, word press etc. The resources which are not exposed to internet or outside of the VPC resides in Private subnets. E.g. Databases.

VPC provides Infrastructure as a Service (IAAS) as well as Network as a Service (NAAS). It's secure & by default AWS creates VPC with 3 subnets.

We can launch below infrastructure using terraform code:

The private subnet is inaccessible to the internet (both in and out).

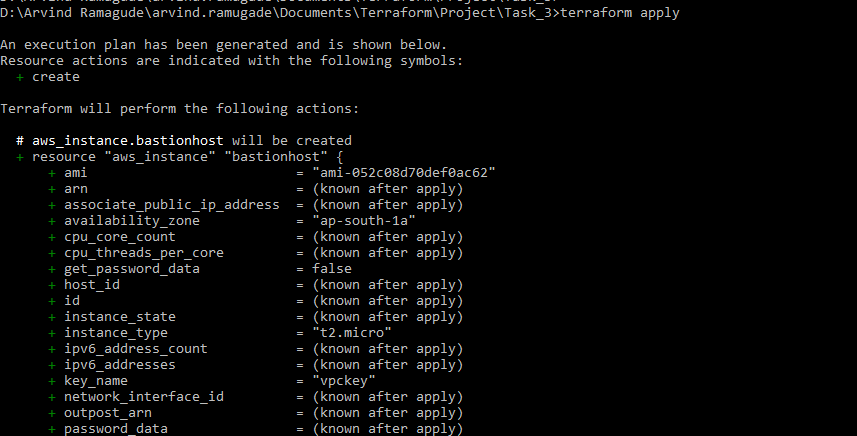
The public subnet is accessible; just dependent on the configuration of the security groups

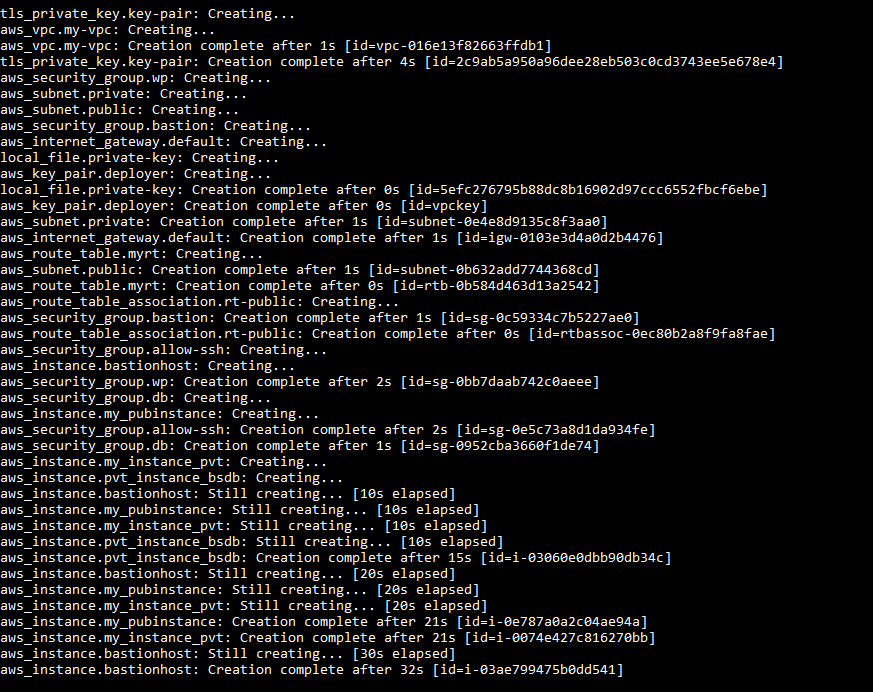
Routing is handled like this:

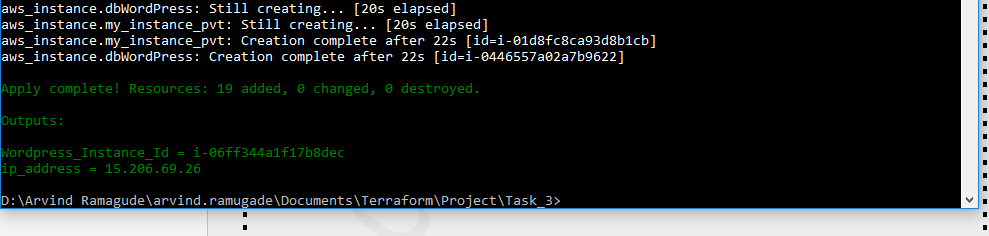
Private subnet is routed through the NAT instance.

Public subnet is routed directly to the internet gateway.

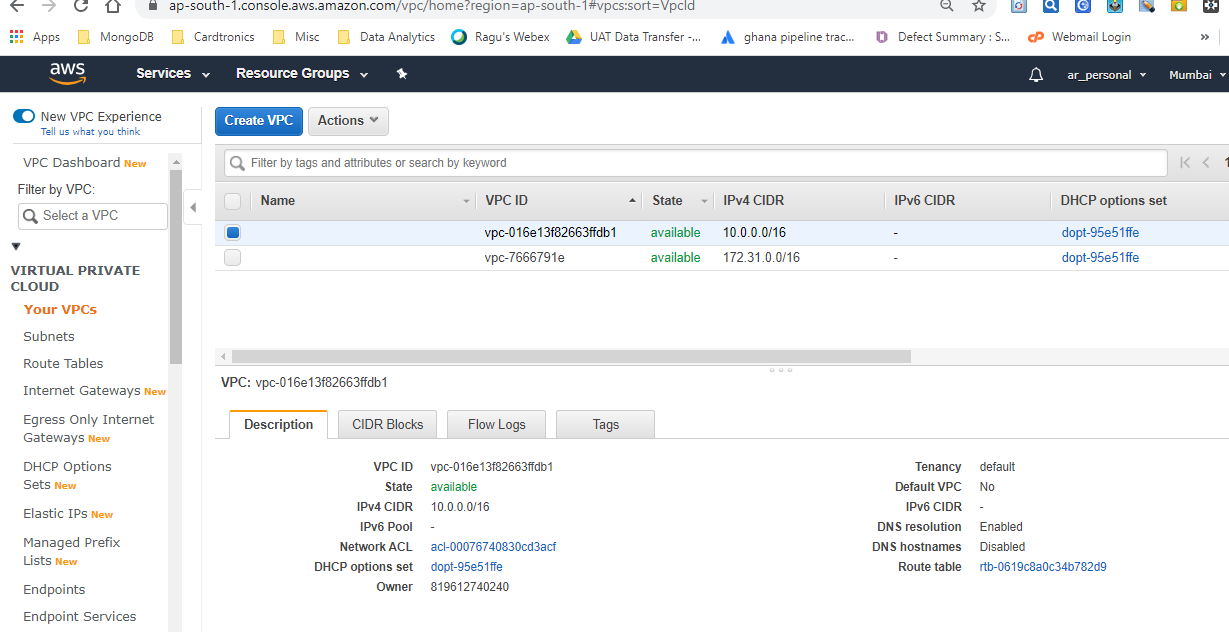








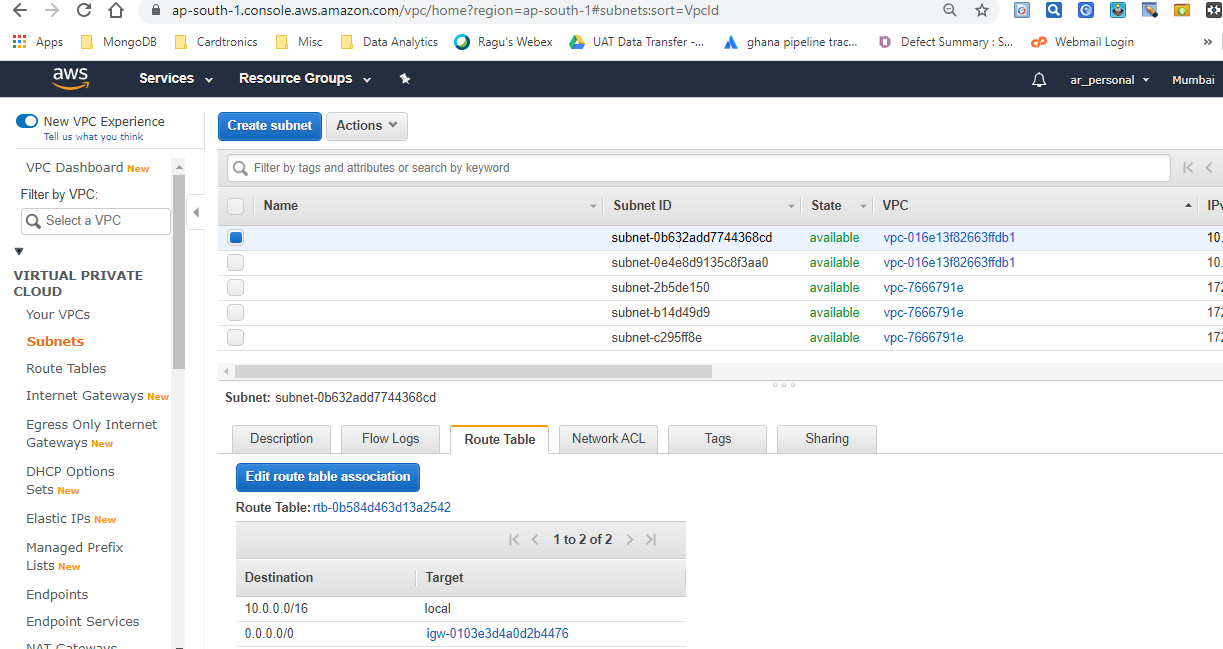
We can verify the creation of **VPC** through AWS console as follows:



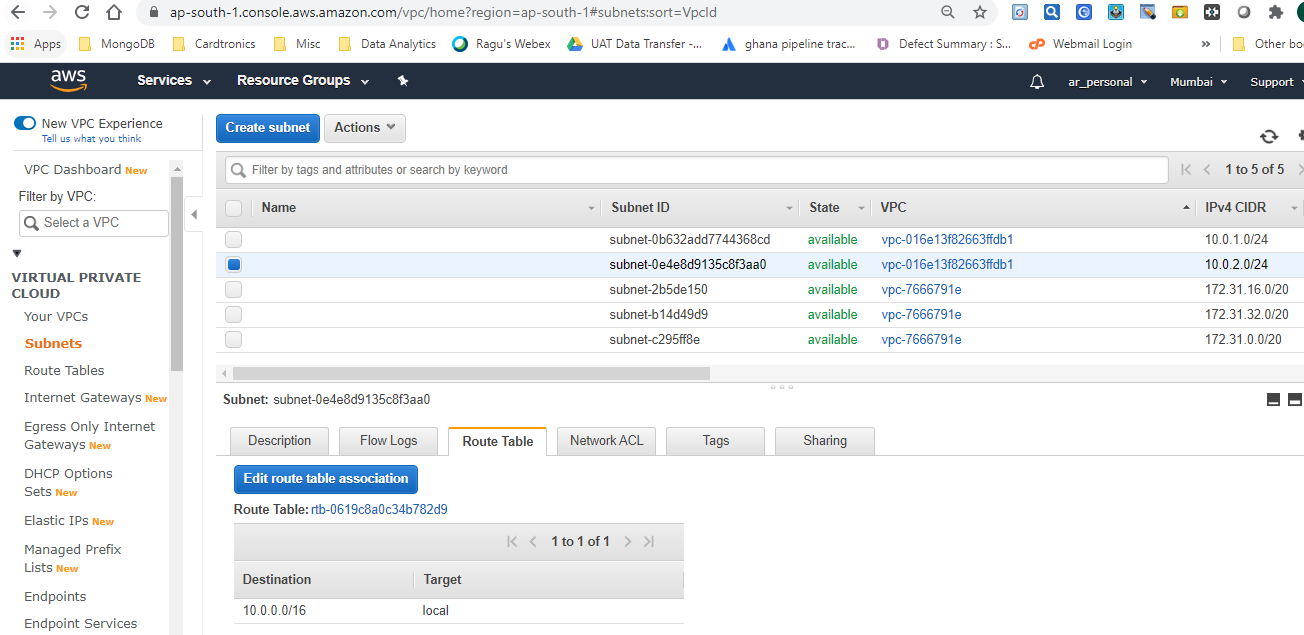
Subnets is logical subdivision of VPC. By default instances are launched in any one of the 3 subnets unless we specify the subnet while launching it. VM should be launched inside subnet. It's not possible to launch VM/EC2 instance without subnet.

By default DNS hostname is disabled hence we can't see the DNS name once the instance is launched. However, this setting can be changed inside the VPC for that subnet so that all the VM's launched/already launched will reflect DNS names.

**Public Subnet with Route Table**



**Private Subnet with Route Table**



AWS reserves total 5 IPs (1st 4 IP & last IP) in each subnet.

* 1st IP is reserved for Network.
* 2nd IP is reserved for Router.
* 3rd IP is reserved for DHCP server
* 4th is for future use
* 5th one is reserved for Network Broadcast.

This is the reason for each subnet we'll have 5 IP's less than the range.

e.g. 10.0.0.0/24 - ideally it should have 2 raise to 8 (32-24=8) i.e. 256 IP addresses. However, since AWS reserves 5 IP's for each subnet it will have only 251 IP addresses for use.

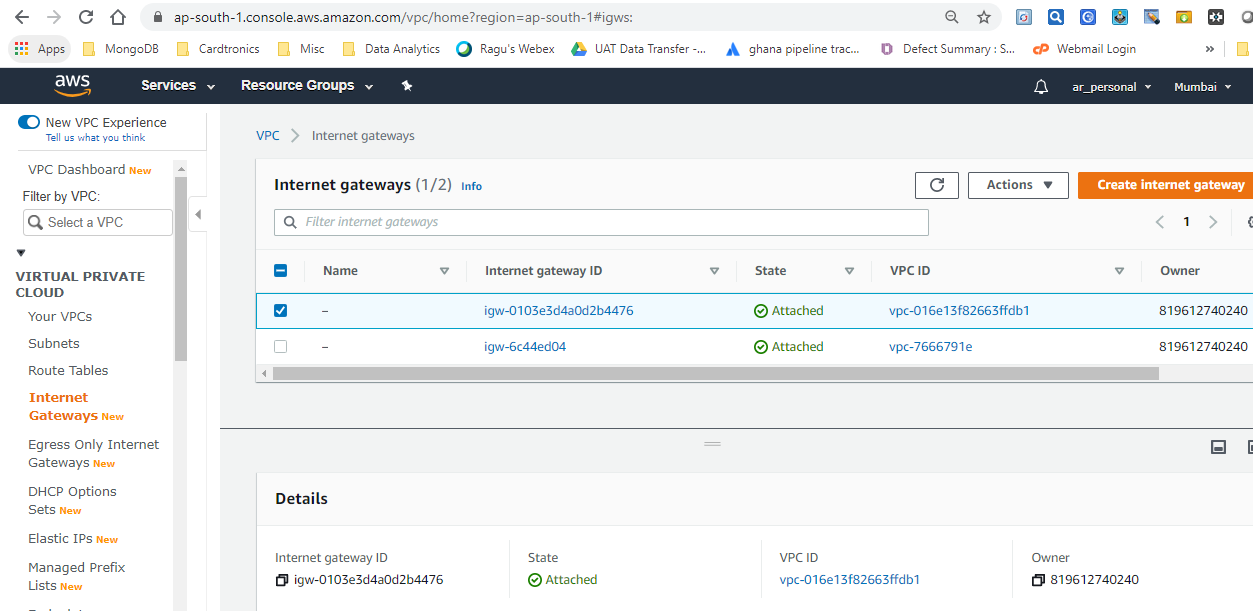
For every subnet DHCP created by AWS. DHCP provides public IP, Subnet Mask & Internet Gateway (IGW) for each subnet.

Inside subnet there are switches & routers. All instances (VM) can communicate using private IP across different subnets.

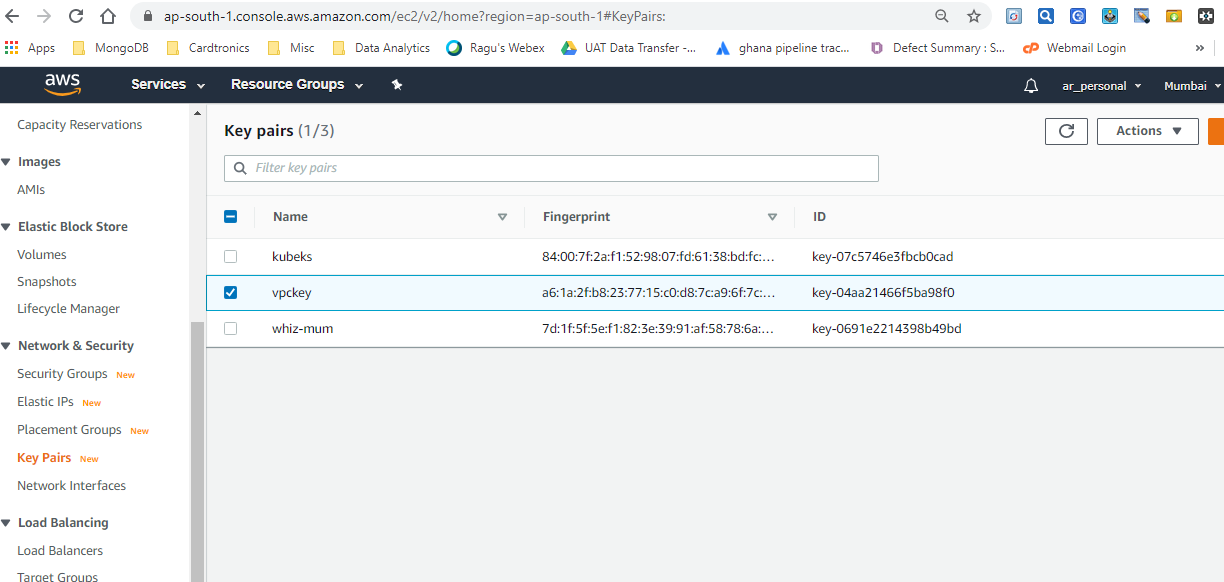
**Internet Gateway**

In AWS world, the public IP address of EC2 instance is actually the IP address of Router which is used to communicate outside the subnet. Router which is having public facing IP (0.0.0.0) is known as Internet Gateway (IGW). It uses both SNAT & DNAT terminologies. IGW is attached to VPC.

**Internet Gateway through AWS Console:**



**Key-pair creation**: Through AWS console we can verify the key (vpckey) which has been created.



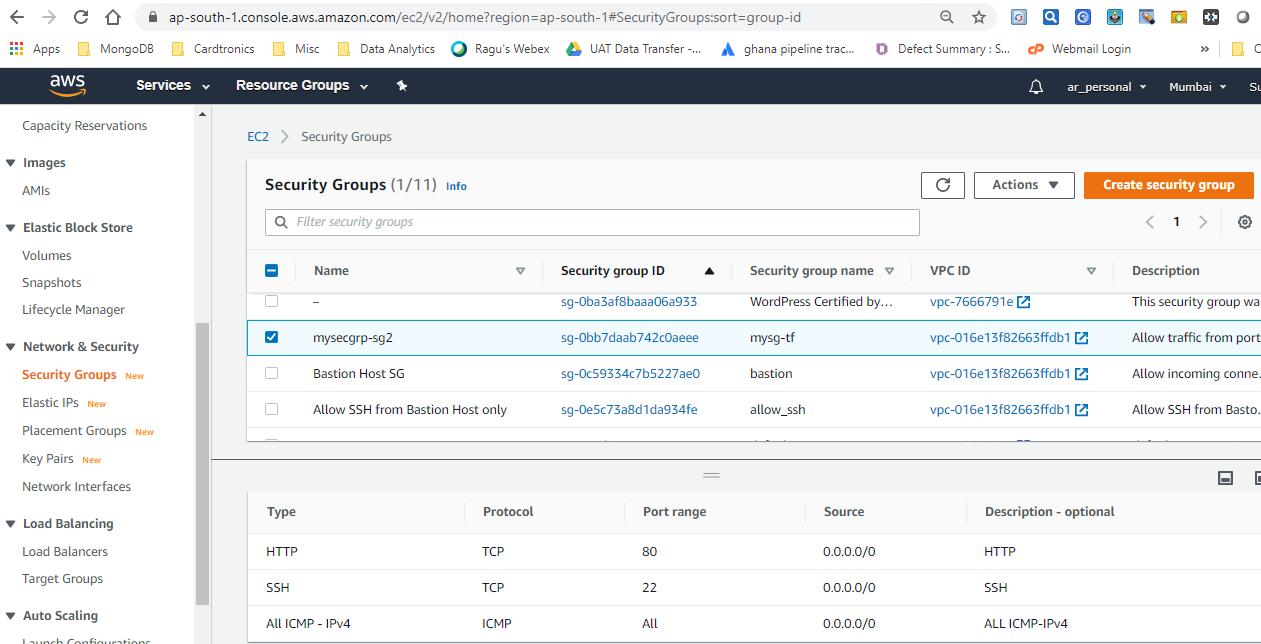
Security groups works like a firewall & it works at the instance level. It’s first layer of protection for an instance. A security group acts as a virtual firewall for your EC2 instances to control incoming and outgoing traffic.

Inbound rules control the incoming traffic to your instance, and outbound rules control the outgoing traffic from your instance.

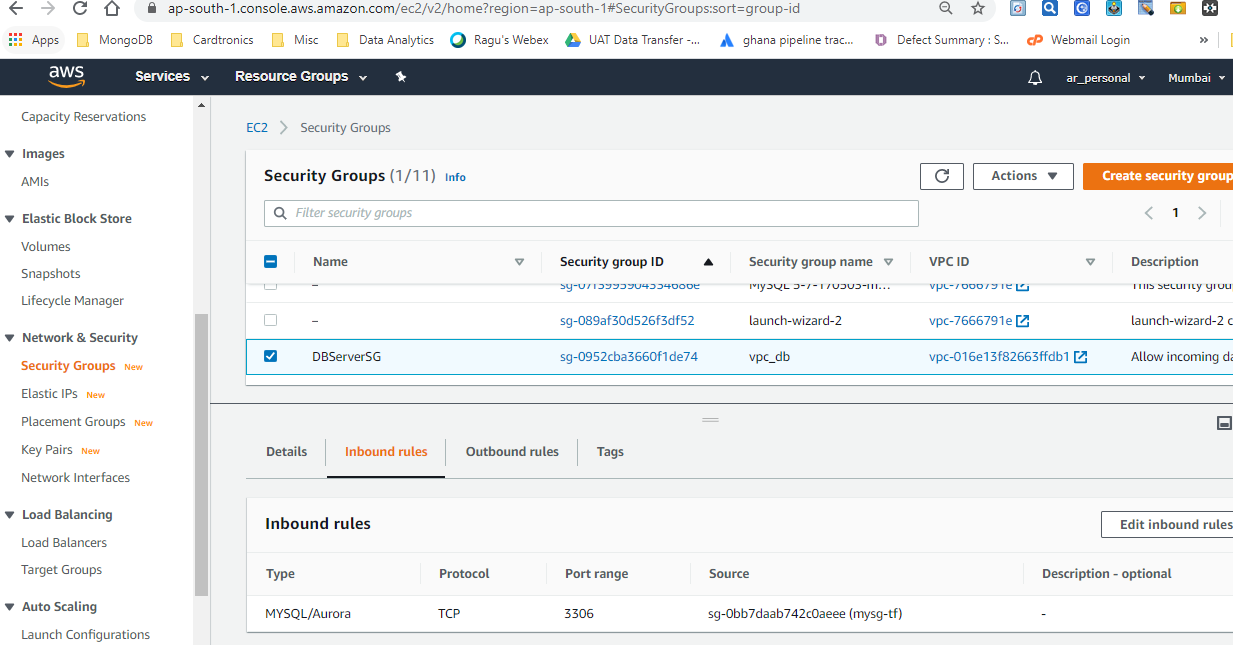
Inbound rules are designed to deal with incoming traffic whereas outbound rules are for outgoing traffic. In Private cloud inbound rule is known as ingress while outbound rule is known as egress rule. By default inbound/ingress rules are explicitly denied whereas outbound/egress rules are allowed.

**Security Group for WordPress**

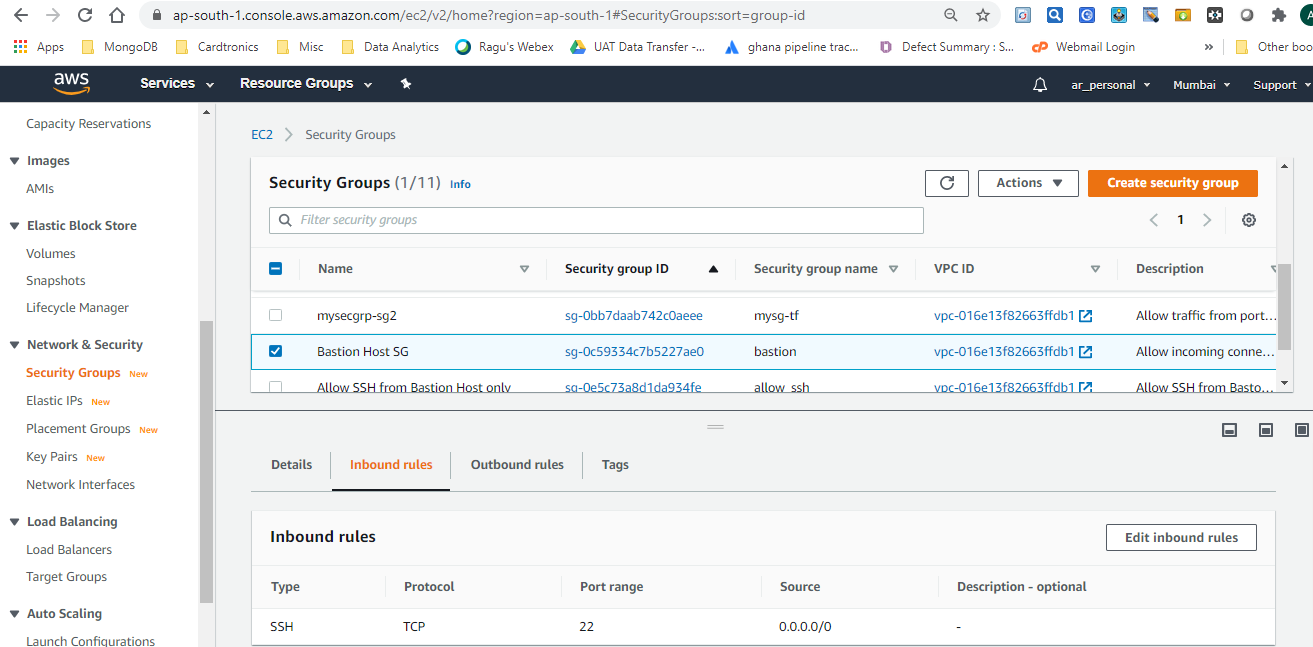
It allows HTTP, SSH & ICMP inbound traffic.



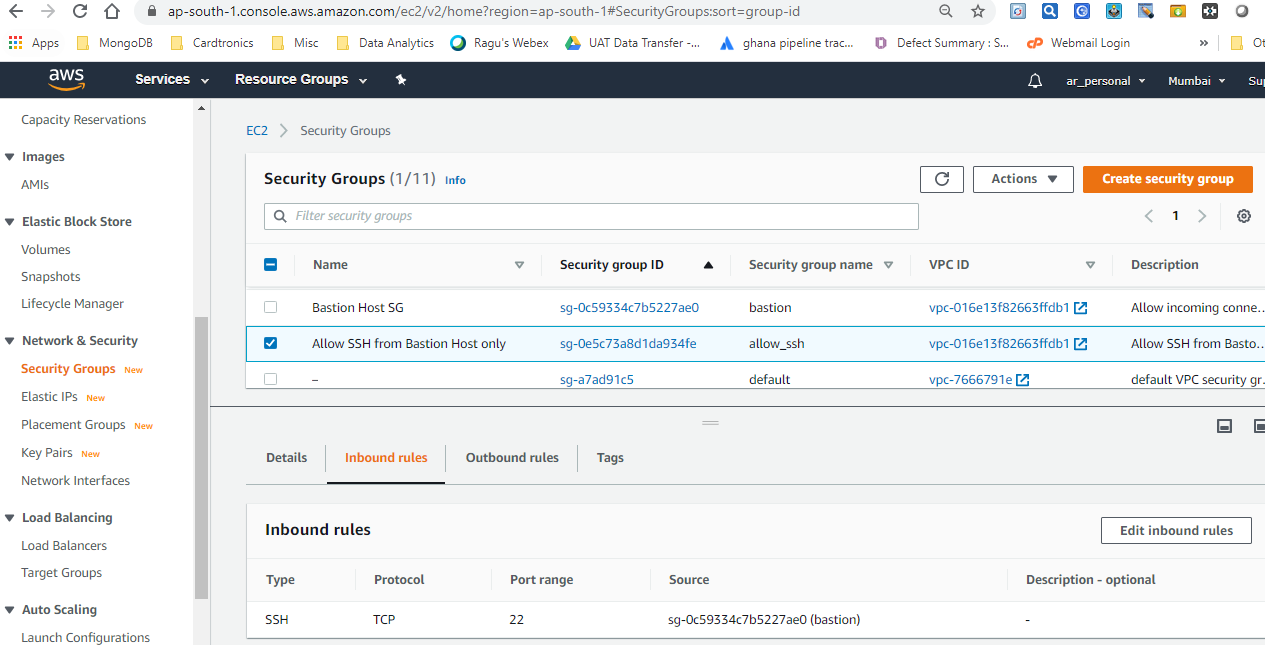
**DB Security Group** – This allows traffic only from WP security group created above.



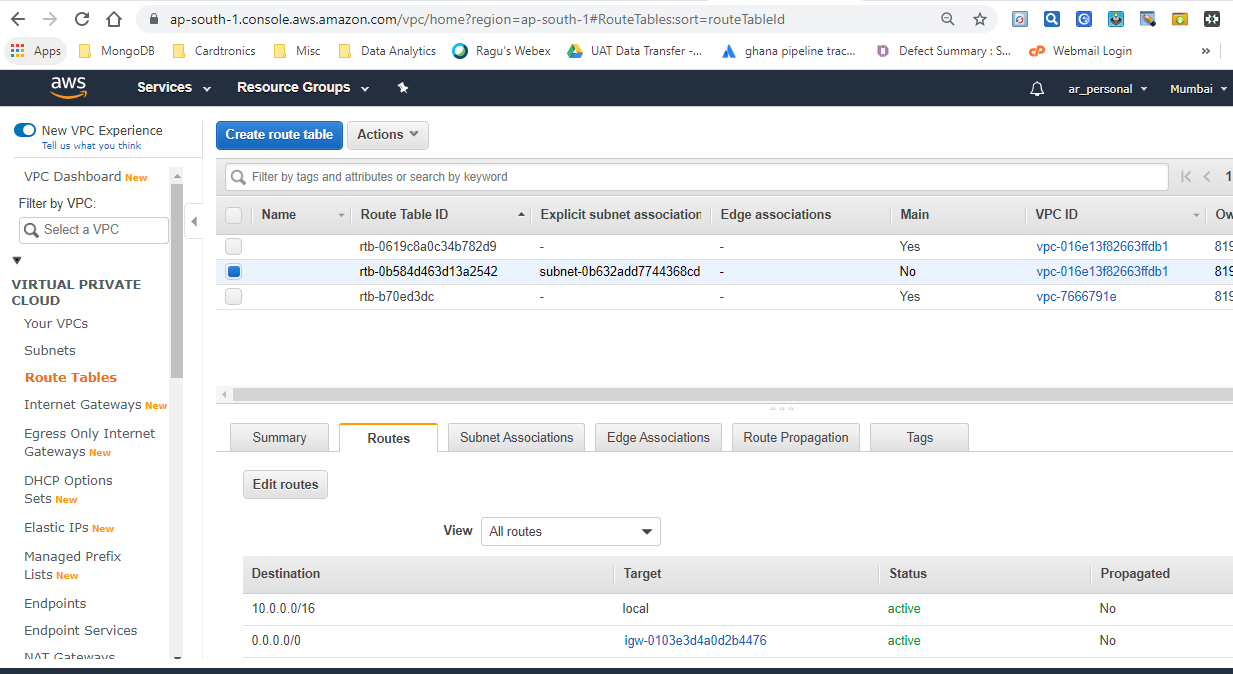
**Security group for Bastion Host** – This allows only SSH traffic from outside VPC.



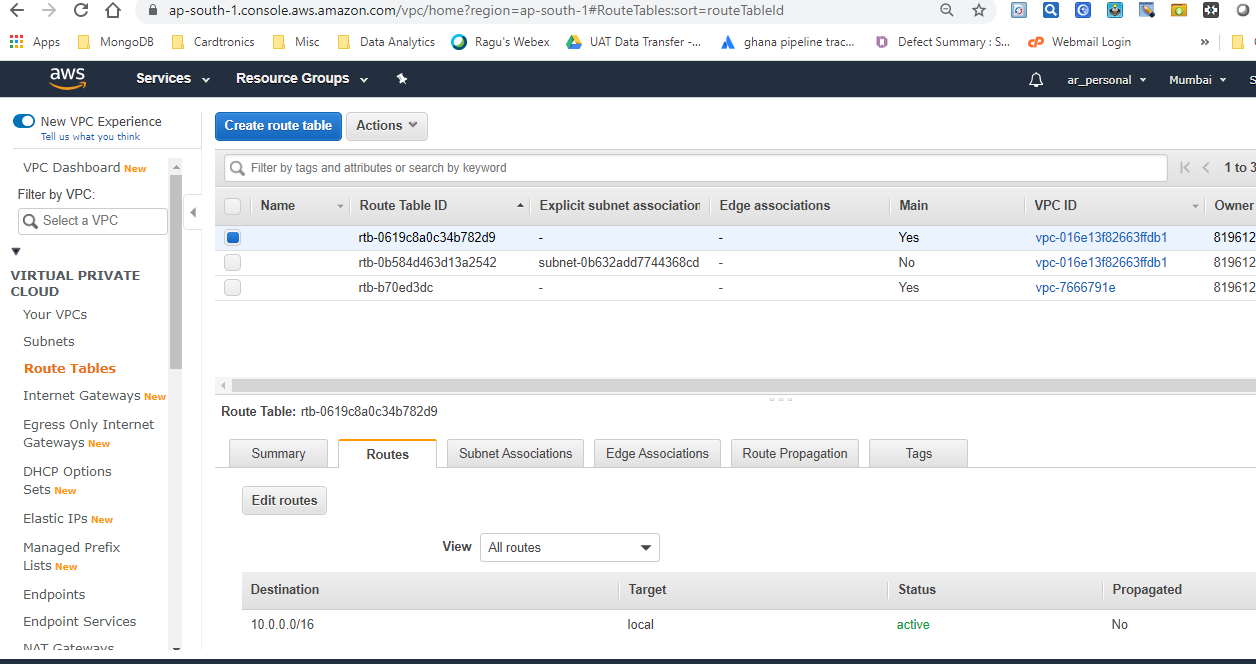
Below Security group **allows SSH from Bastion host only**. This can be attached to Private instances.



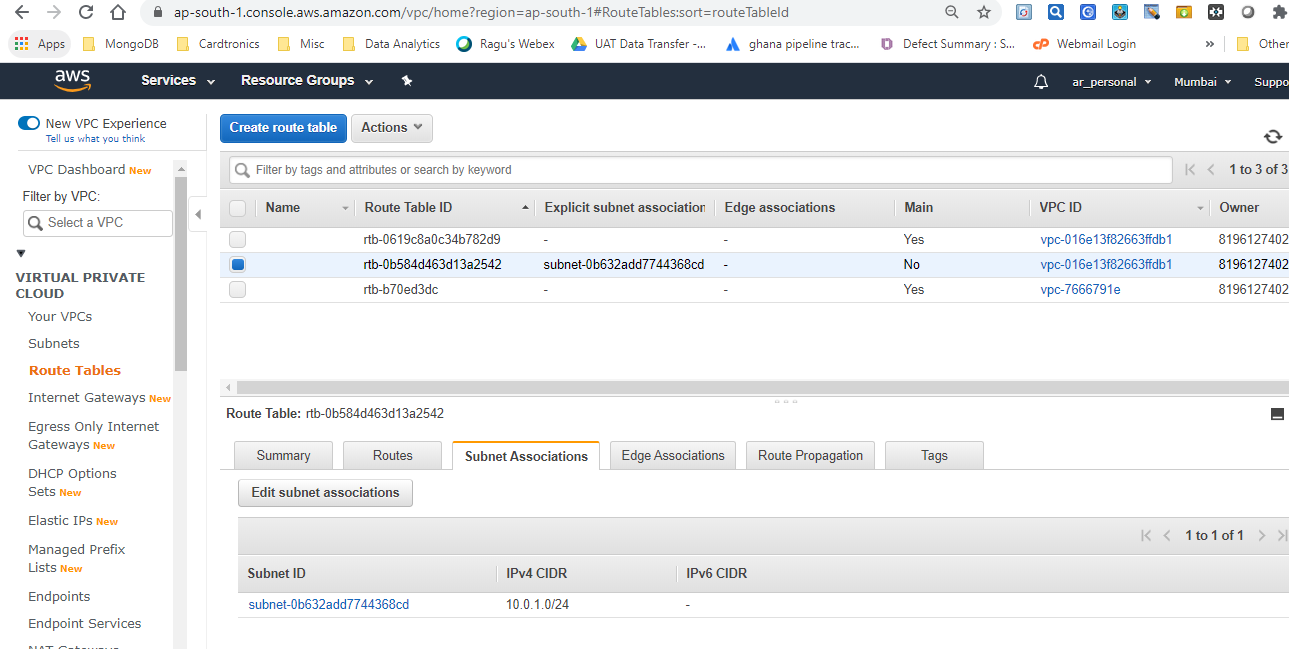
**Route Table**



**Association of Route Table with Private Subnet**



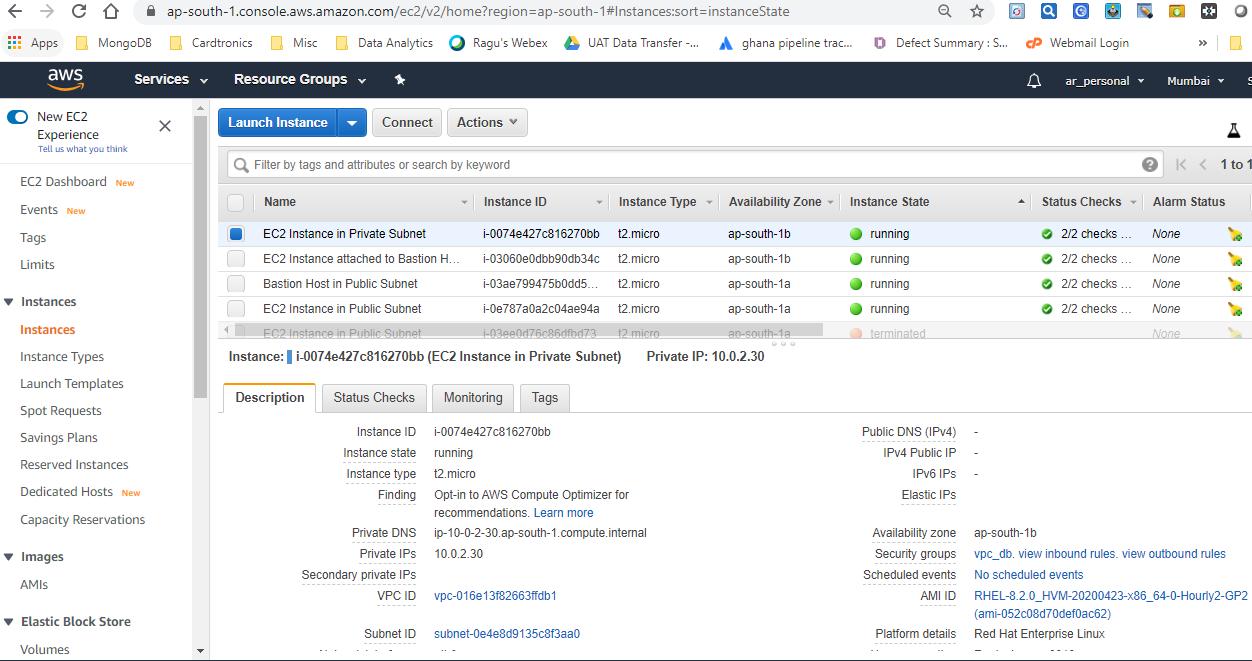
**Association of Route Table with Public Subnet**



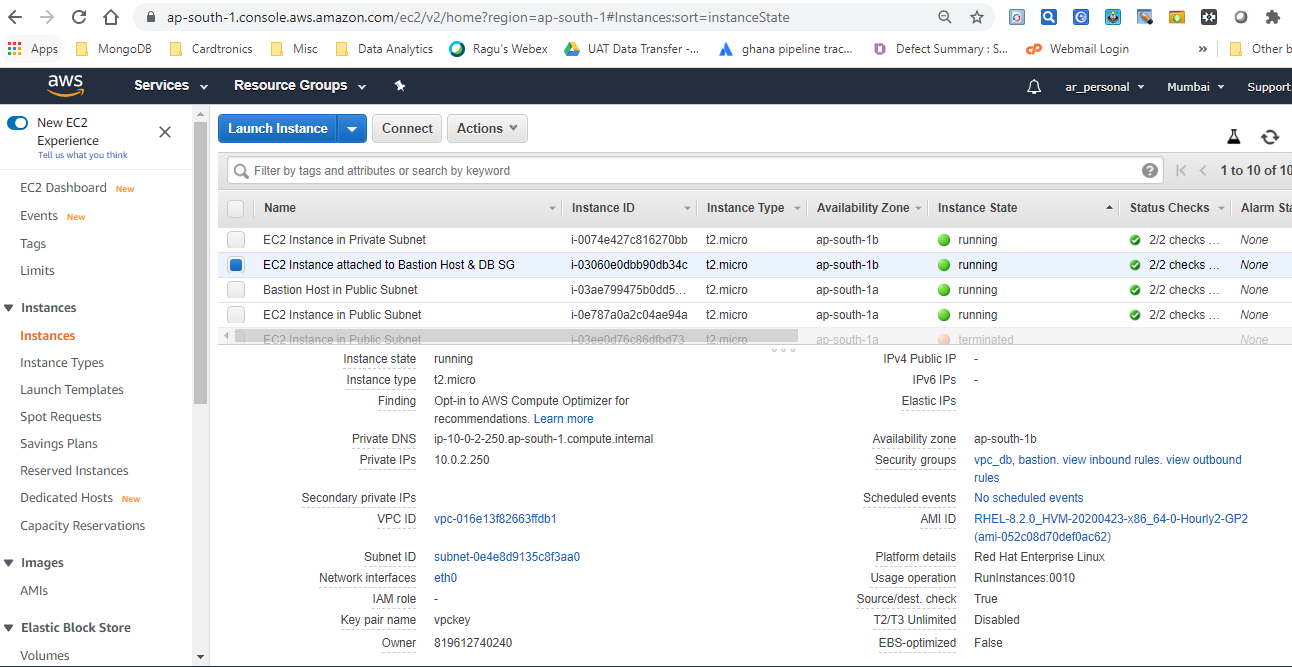
**EC2 Instance with Private IP**

Whenever instance is launched without enabling the Auto-assign public IP setting, it won't have public assigned.

We won't be able to SSH to the instance which has private IP & no key assigned to it during launch.

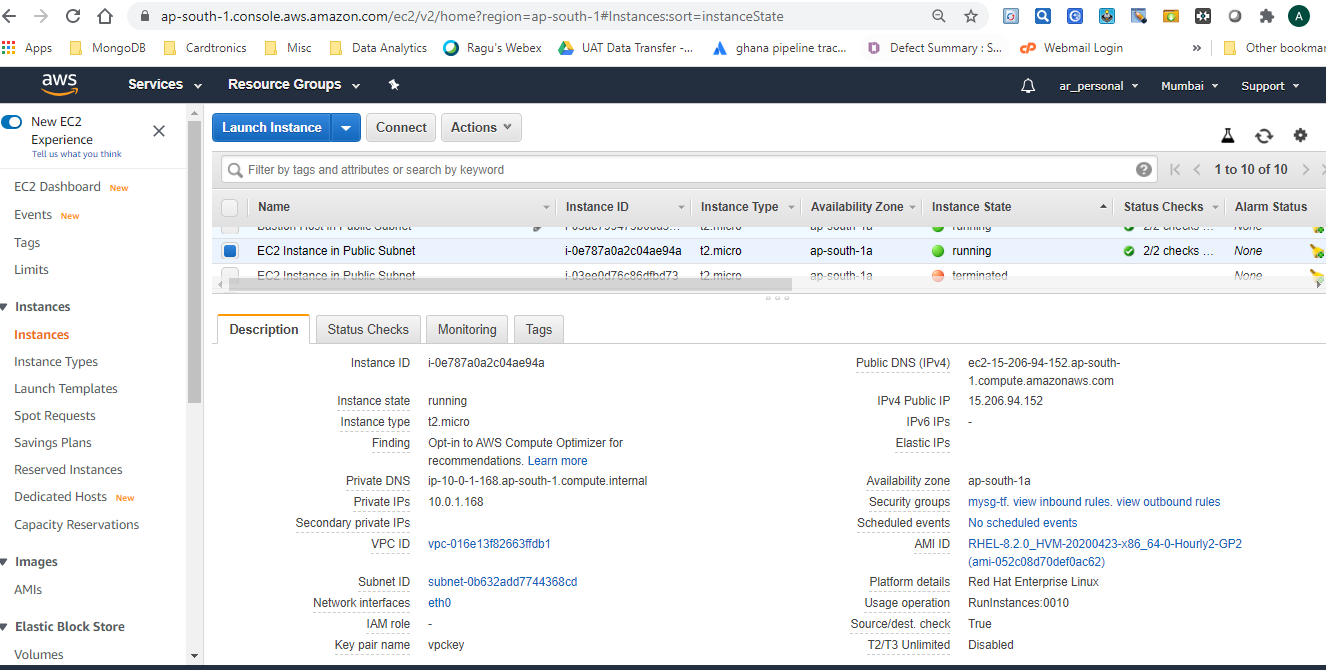


Below is DB Server instance which is attached to DB Server SG & Bastion Host. SSH is allowed from only Bastion host to this instance.

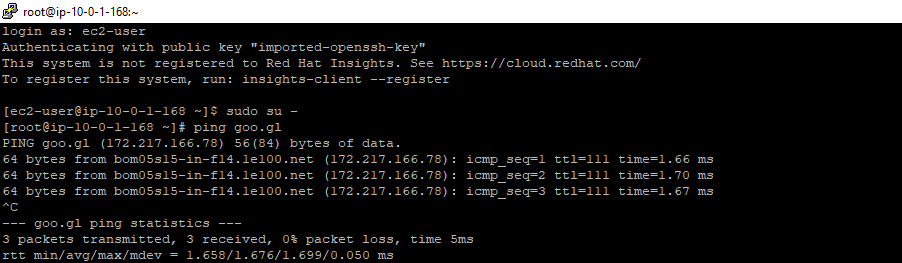


**Ec2 instance with Public IP**

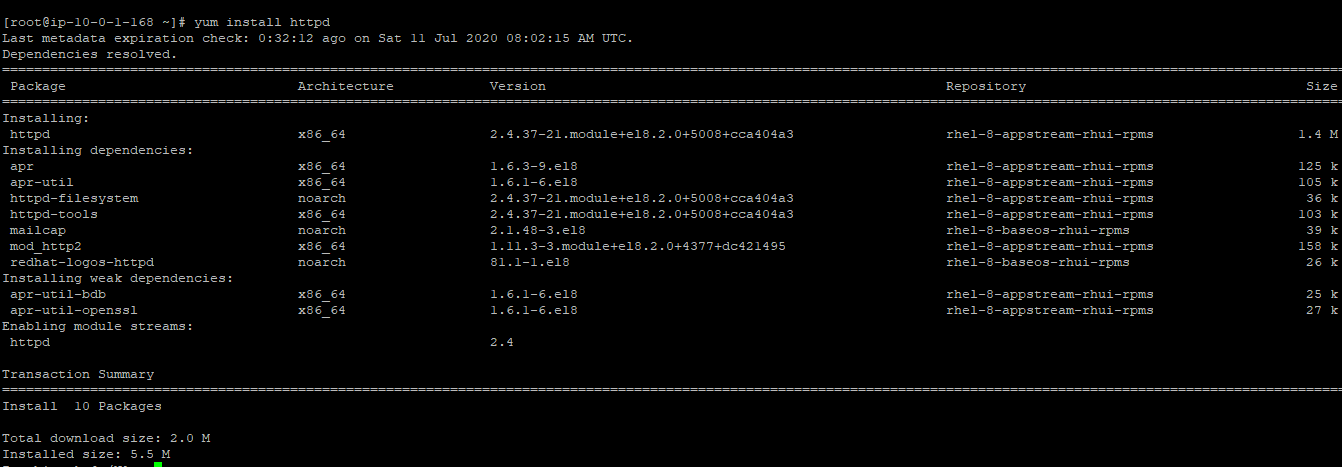
Each public instance has 2 IP's out of which Public IP is assigned to Router. As & when we launch the VM, public IP is assigned through DHCP & it's added to Router so as to enable NATTING.



**Login to EC2 instance having Public IP using Putty**

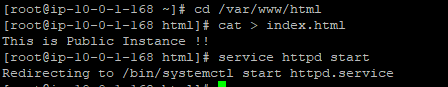


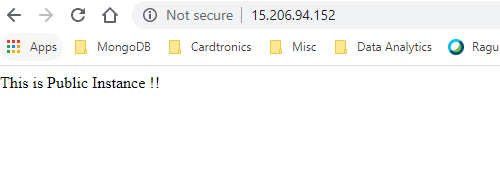
Through this instance we can install Apache web server since it has internet connectivity.





We can create our own **index.html & verify the site**



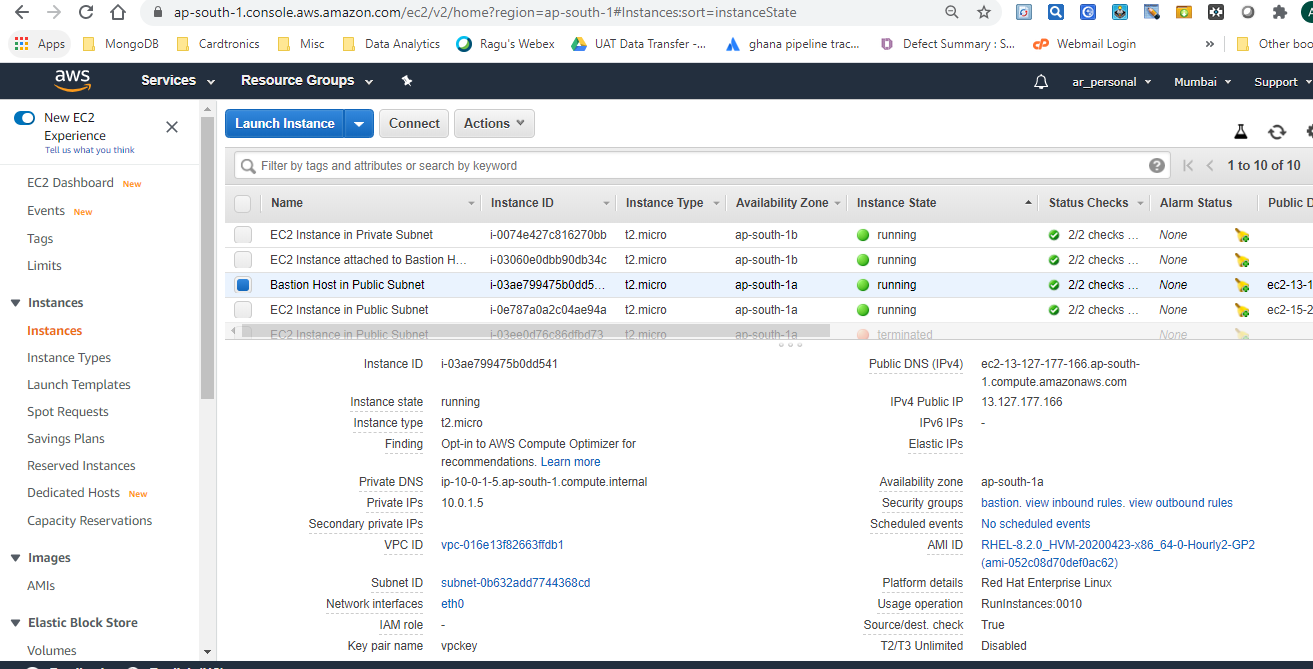


**Bastion Host in Public Subnet**

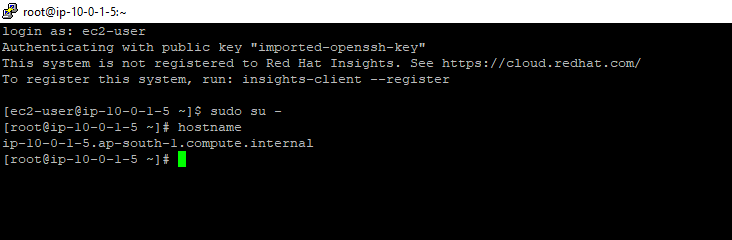
A bastion host is a server which provides access to instances in a private network from an external network, such as the Internet. In other words it allows only SSH connections from an external network to the Linux instances launched in a private subnet of Amazon Virtual Private Cloud (VPC).

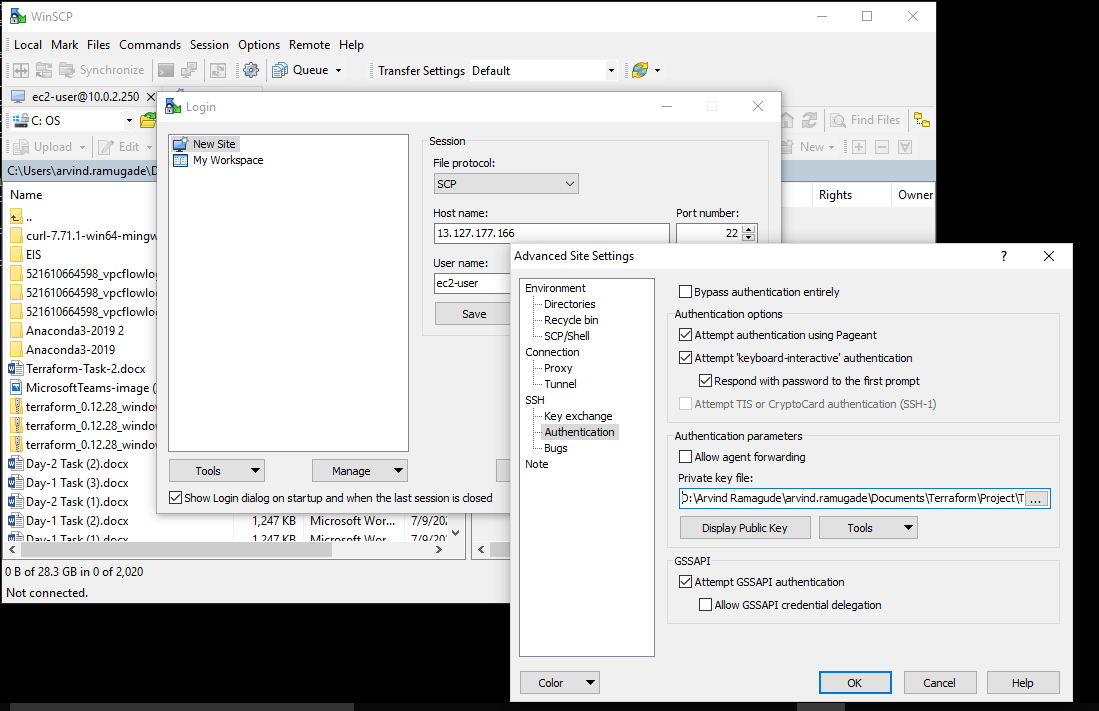
The bastion host runs in a public subnet of Amazon VPC. Blow we can see how external user uses bastion host as a jump server to connect to the Linux instances,

**Bastion Host as can be seen in AWS console**

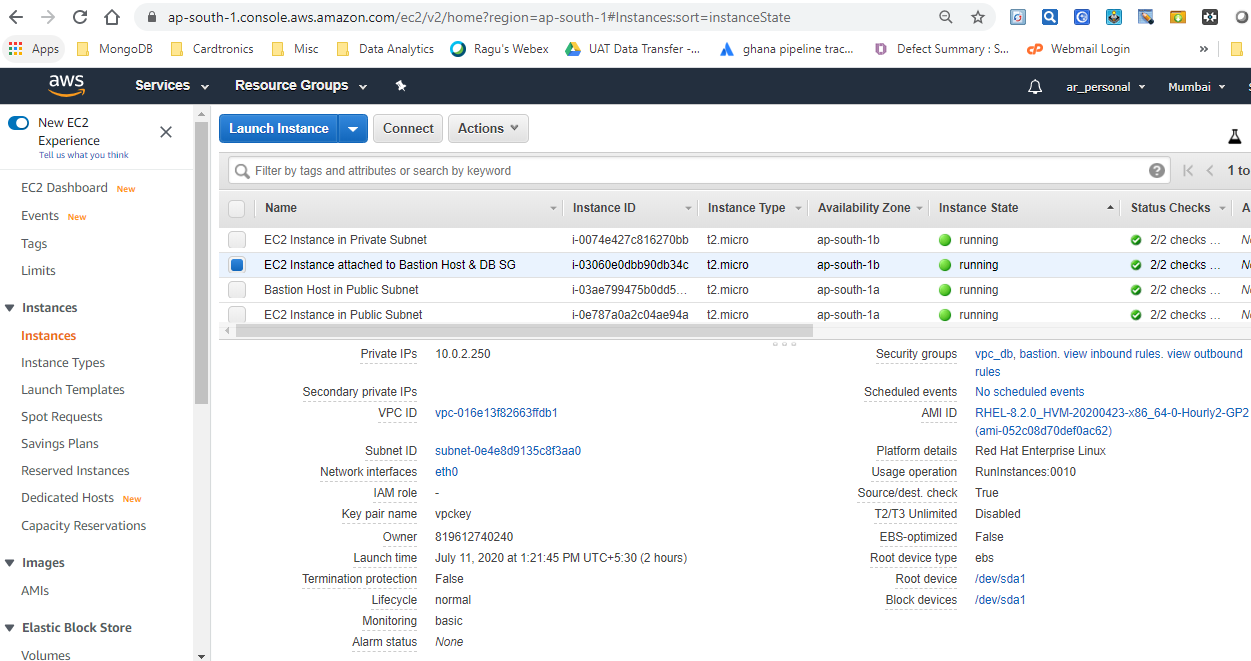


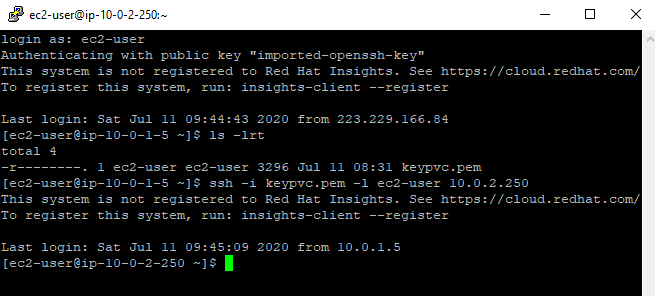
**Login to Bastion Host**



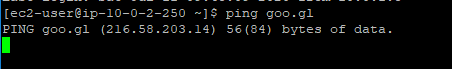


**Use Ssh from Bastion host to below private instance using pem key**

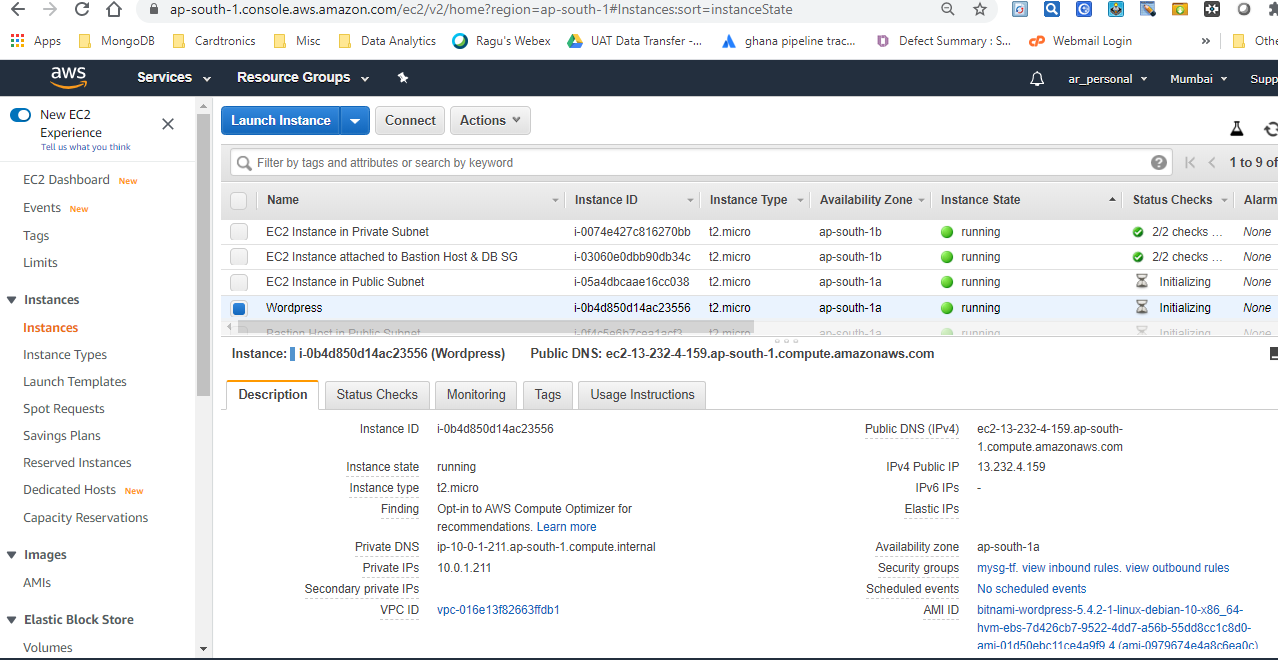




Through above **private instance we can’t connect to internet** as shown below

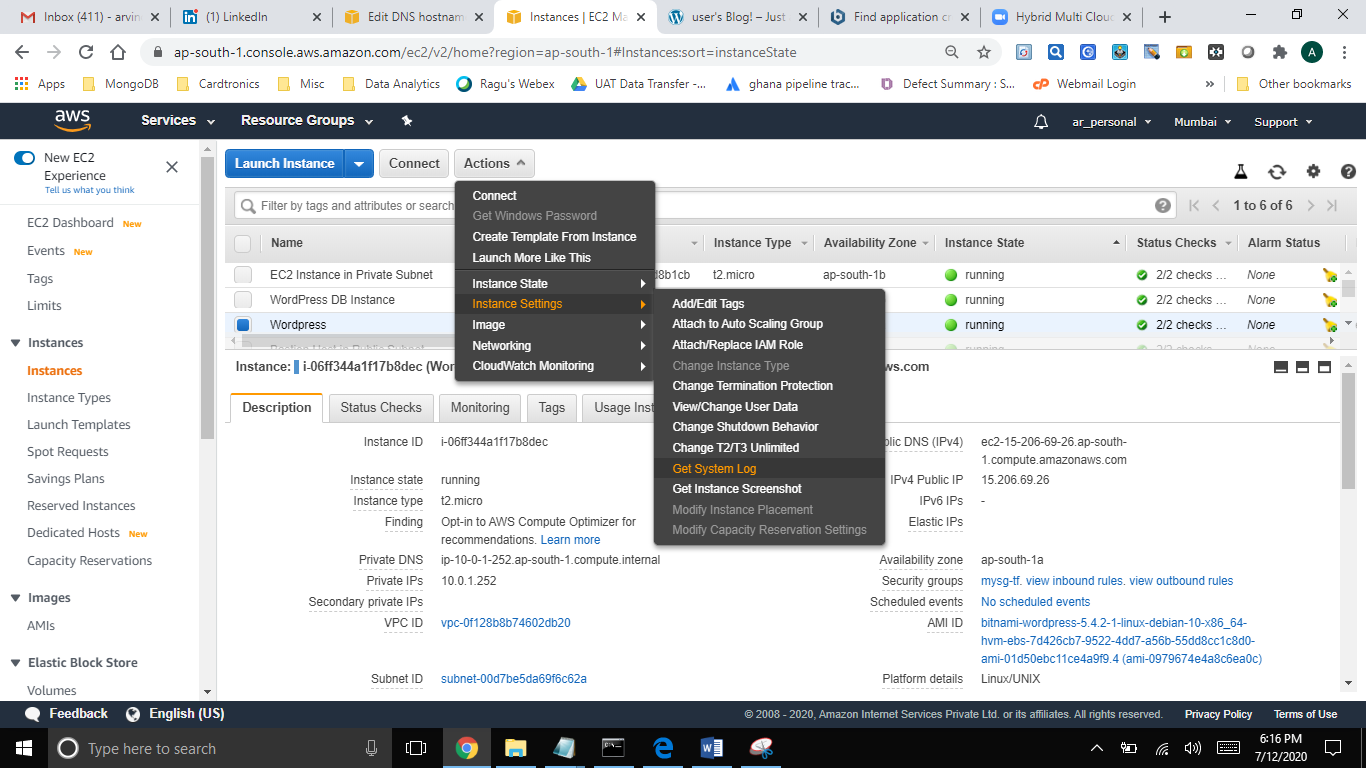


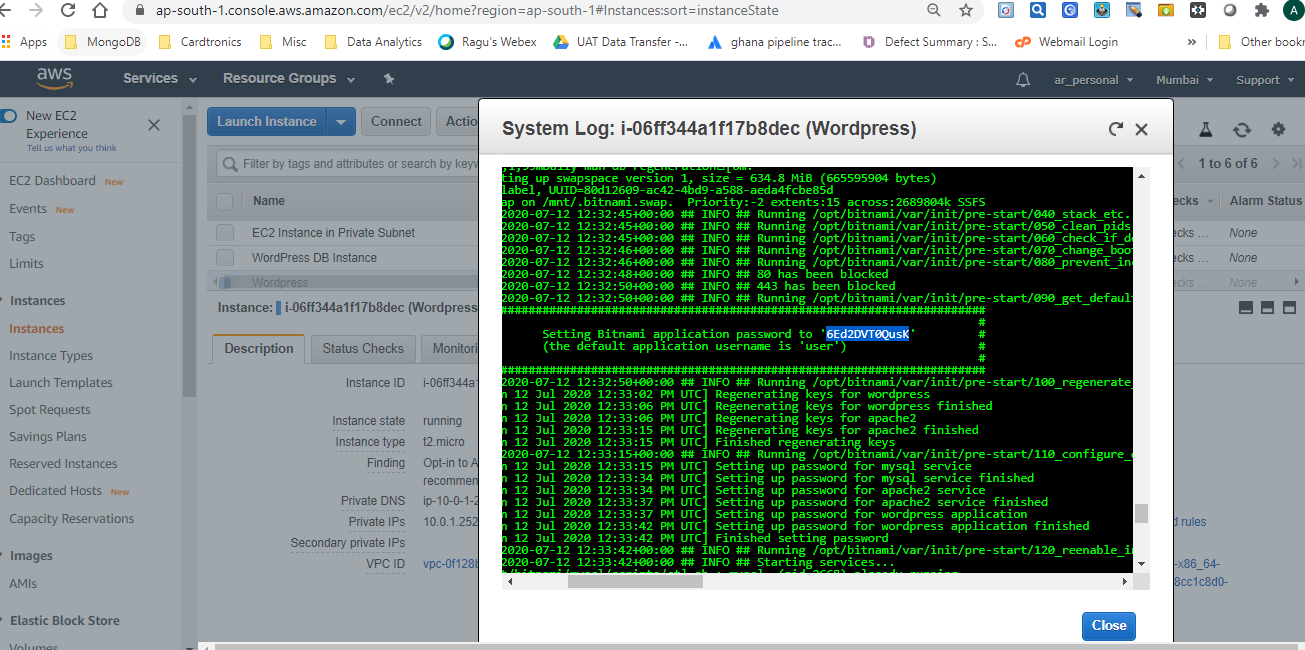
**WordPress Instance in Public Subnet with Security Group:**



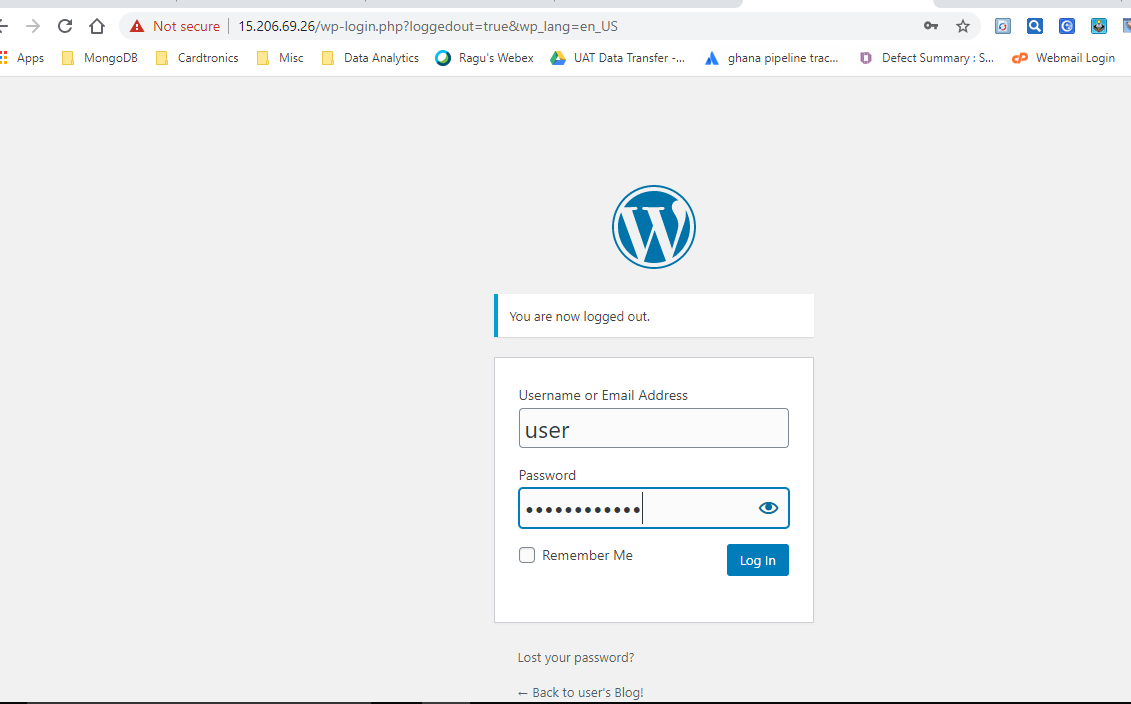
We can get password as follows

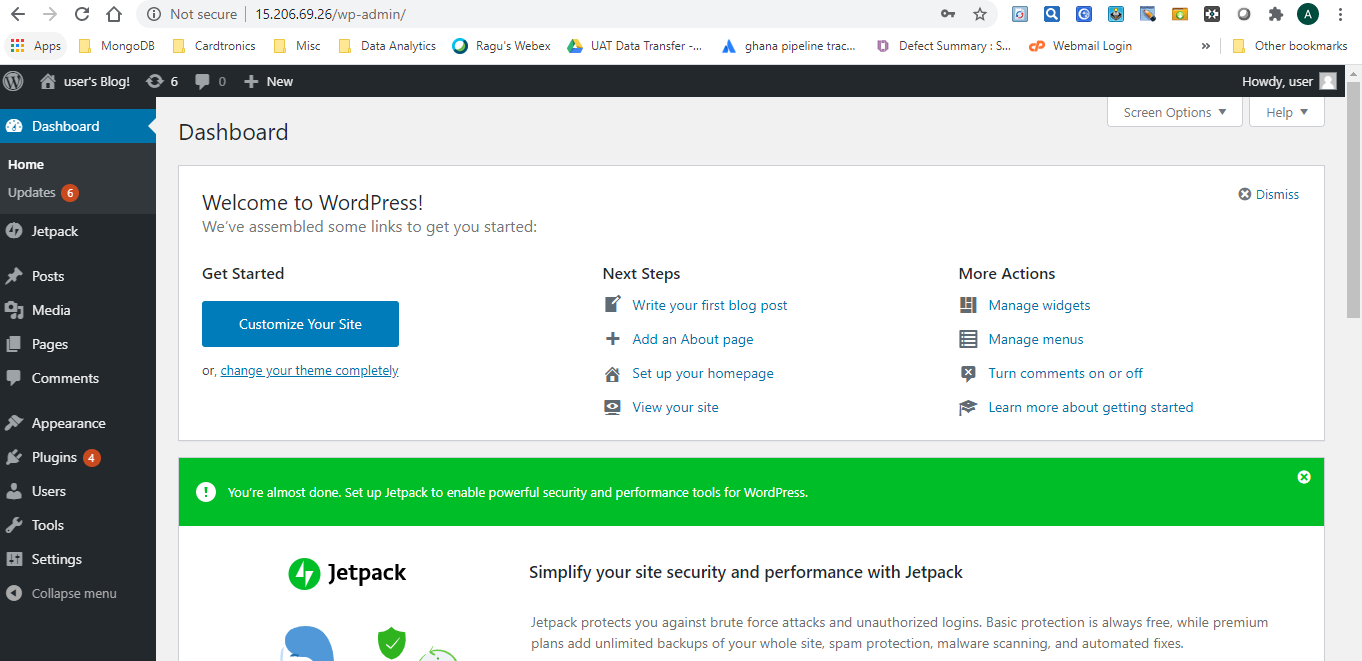
Select WordPress instances & select following option

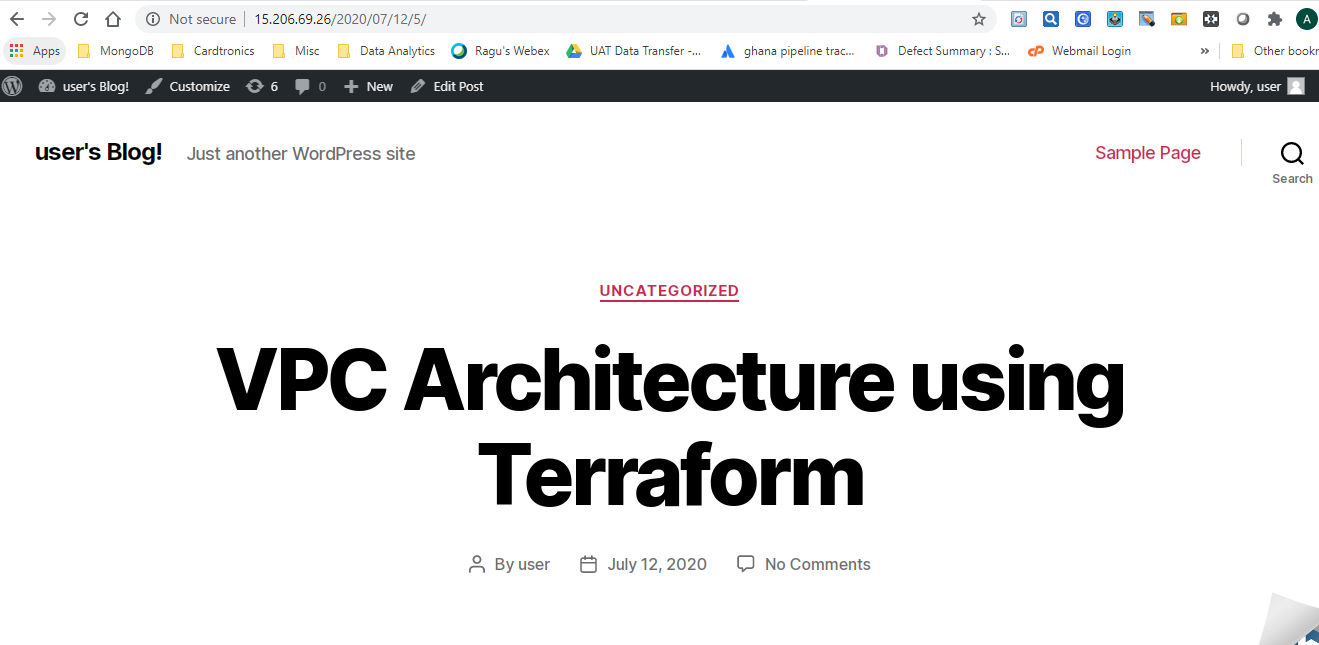


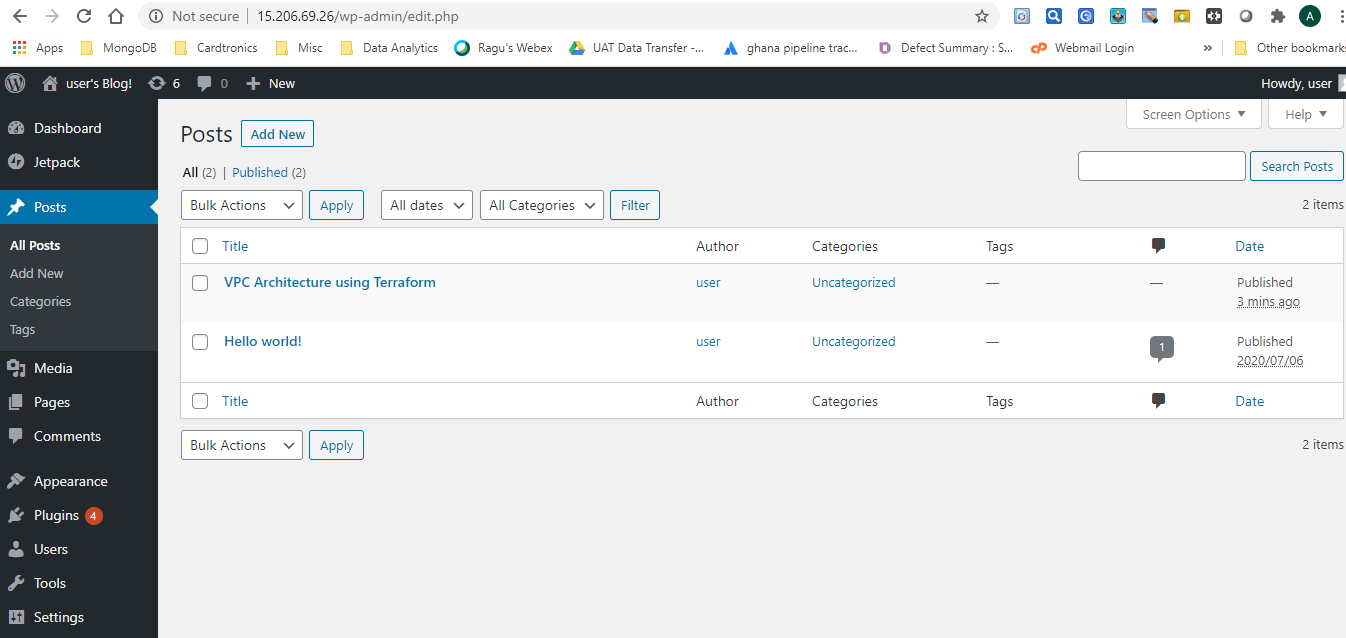


**Login to WordPress as under**

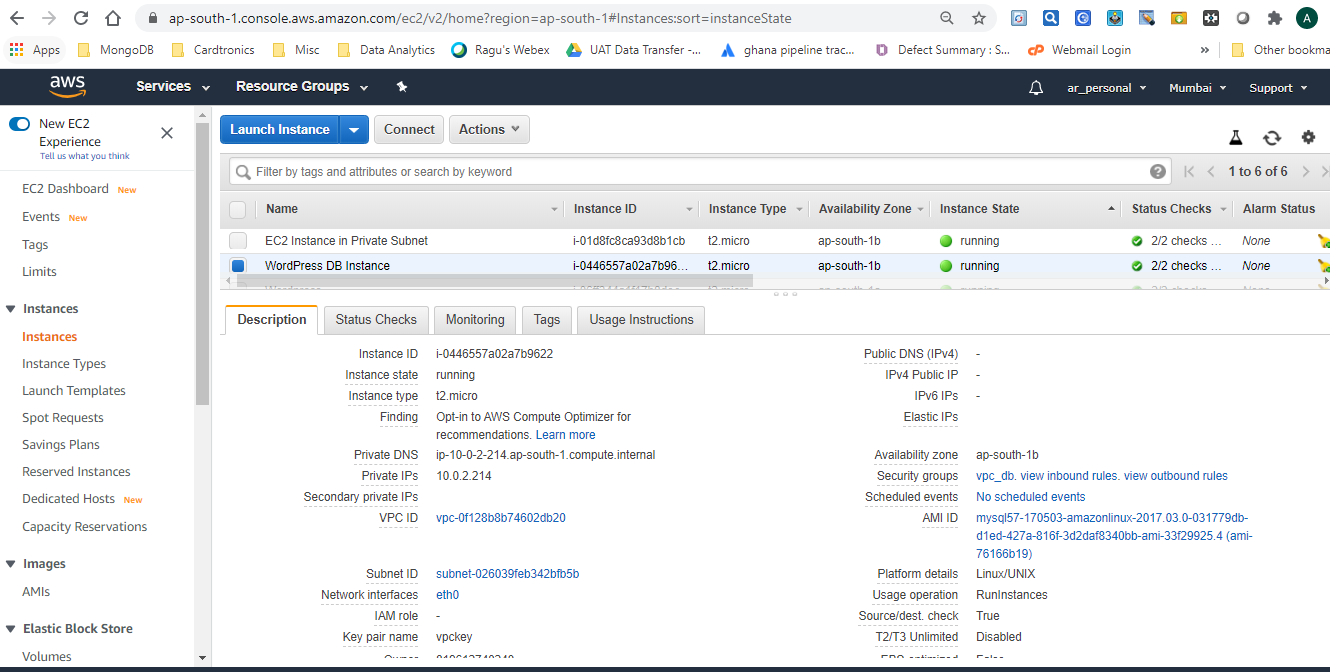








**MYSQL Db Instance is as under**



**NAT**

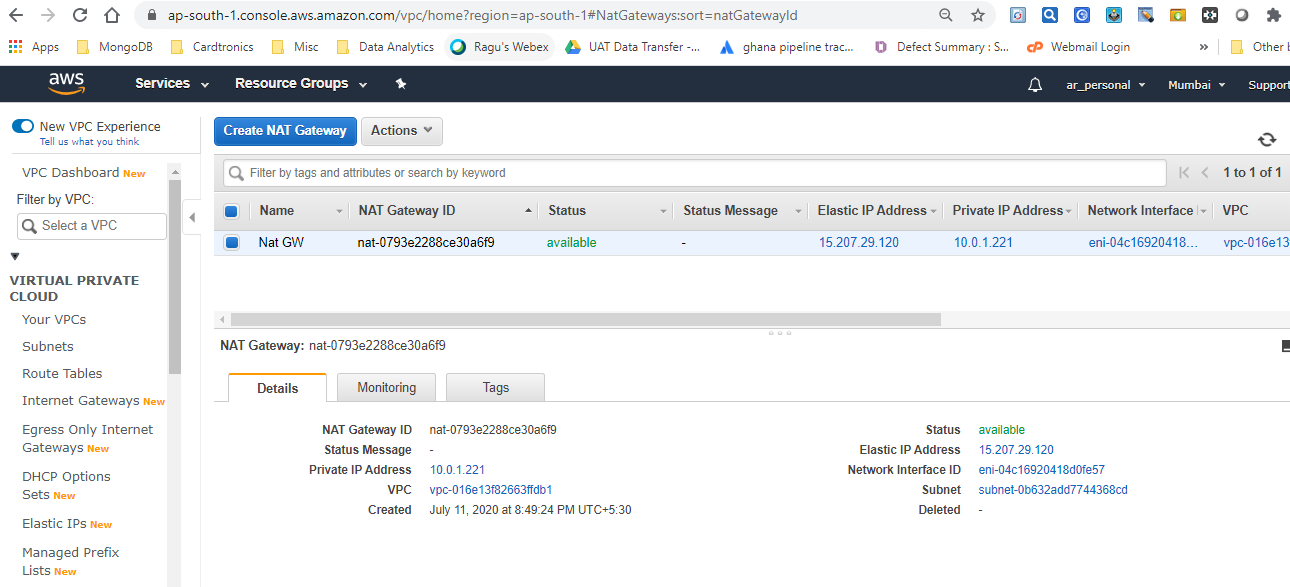
When an instance in private subnet requests information from internet, NAT device keeps track of its private IP address & makes a request using its own public IP giving the impression that it is the NAT who is requesting the information. Once the reply comes from internet NAT device checks which device has made the request & returns the information to it. Thus the private IP of the instance never gets exposed to internet. This is called network address translation. (NAT)

**NAT Gateway:**

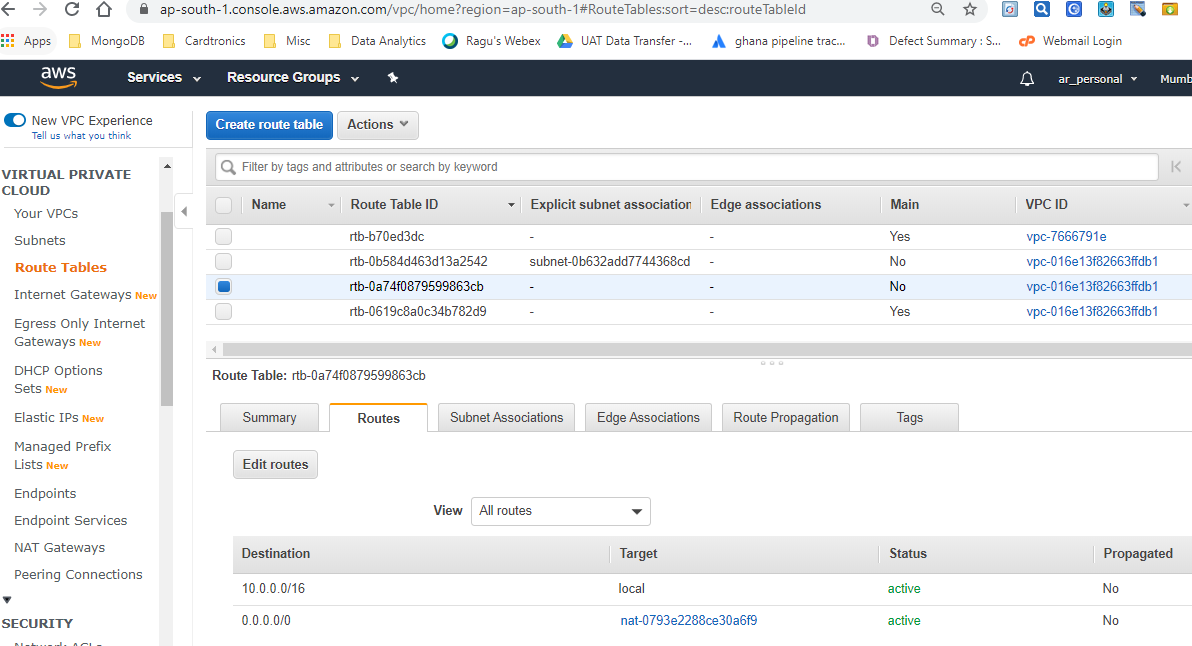
It’s a NAT device managed by AWS. It's highly available & scalable, however not free. NAT allows SNATTING.  It prevents the internet from initiating a connection with those instances.

NAT gateway is attached to subnet whereas internet gateway works at the VPC level. IGW allows both inbound & outbound traffic whereas NAT gateway only allows outbound internet connectivity. NAT gateway is chargeable however it provides bandwidth up-to 45 GB

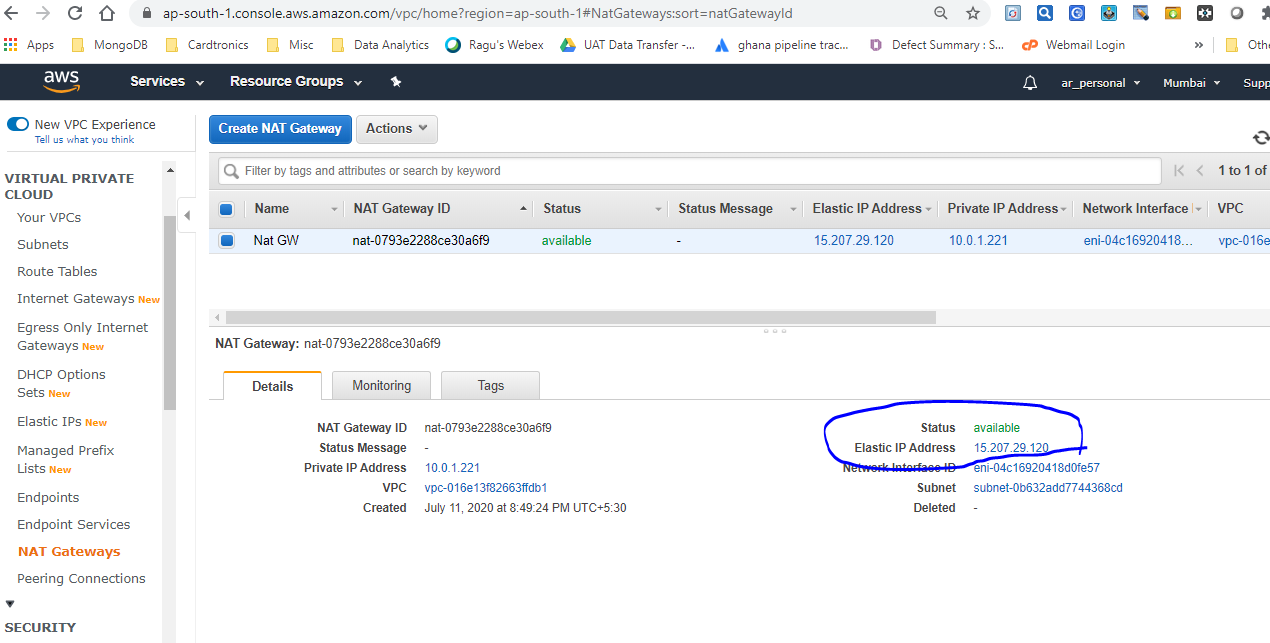
We can verify **NAT Gateway** & its **association with Public Subnet** in AWS Console as under



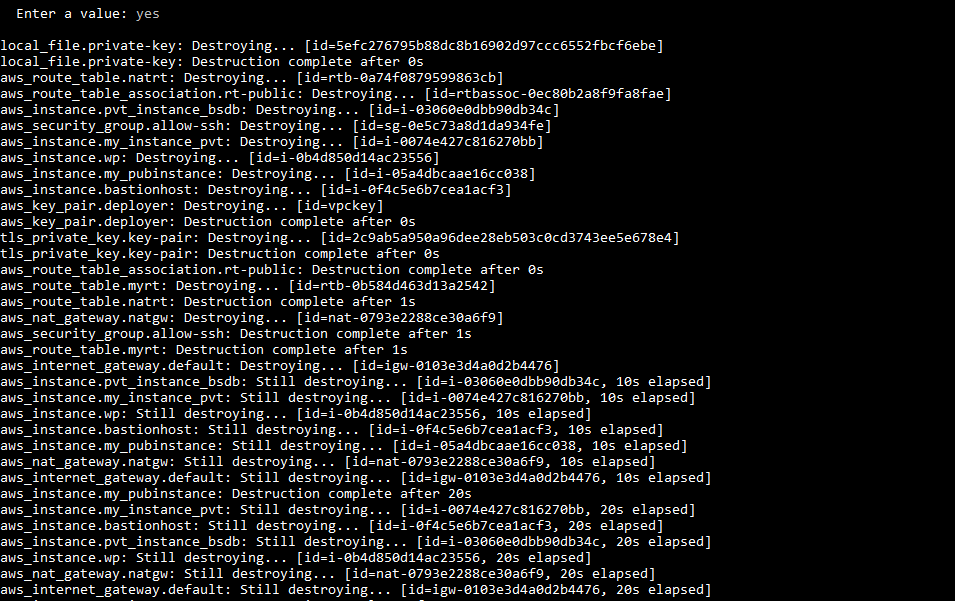
We can also check **verify route table entry as follows**

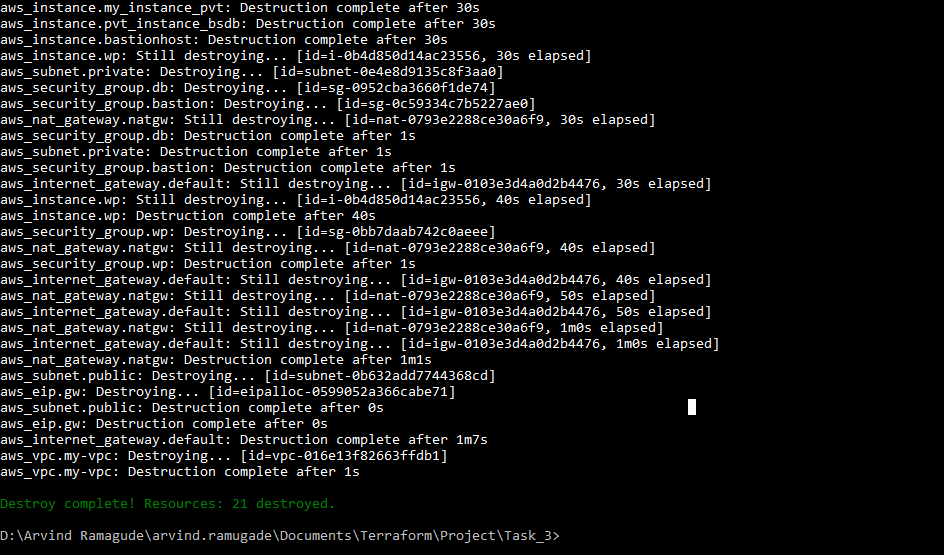


We can also notice that **Elastic IP gets assigned to NAT GW**

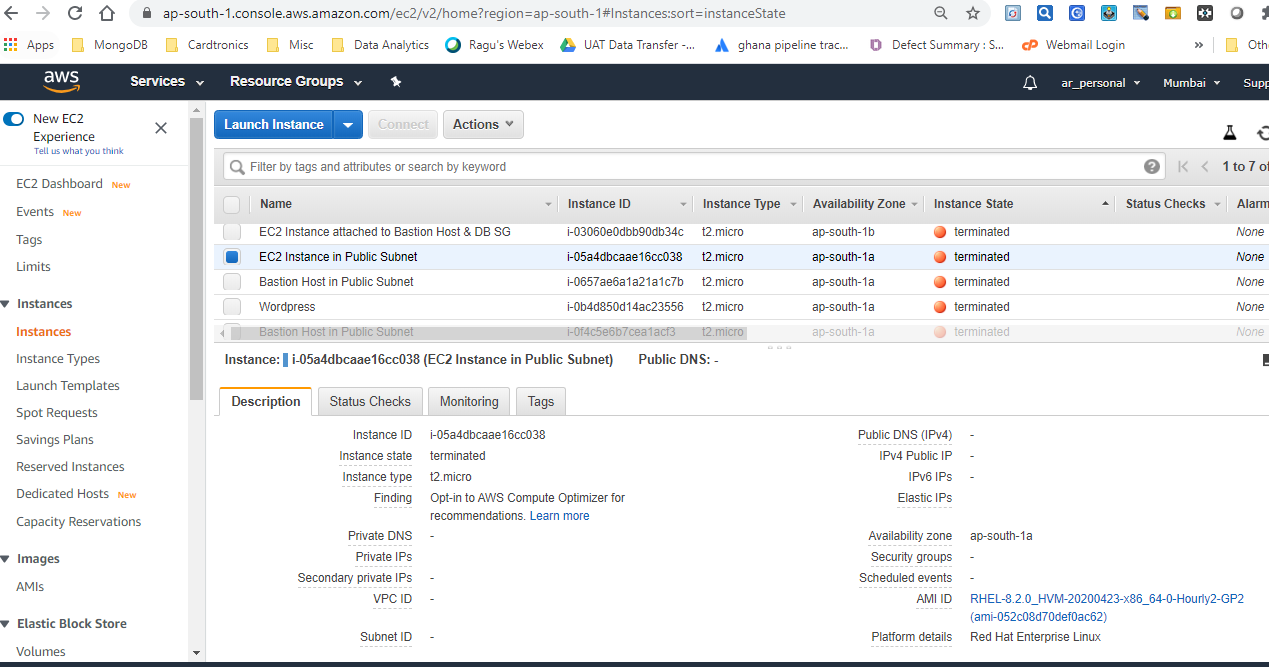


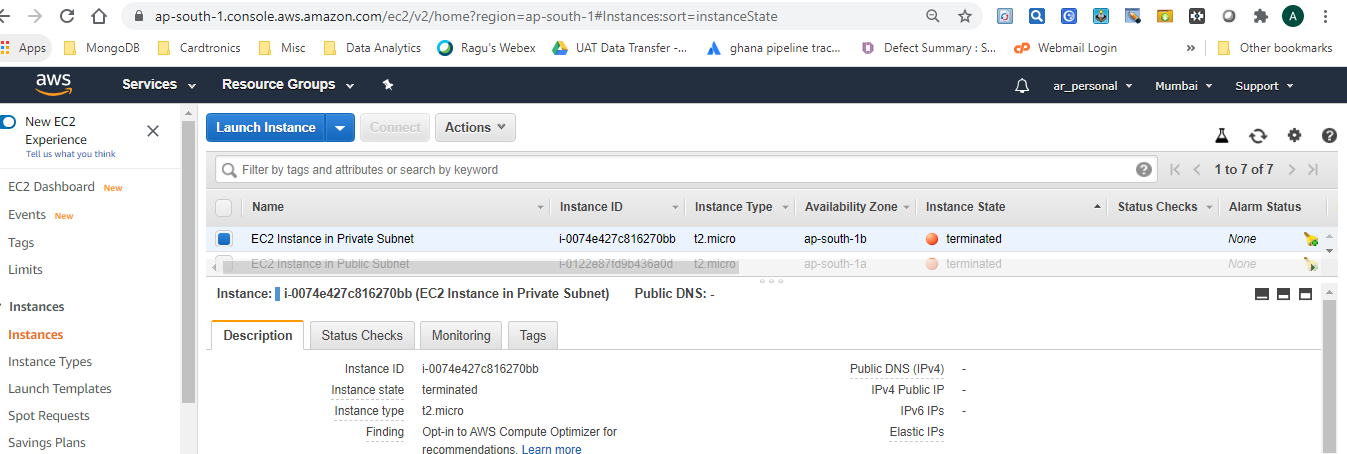
Entire infrastructure can be destroyed with one single command



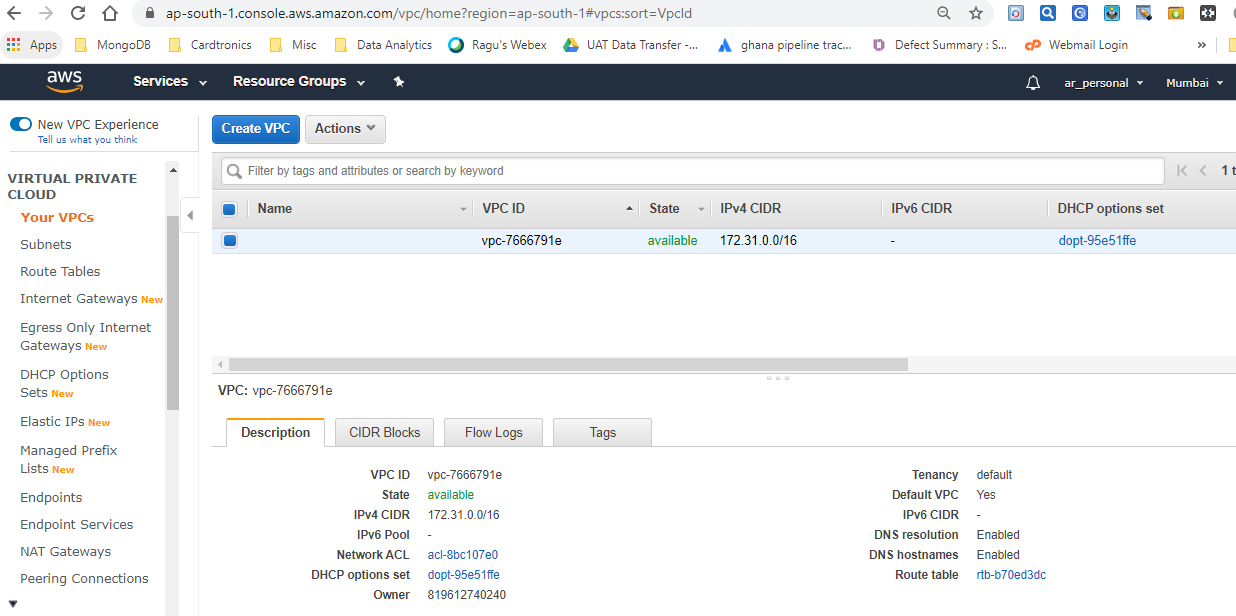


Verify EC2 instance status through AWS Console

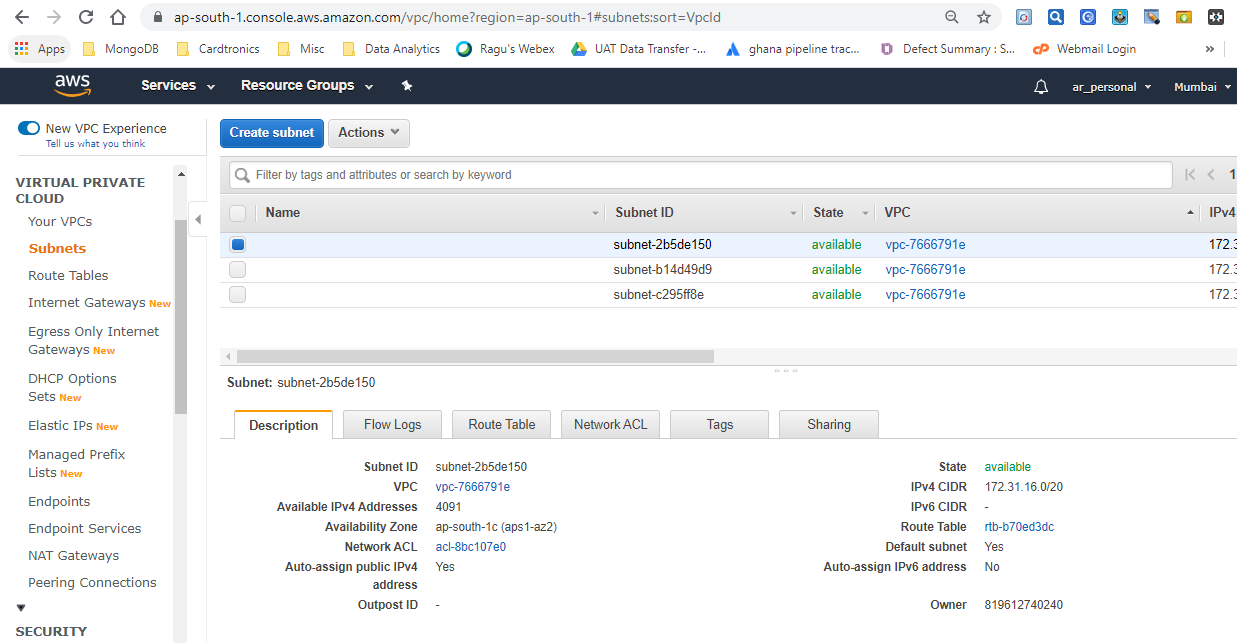




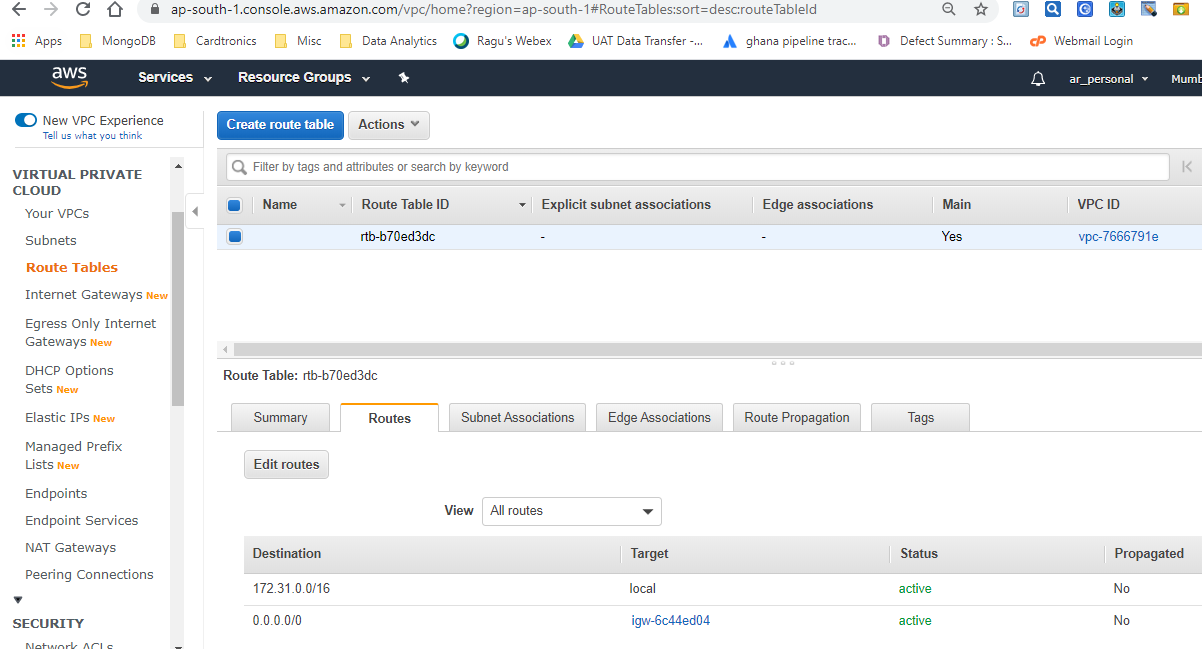
**Default VPC remains**, however the **VPC created by Terraform gets deleted.** (Same for subnets as well)



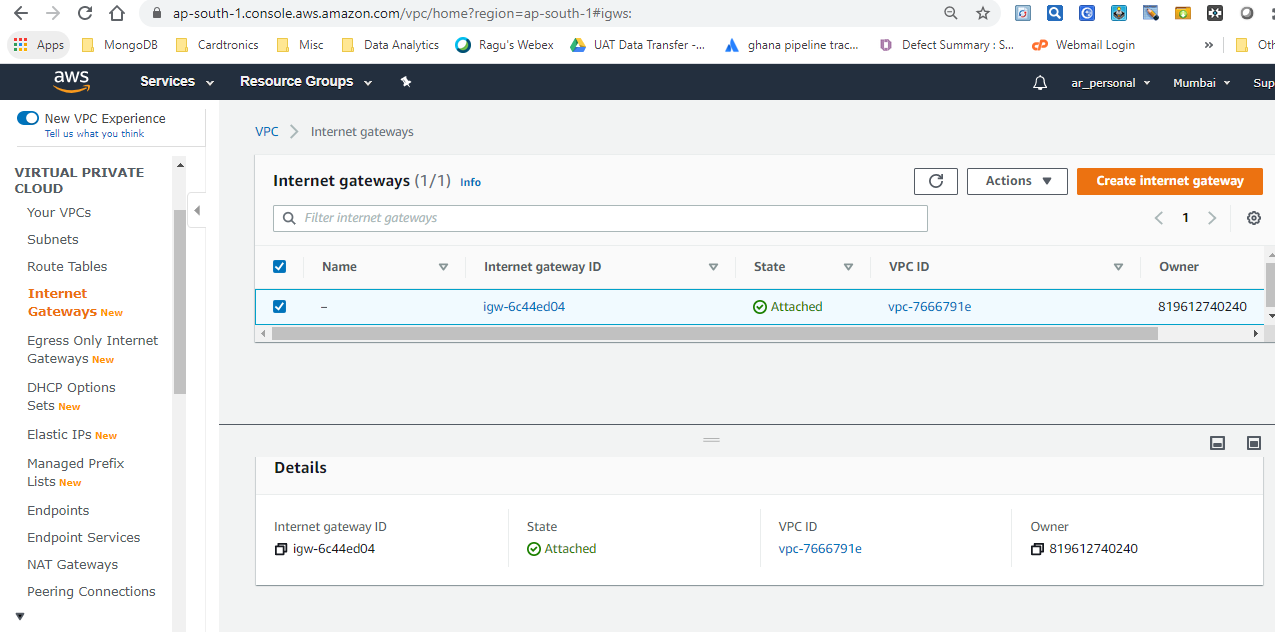
**Sunsets –Subnets associated with default VPC remains.**



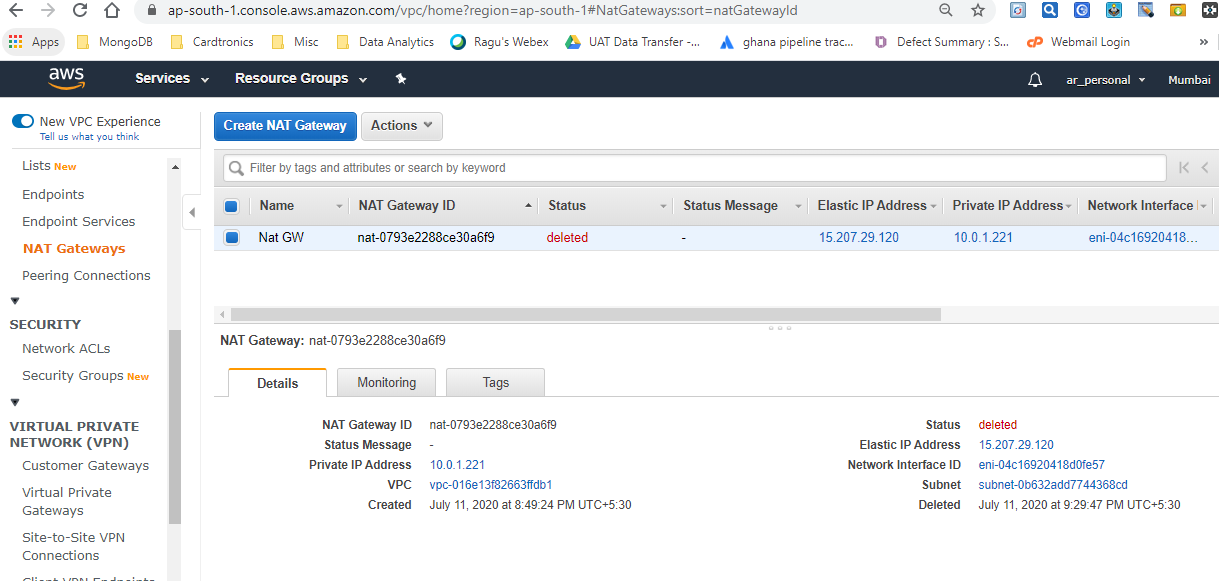
**Default Route Table remains**, however **Route Table created through code gets removed.**

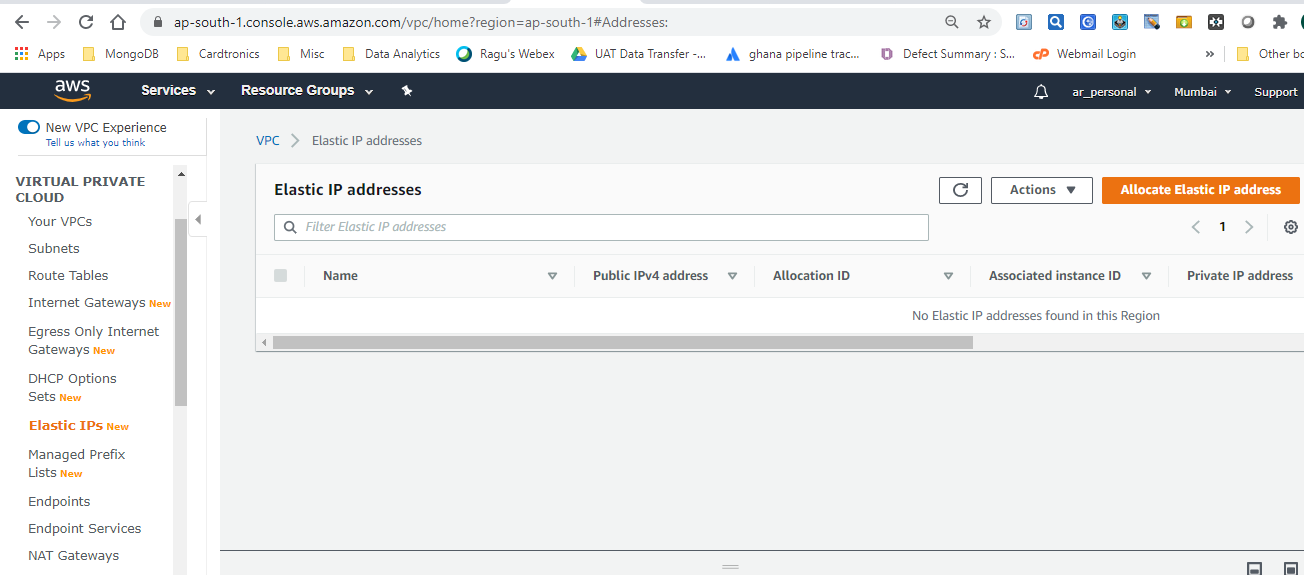


**Default IGW remains**



**Nat Gateway also gets deleted & EIP gets released**





This concludes that we can launch as well as destroy entire set up using Terraform code without having to bother about manually starting services/configuring other services on AWS.