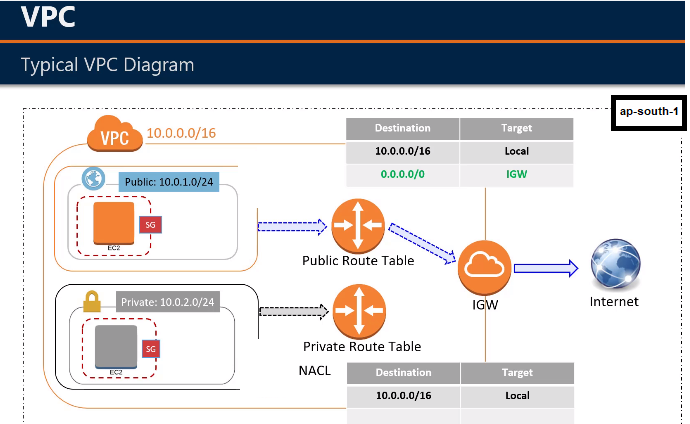
***Set up VPC with Public & Private subnets using Terraform***

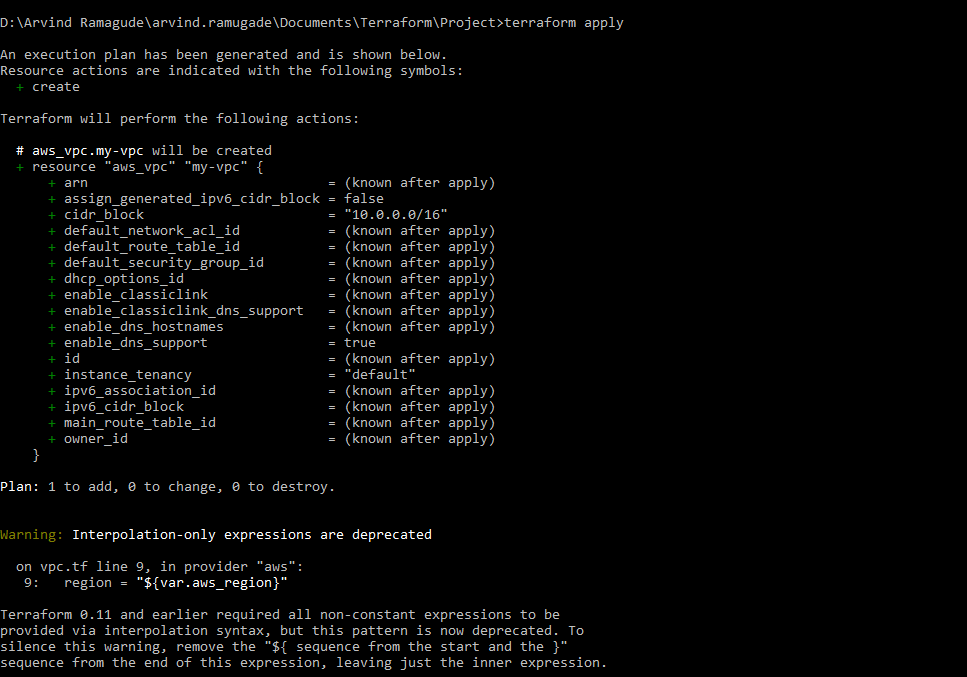
VPC - It's a virtual private cloud which is logically isolated from other networks.

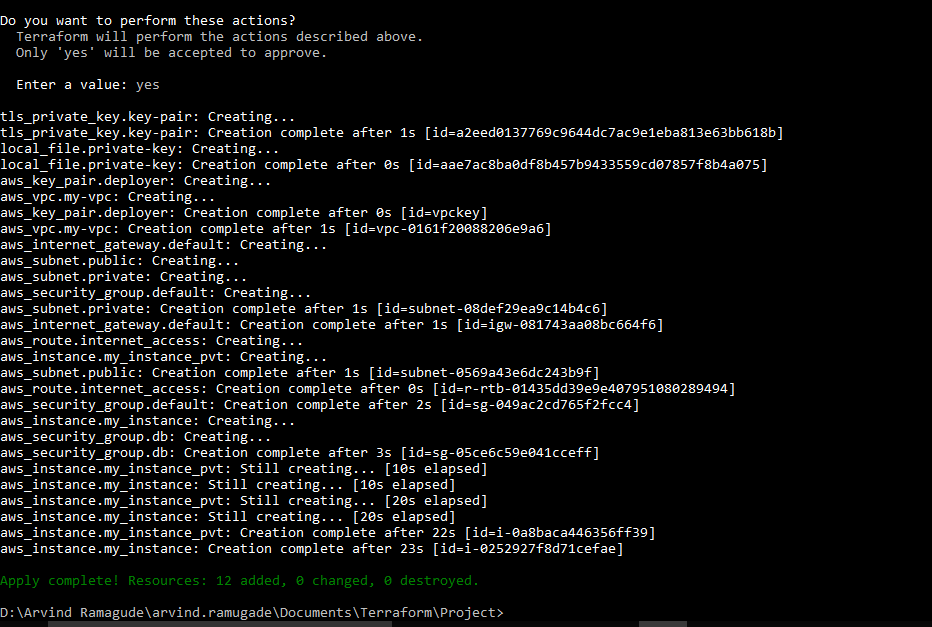
It's secure & by default AWS creates VPC with 3 subnets.

VPC provides Infrastructure as a Service (IAAS) as well as Network as a Service (NAAS)

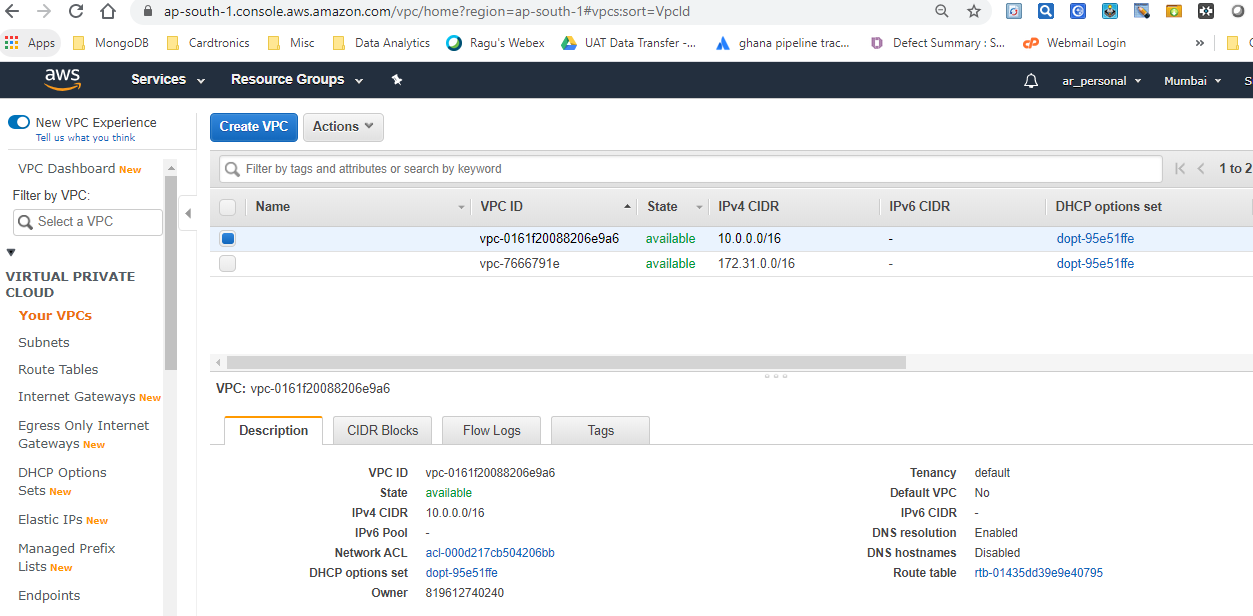
We can launch below infrastructure using terraform code:







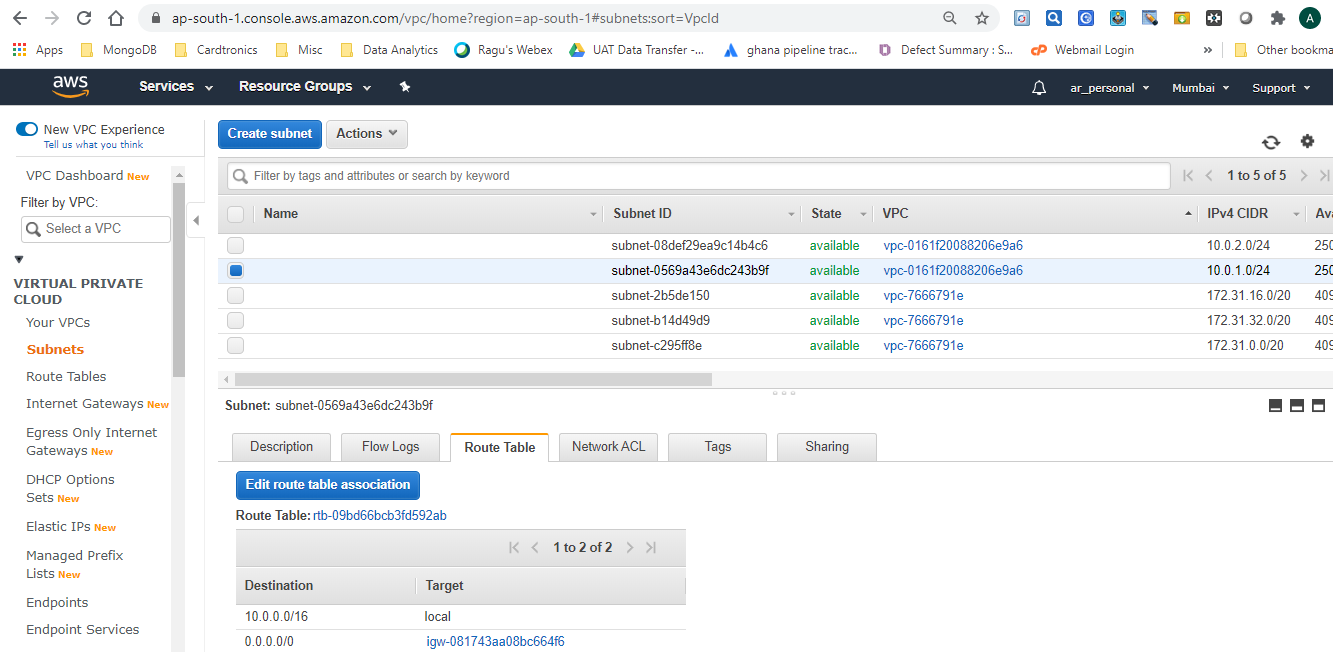
We can verify the creation of VPC through AWS console as follows :



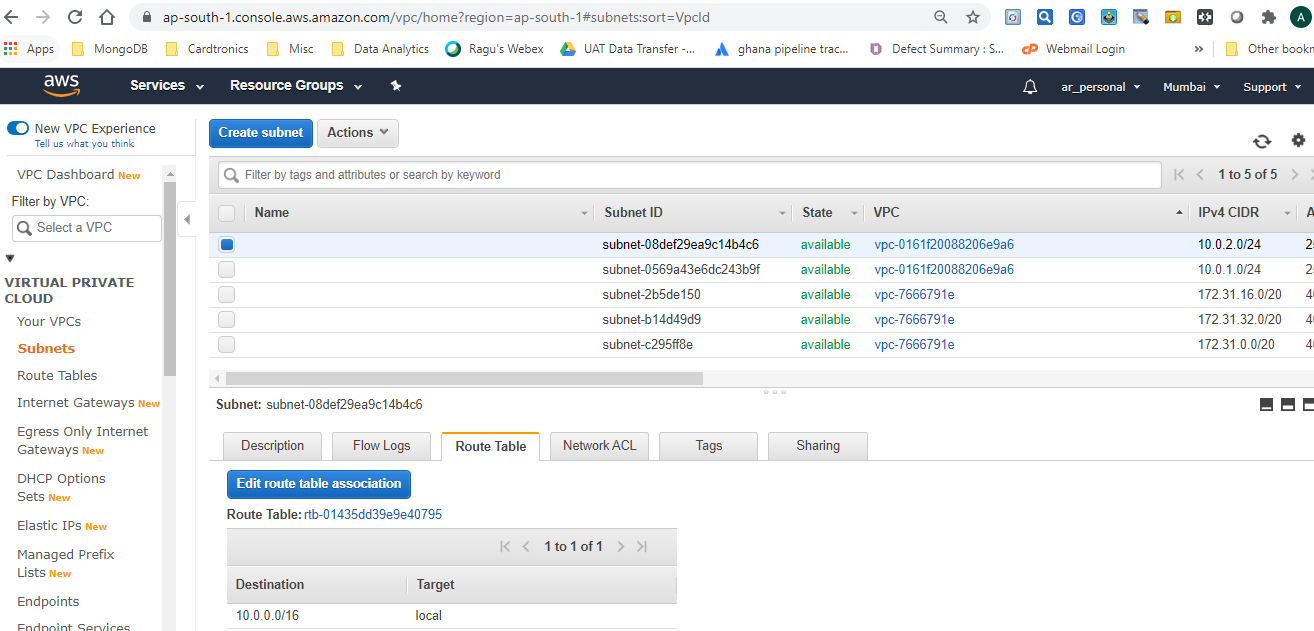
Subnets is logical subdivision of VPC. By default instances are launched in any one of the 3 subnets unless we specify the subnet while launching it. VM should be launched inside subnet. It's not possible to launch VM/EC2 instance without subnet.

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.

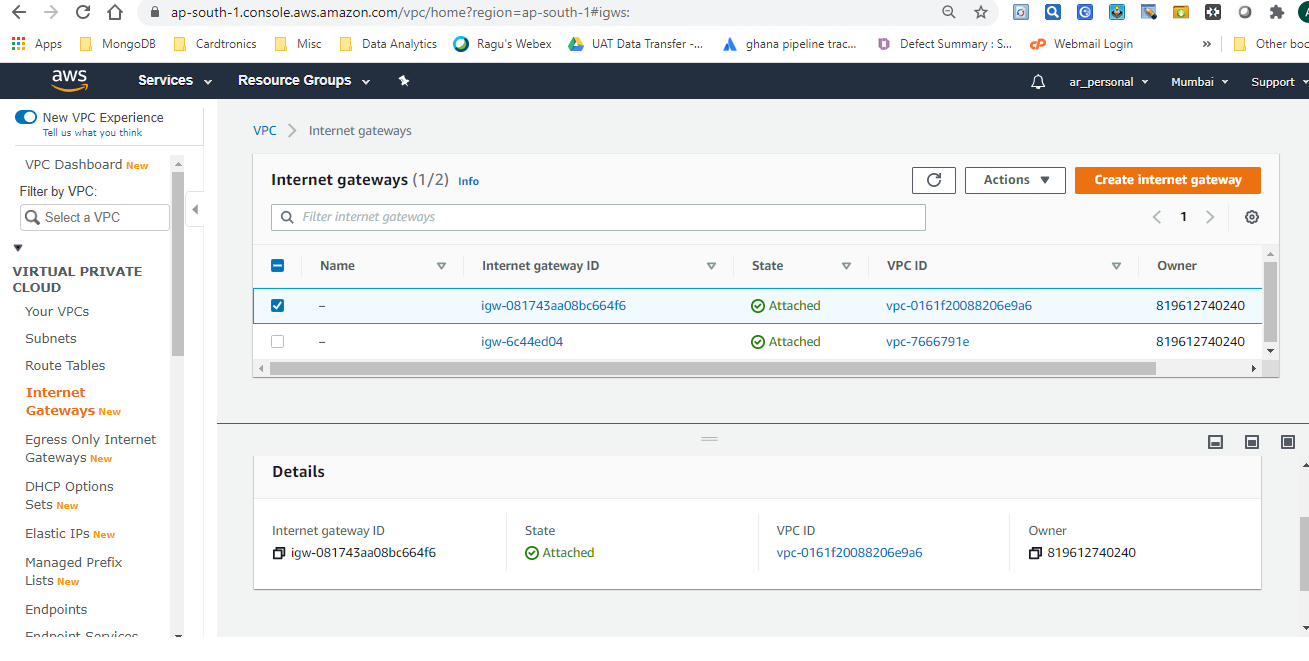
e.g. 10.0.0.0/24 - ideally it should have 2 raise to 8 (32-24=8) i.e. 256 IP addresses. However, since AWS reserves 5 IP's for each subnet it will have only 251 IP addresses for use.

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

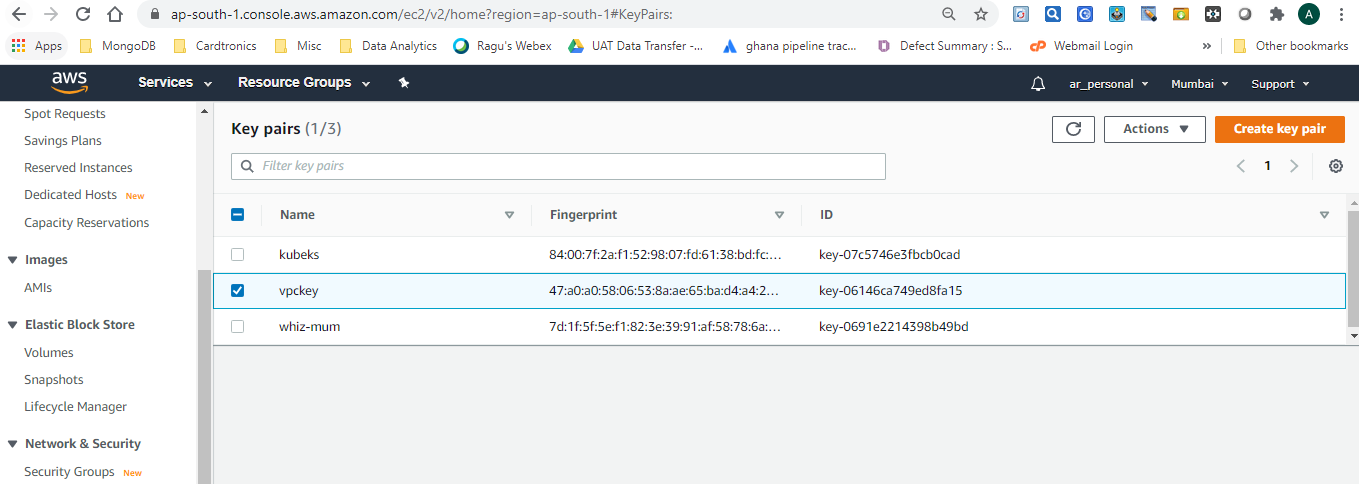
Inside subnet there are switches & routers. All instances (VM) can communicate using private IP across different subnets.

Internet Gateway

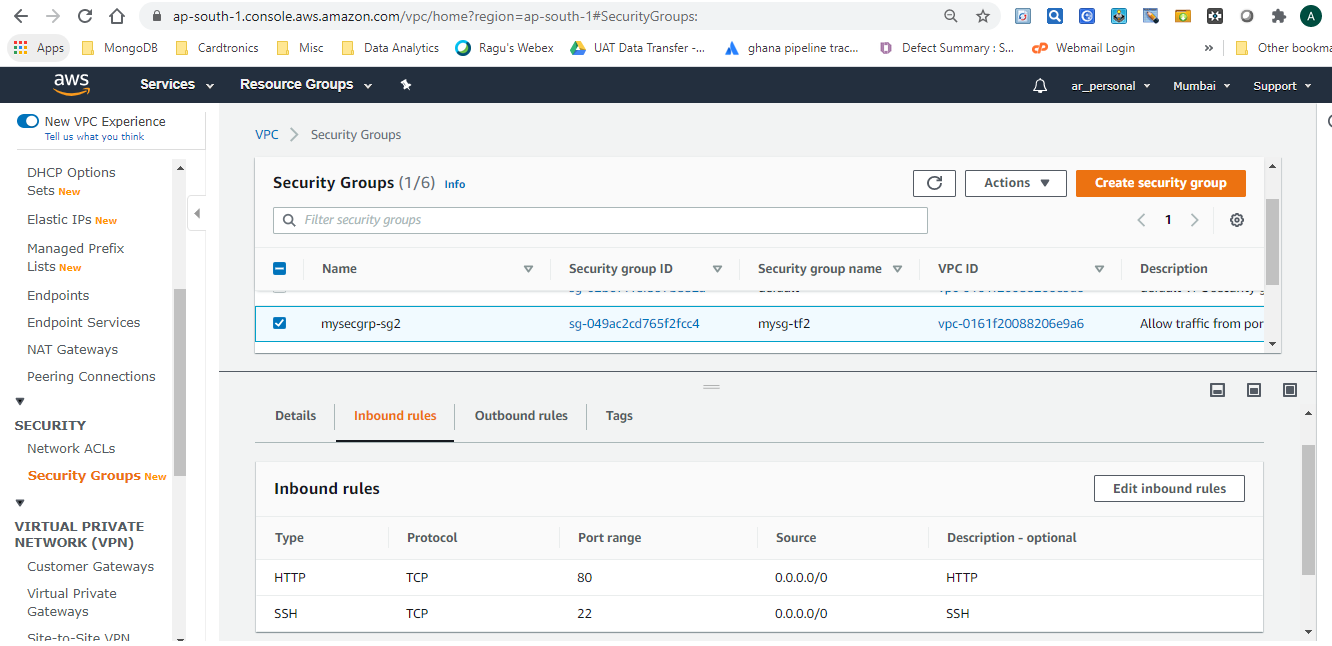
In AWS world, the public IP address of EC2 instance is actually the IP address of Router which is used to communicate outside the subnet. Router which is having public facing IP (0.0.0.0) is known as Internet Gateway (IGW). It uses both SNAT & DNAT terminologies. IGW is attached to VPC.



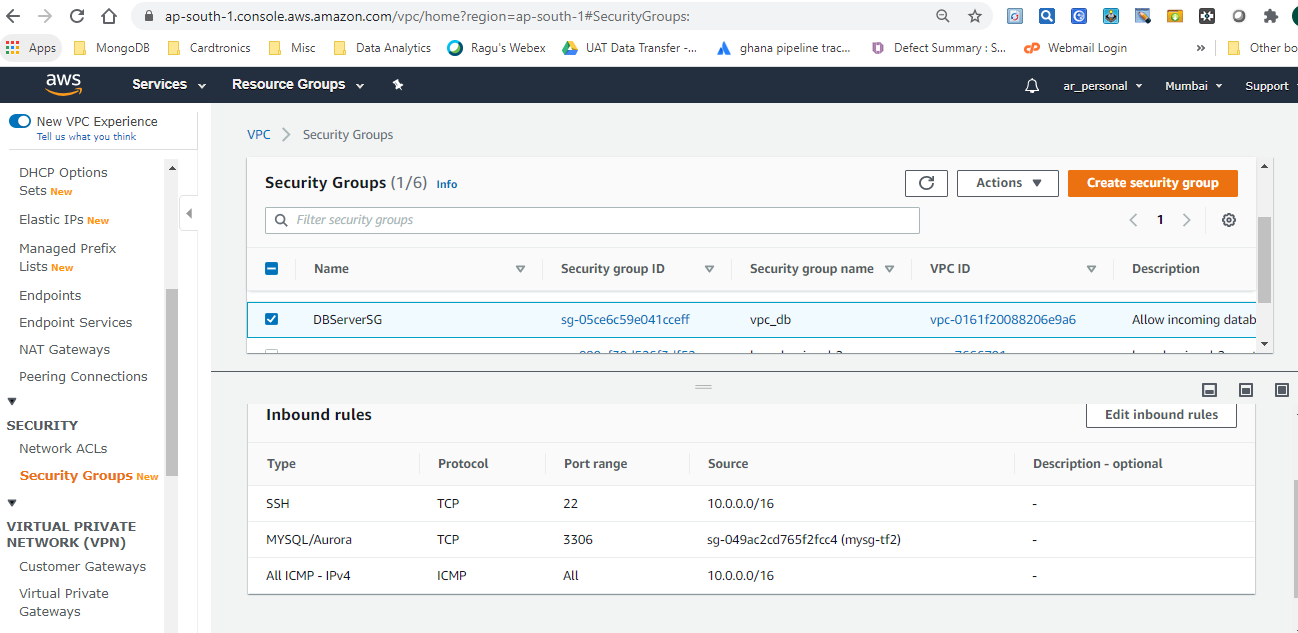
Key-pair creation



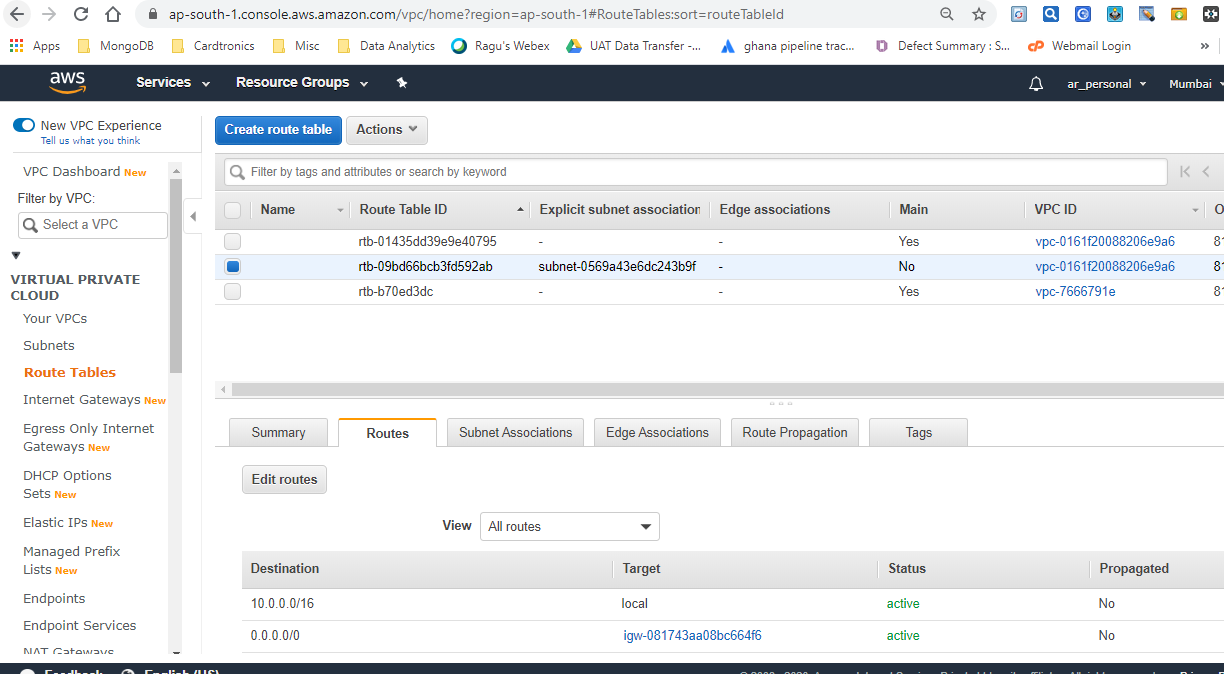
Security Group for WordPress



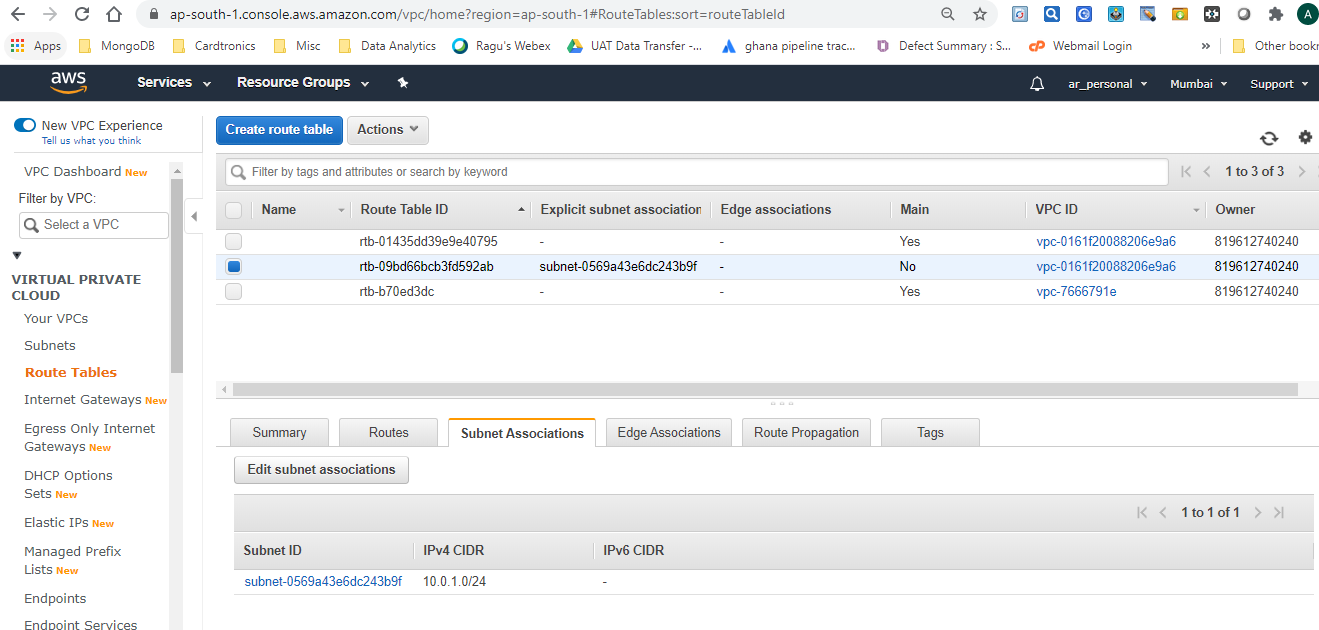
DB Security Group



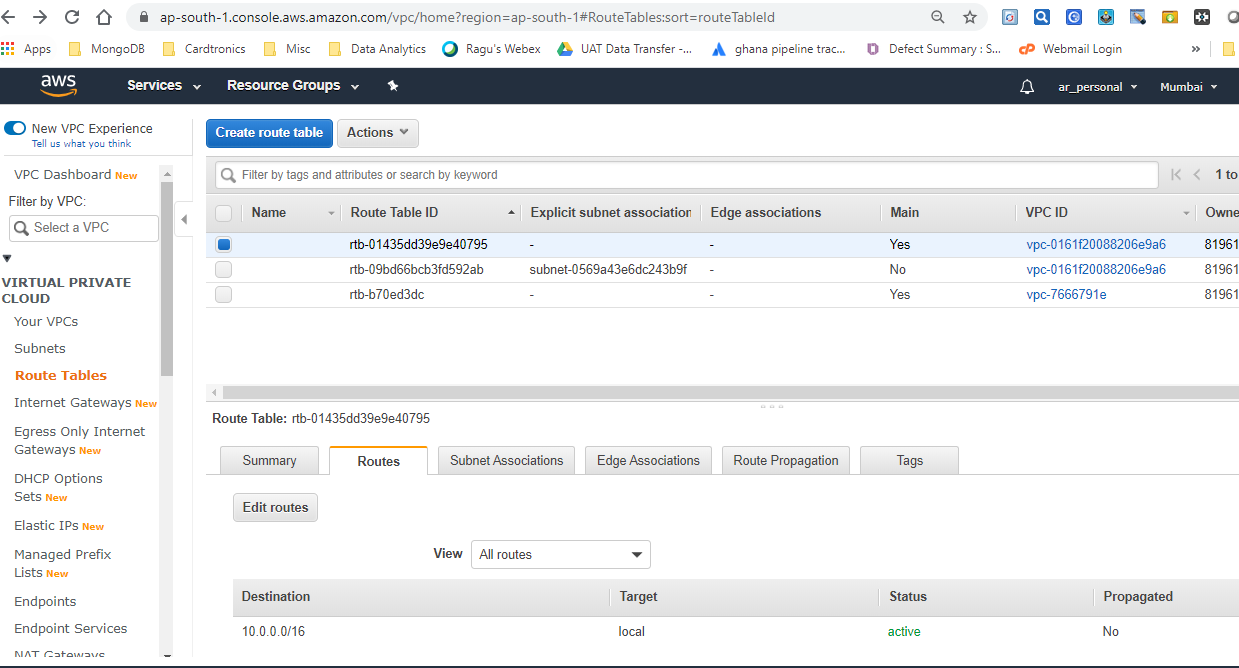
Route Table



Association of Route Table with Private Subnet



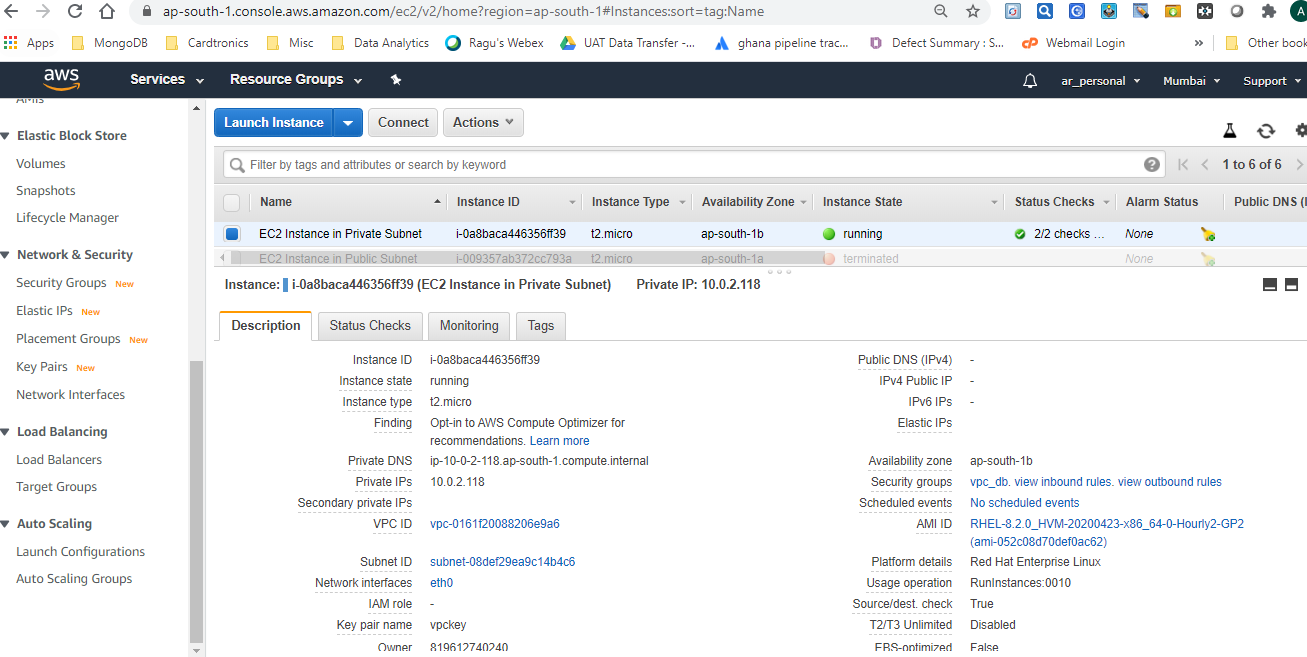
Association of Route Table with Public Subnet



EC2 Instance with Private IP

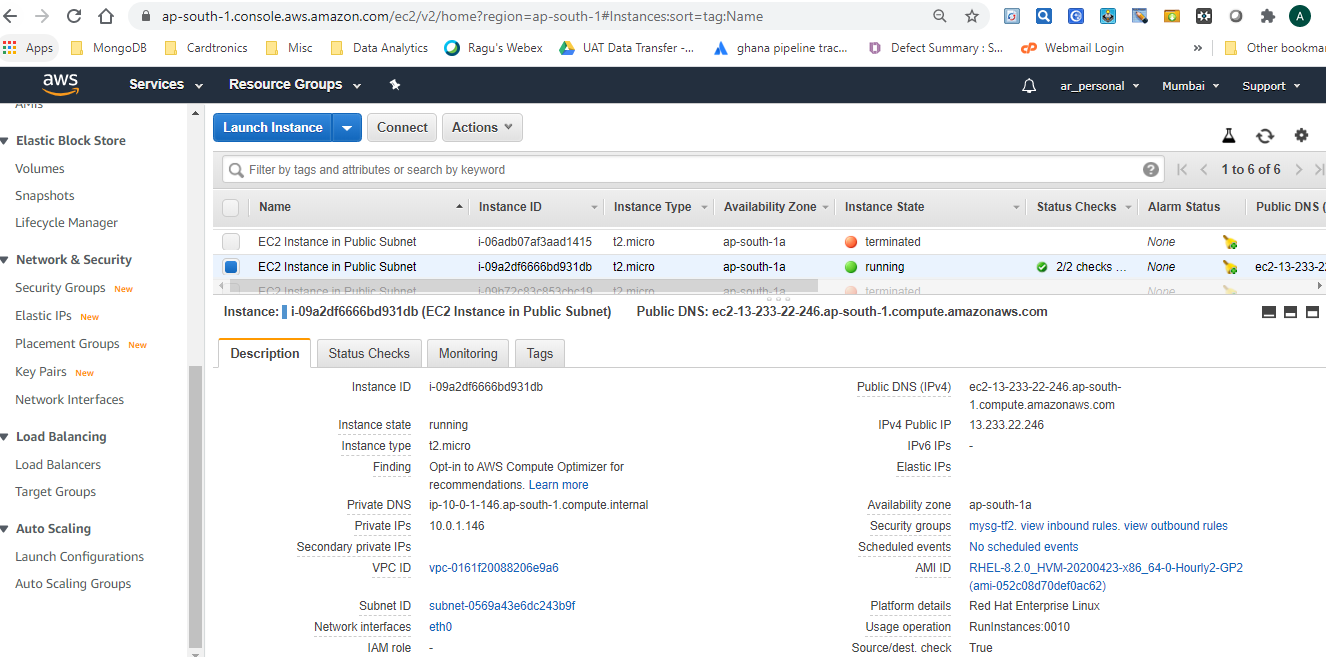
Whenever instance is launched without enabling the Auto-assign public IP setting, it won't have public assigned.

We won't be able to SSH to the instance which has private IP.

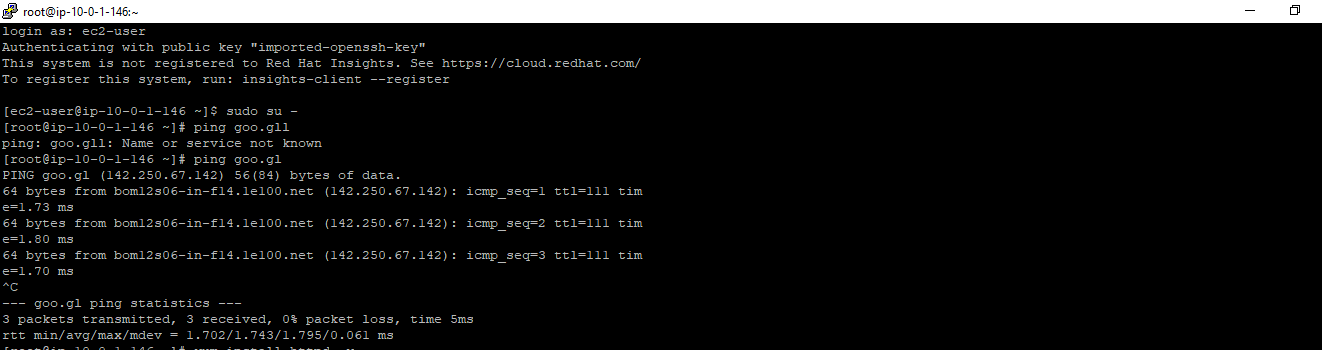


Ec2 instance with Public IP

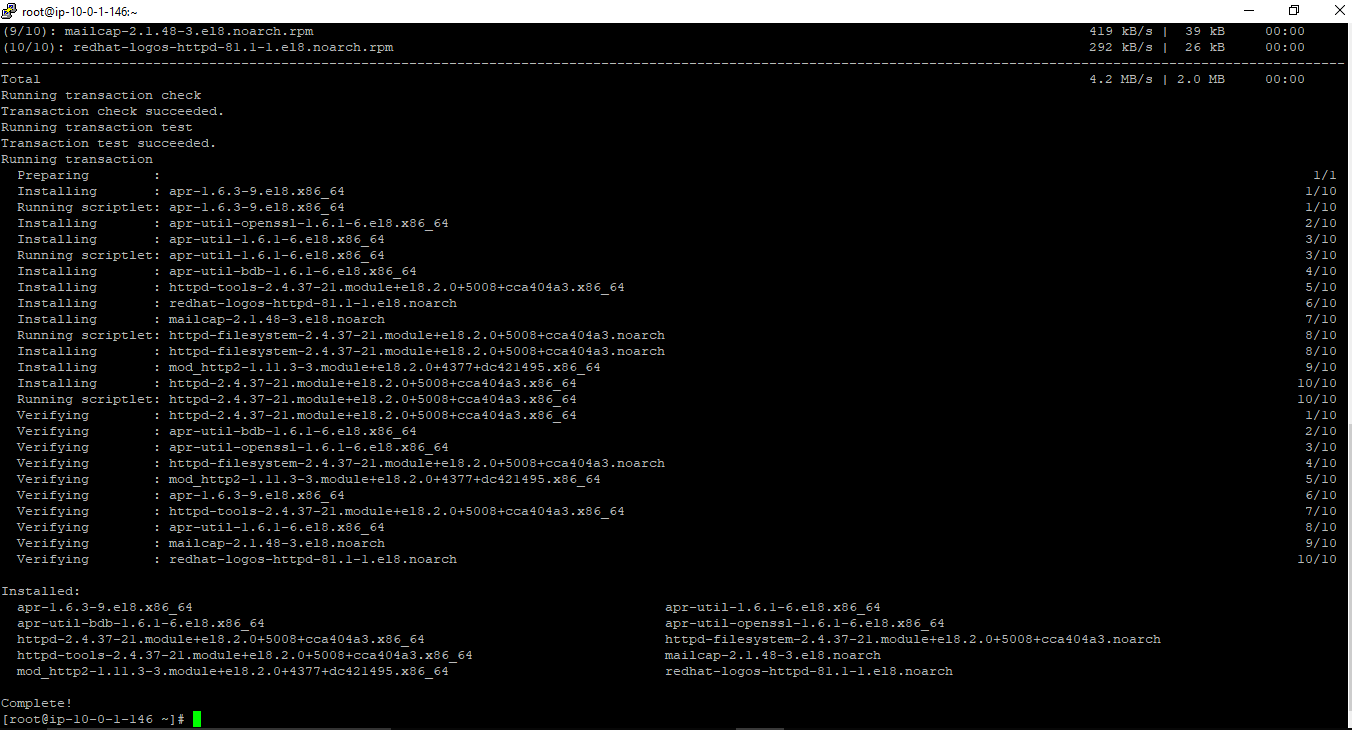
Each public instance has 2 IP's out of which Public IP is assigned to Router. As & when we launch the VM, public IP is assigned through DHCP & it's added to Router so as to enable NATTING.



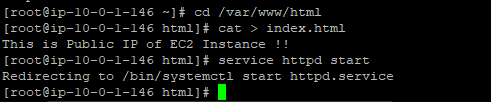
Login to EC2 instance having Public IP using Putty

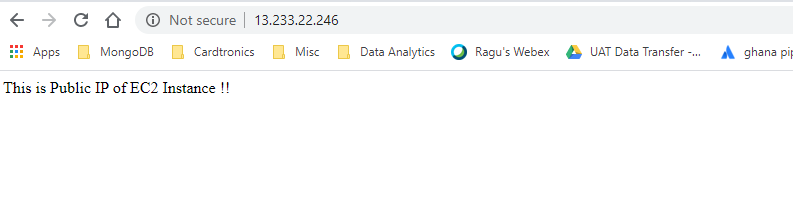


Through this instance we can install Apache web server since it has internet connectivity.

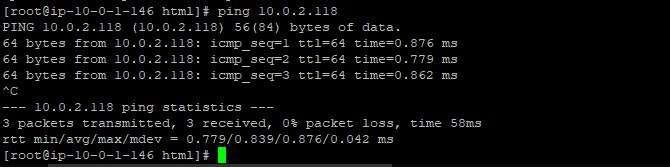


We can create our own index.html & verify the site

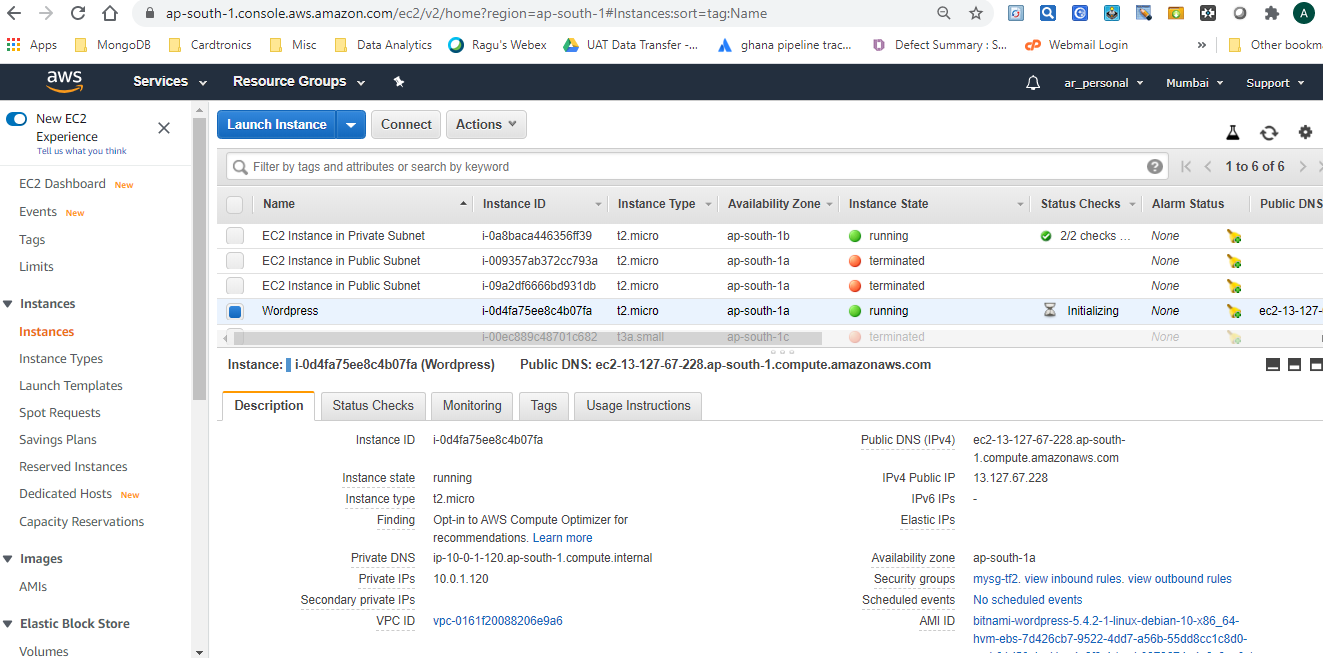




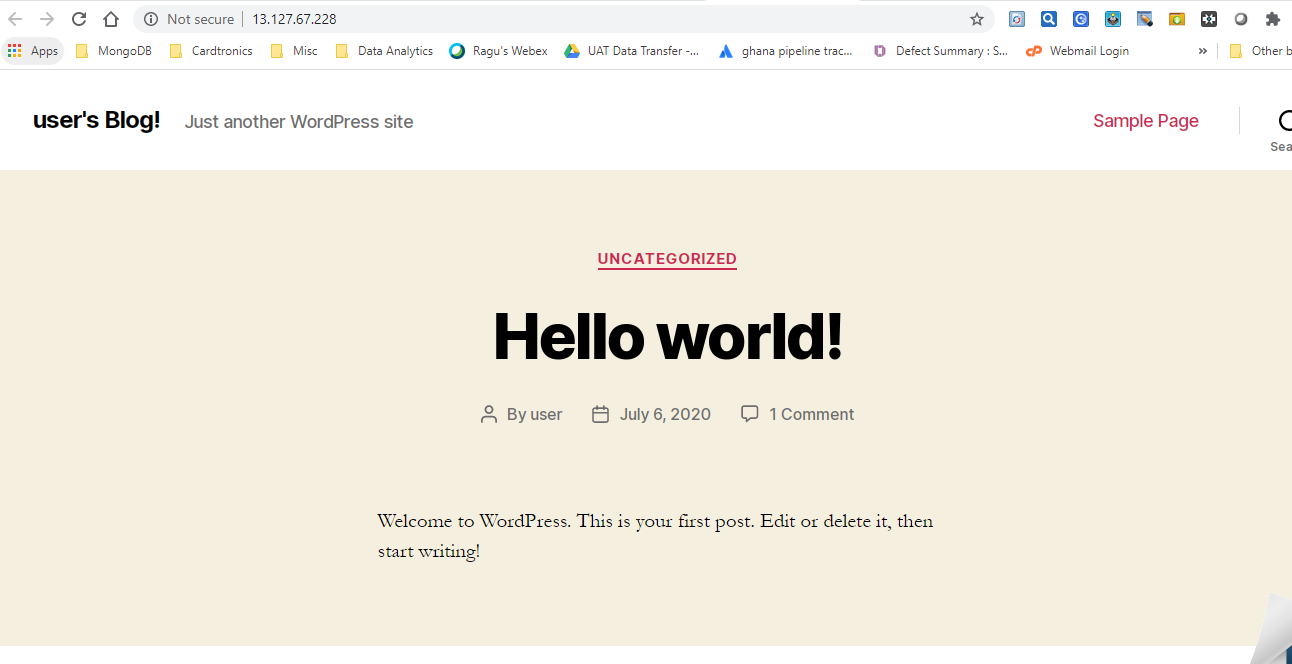
We can Ping to Private EC2 instance through Public EC2 instance as they belong to same VPC



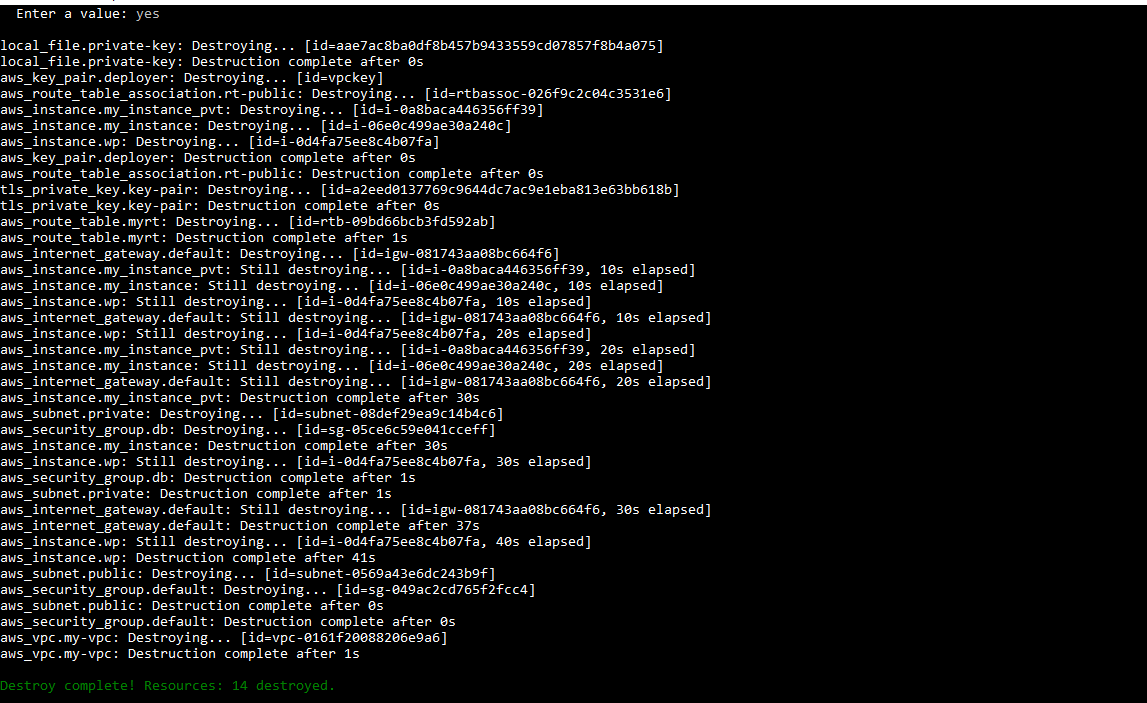
WordPress Instance in Public Subnet with Security Group:

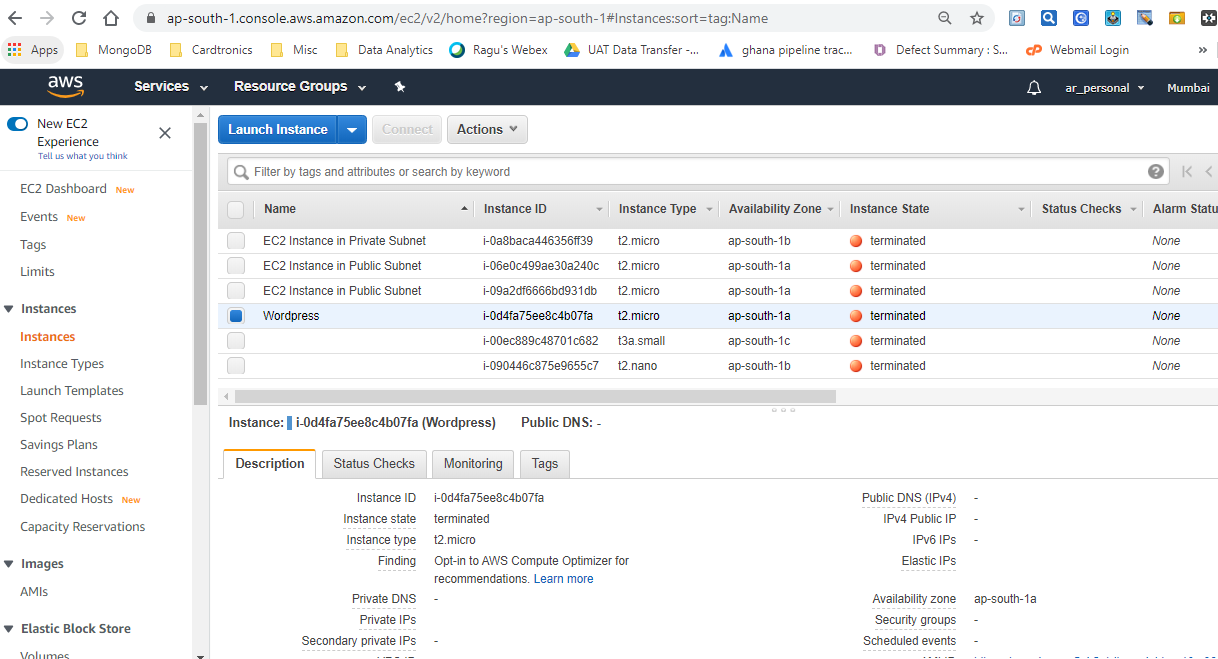


Verify that WordPress is running as under

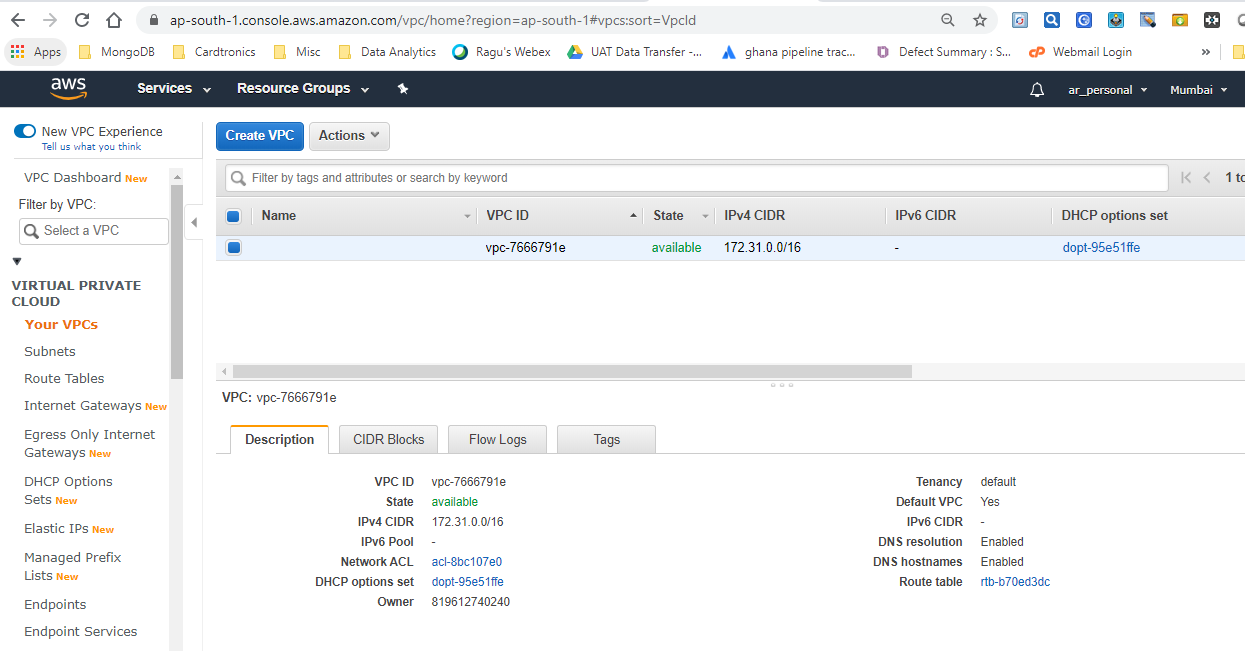


Entire infrastructure can be destroyed with one single command

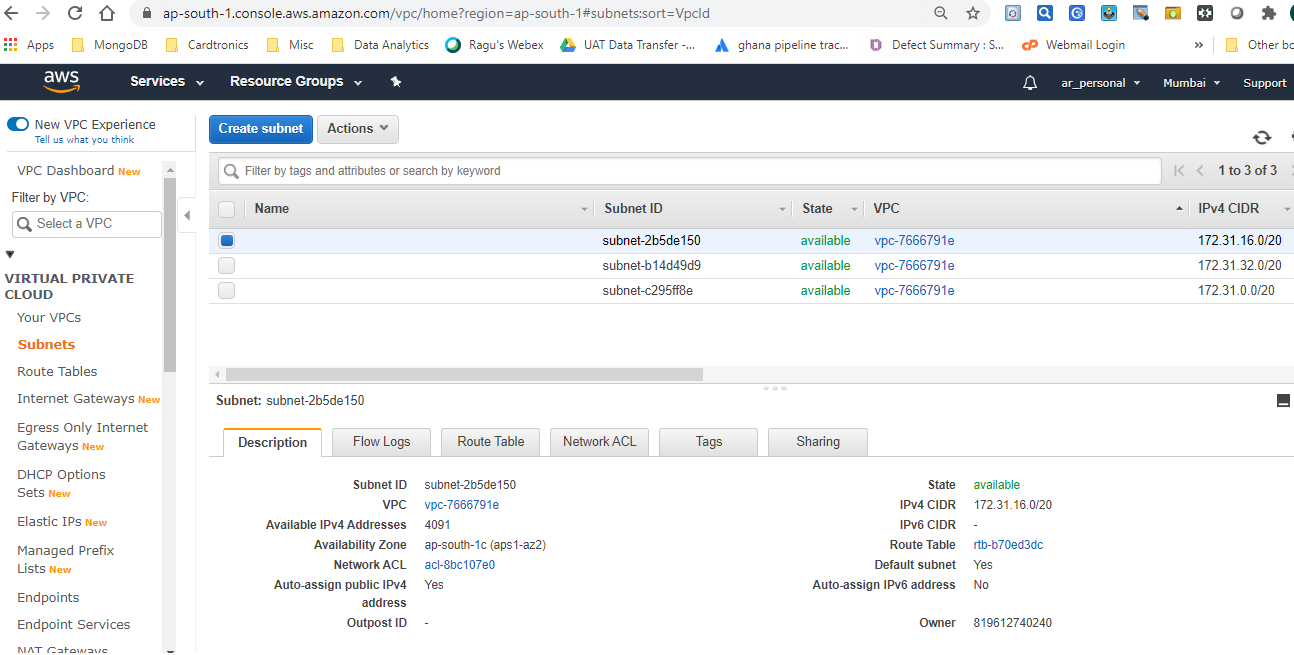




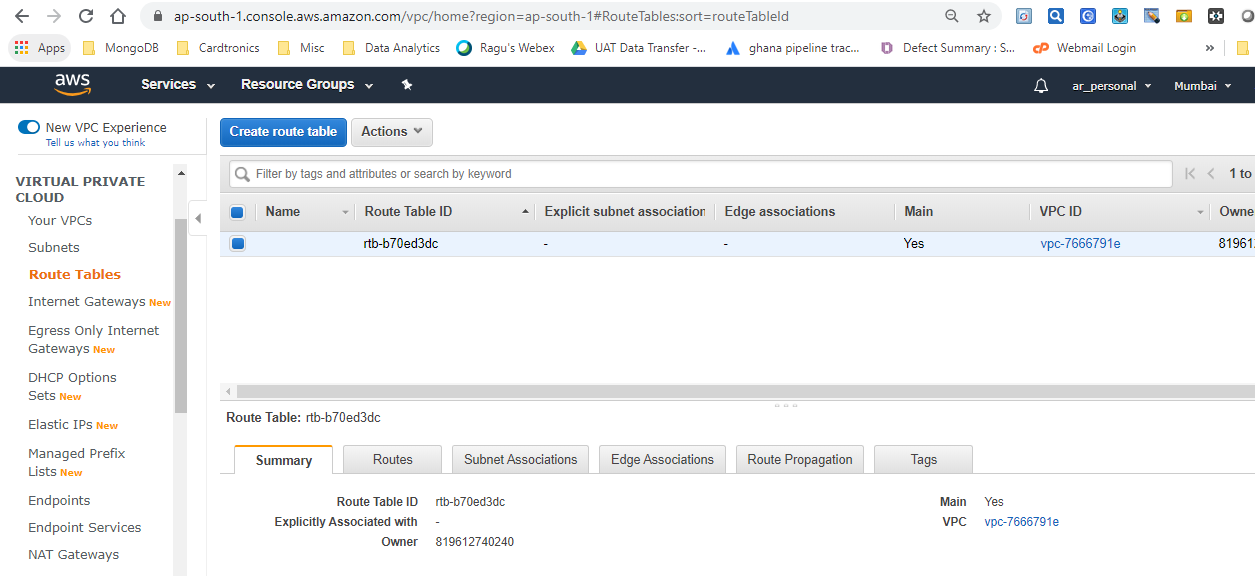
Default VPC remains, however the VPC created by Terraform gets deleted. (Same for subnets as well)



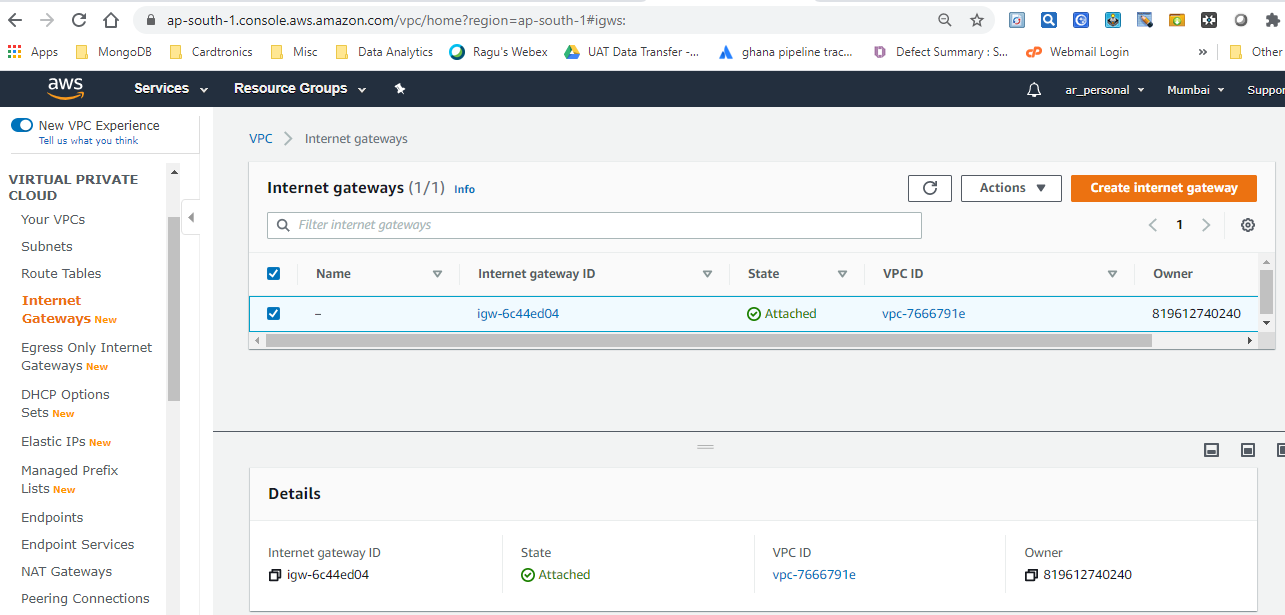
Sunsets –Subnets associated with default VPC remains.



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



Default IGW remains



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.