**EKS- Day#2 Task**

EKS is a fully managed service which is managed by AWS.

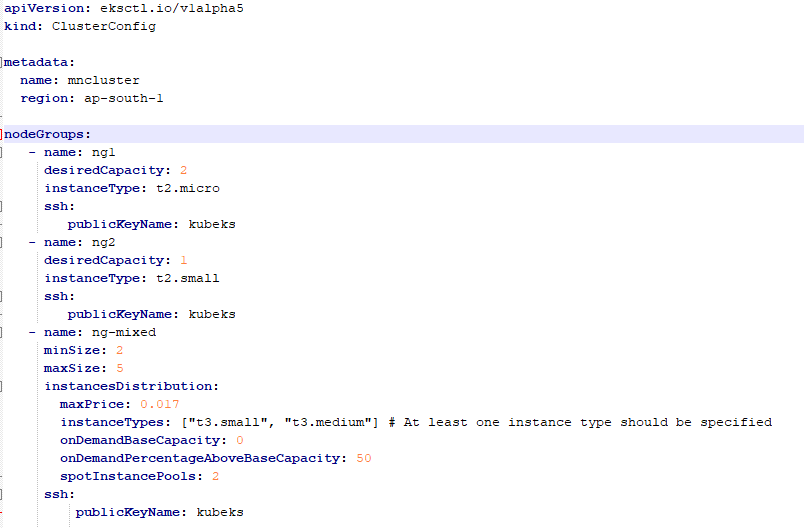
It is highly integrated with AWS services such as EC2, EBS, ELB, EFS, and VPC

**Spot Instances**: Below yaml shows how to set up an eks cluster with a mix of spot instances & on demand instances.

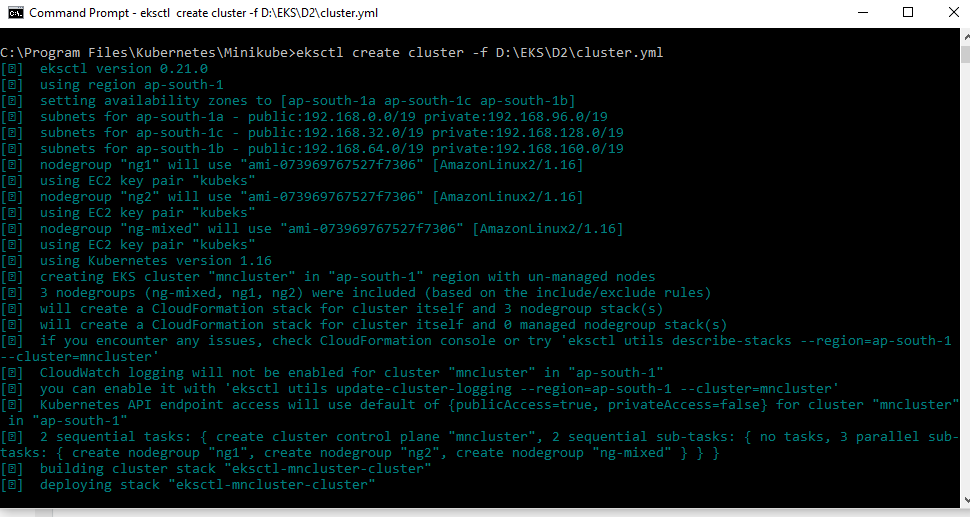
Spot instances (slaves nodes) are launched only when bid price is above the spot price. We can set the bid price for spot instances while launching it.

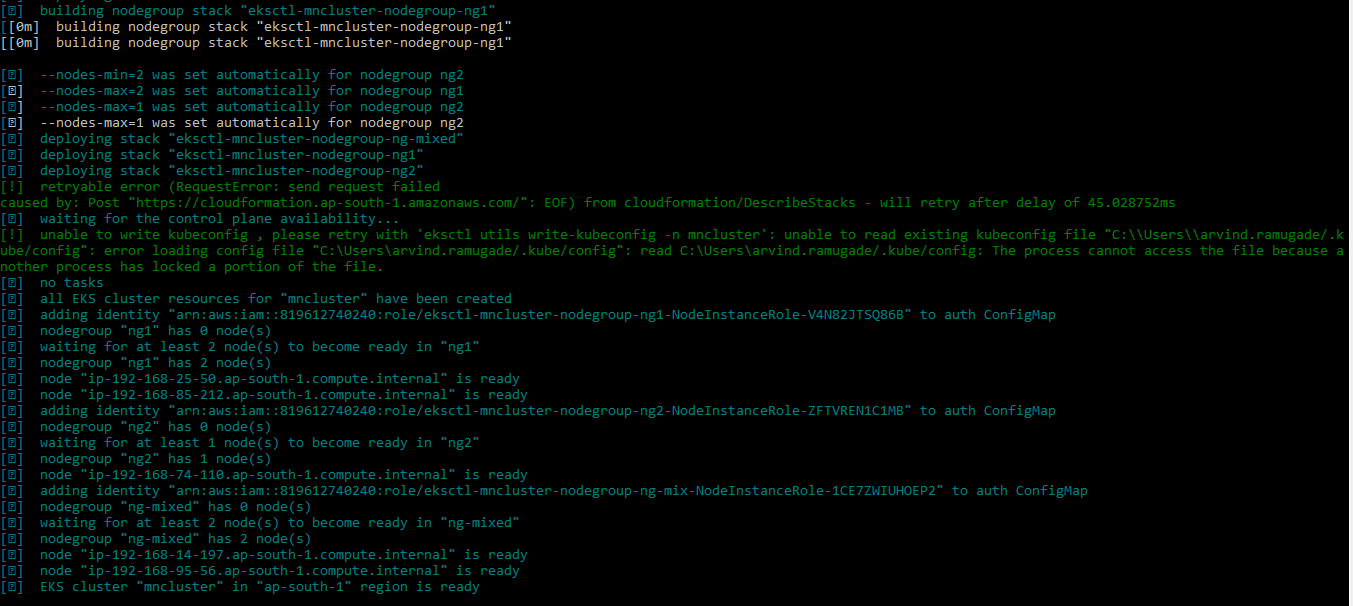
​The instances will be launched only when bid price is above the spot price. If the spot price is above bid price Spot instances will be terminated by AWS. There is a pricing associated with Spot instances.

Using eksctl we can create a single node group with mixed instances such as t2.small, t2.micro etc.

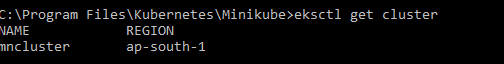


EKS Cluster creation through command line

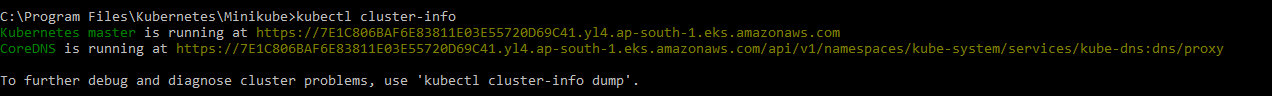


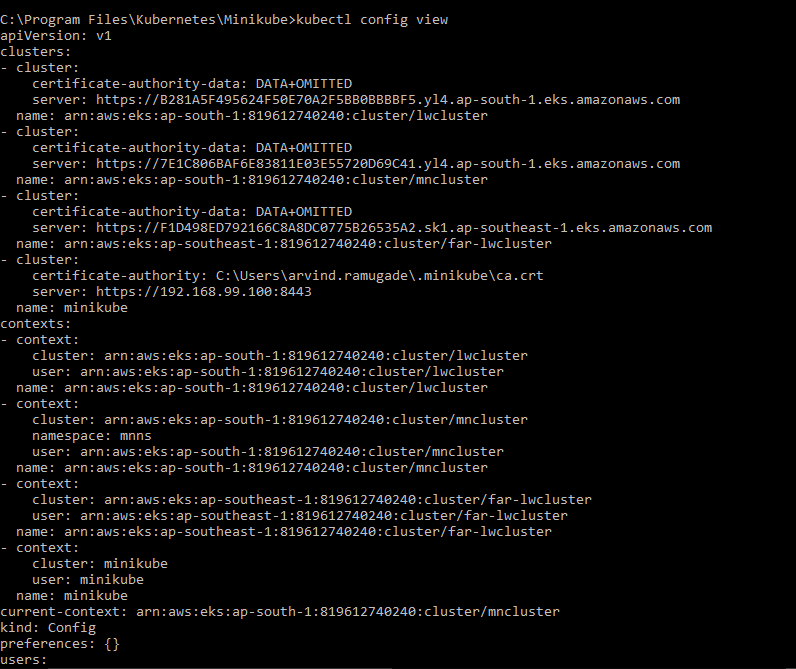


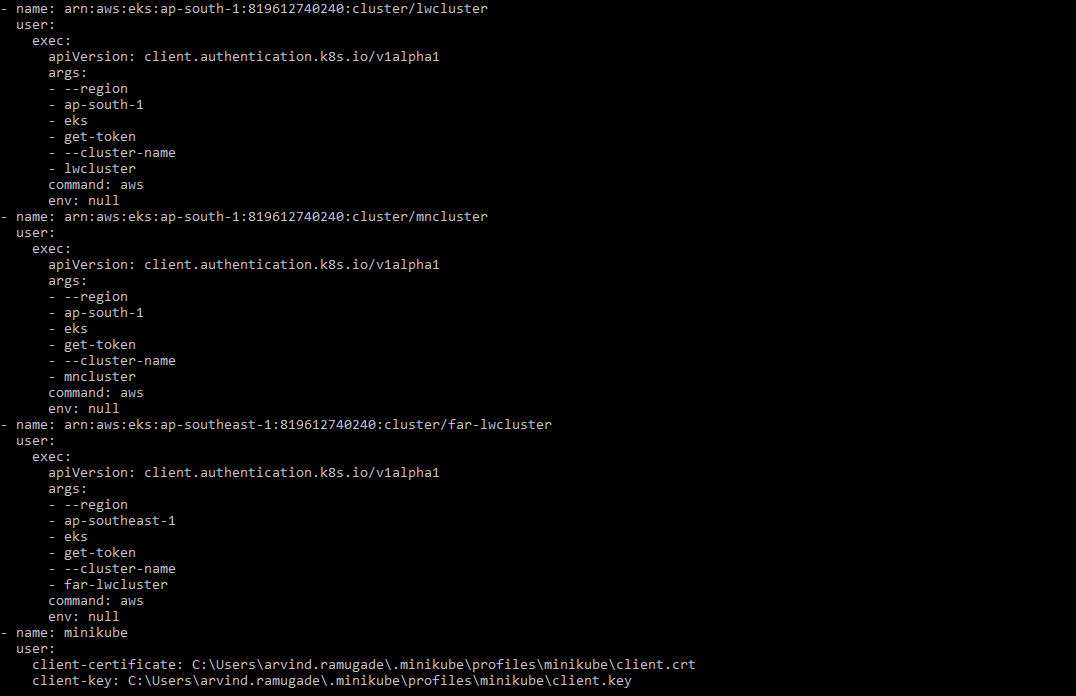
EKS Cluster verification through command line



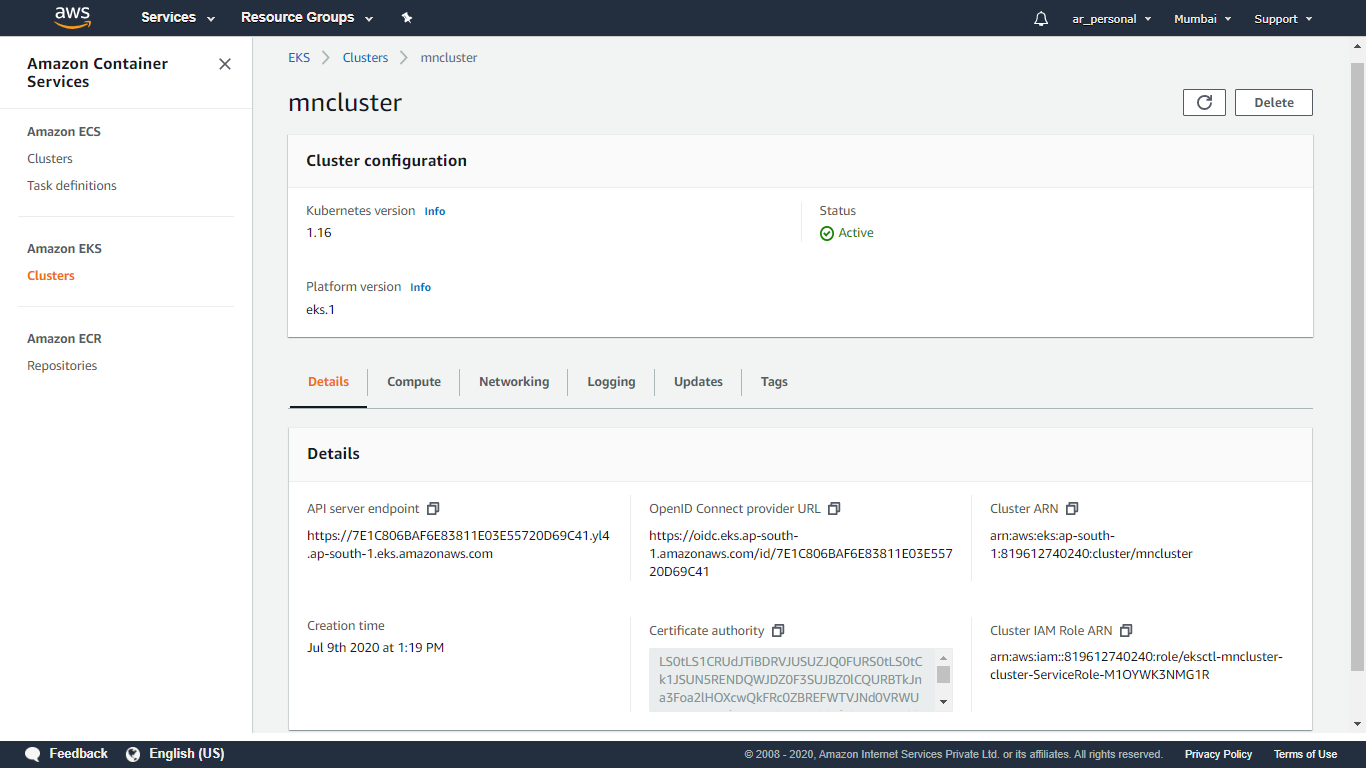


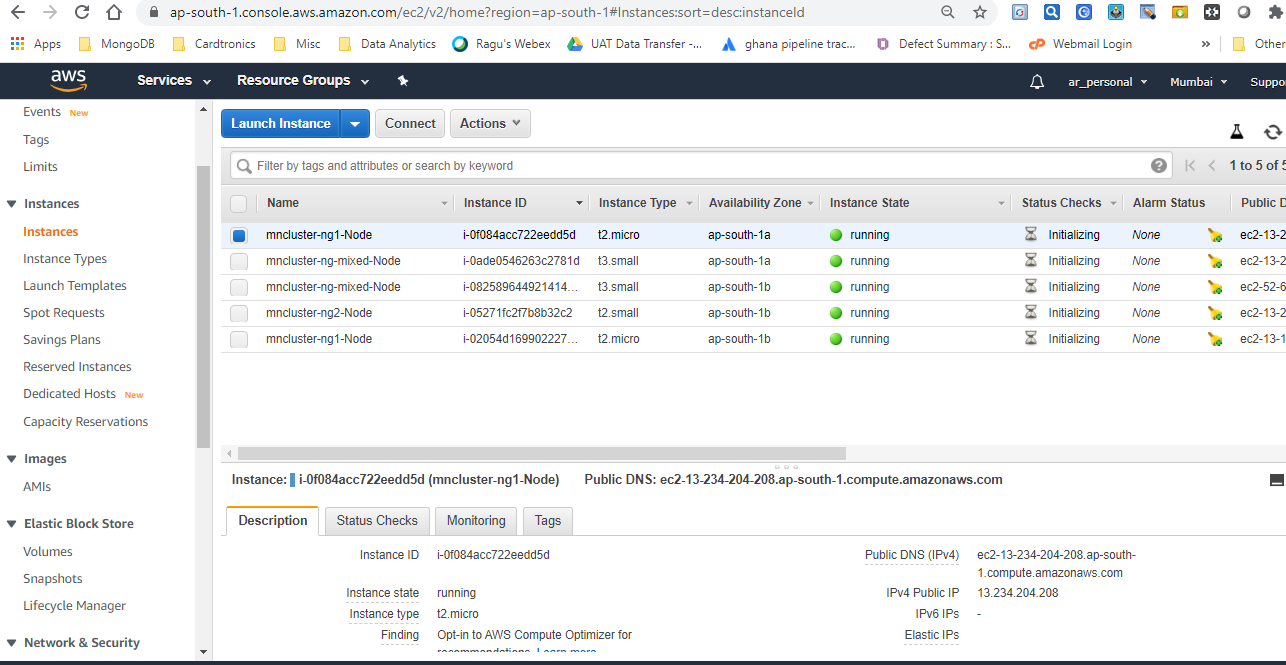






We can verify the cluster creation through AWS Console as well as shown below



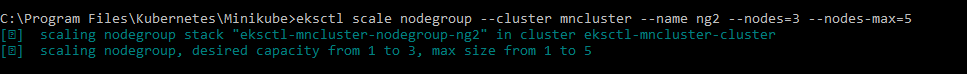


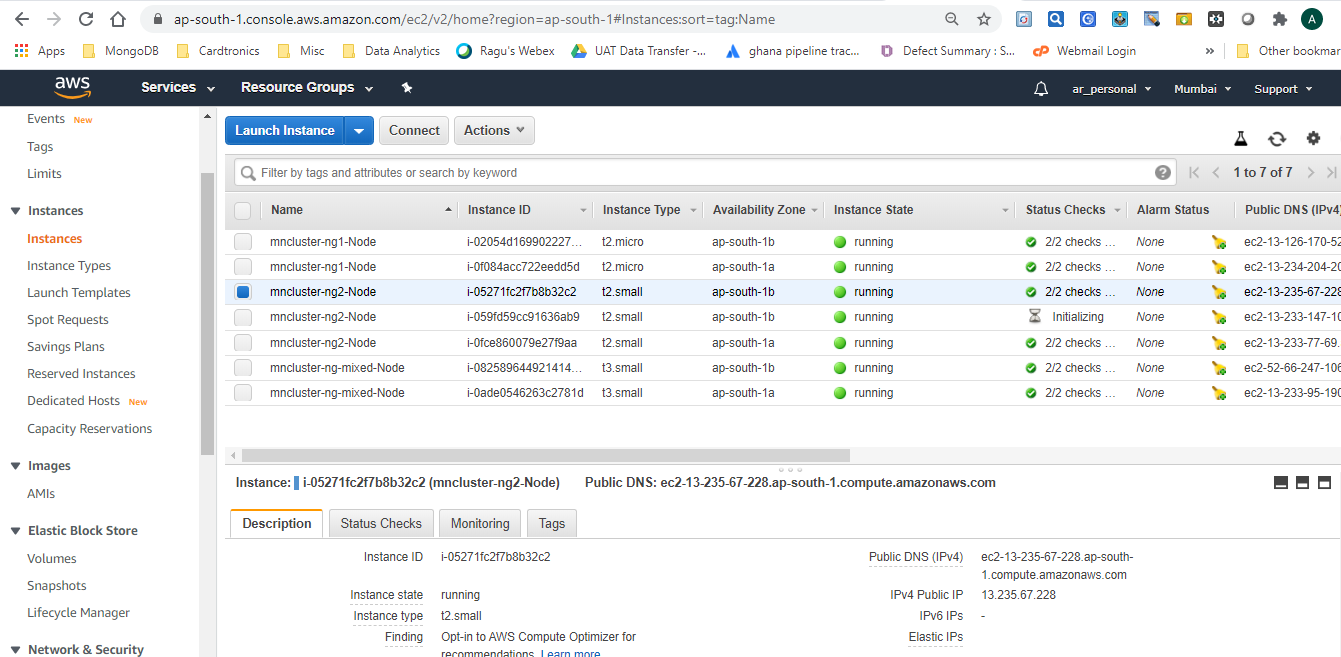
Using below command we can update kube-config file so that we can use kubectl command in EKS cluster

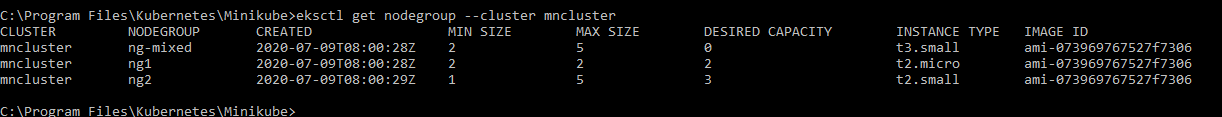
aws eks update-kubeconfig --name=mncluster

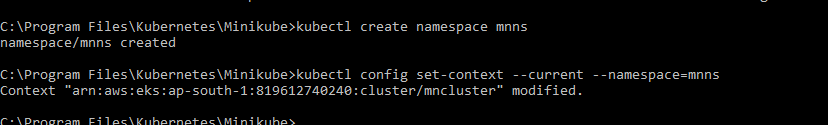
kubectl get nodes

We can scale our cluster for a particular node group as under:









By default in a container there is no connectivity for pods running on multiple nodes. If there are multiple pods in a single slave node they can communicate with each other, however they can't connect with pods in another slave. ​

Using CNI (also known as flannel)​we can achieve this.  CNI, VPC, Subnet created by Eksctl automatically once we set up them multi node cluster (1master& 2 slaves)

k8s\_coredns manages outside network connectivity.

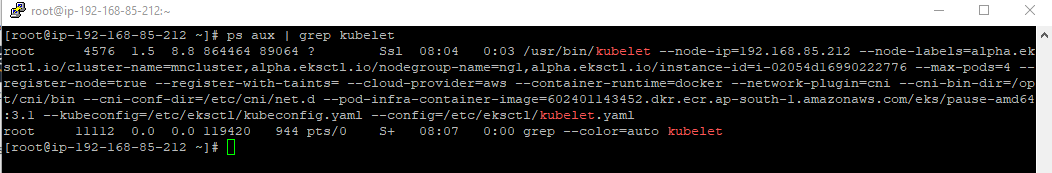
Limit on no. of pods in a node:

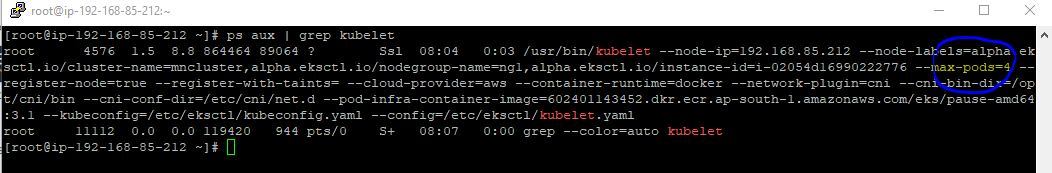
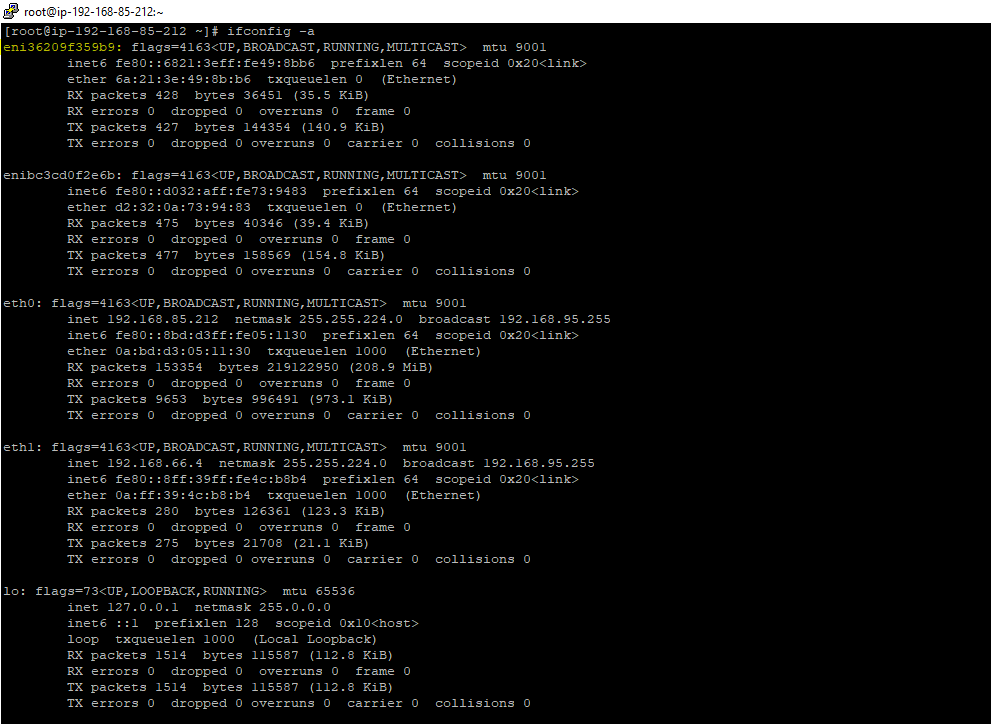
Following command shows maximum no. of pods​which we can launch

ps aux | grep kubectl

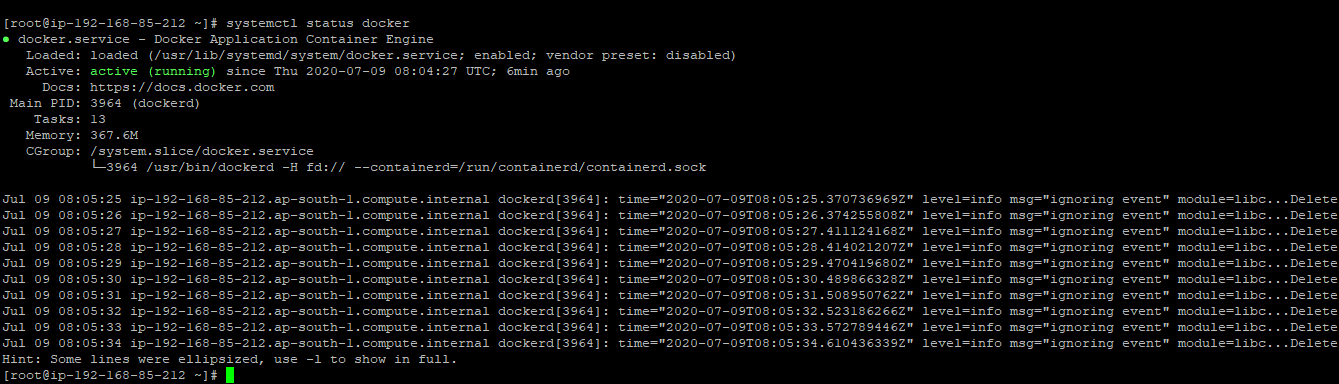
The limit on no. of pods which can be launched in a node varies based on instance type.  e.g. for t2.micro instance it has capability of 4 NIC of which two are used for instance IP address only two NIC are available and we can run only two pods in the t2.micro.

Network interface attached by AWS CNI for pods help in inter connection with other pods on different nodes.

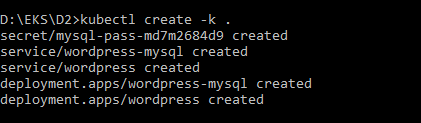


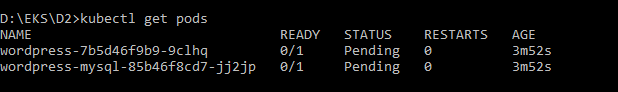
  


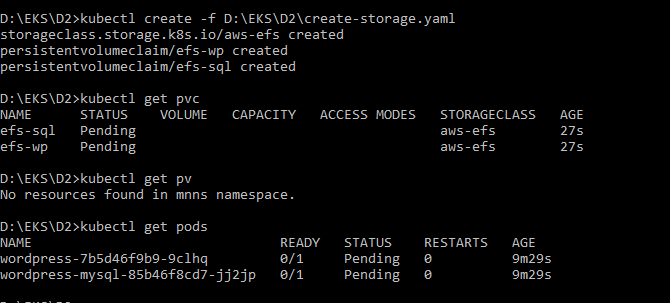
We can confirm that Docker is already installed in the instance due to EKS cluster.









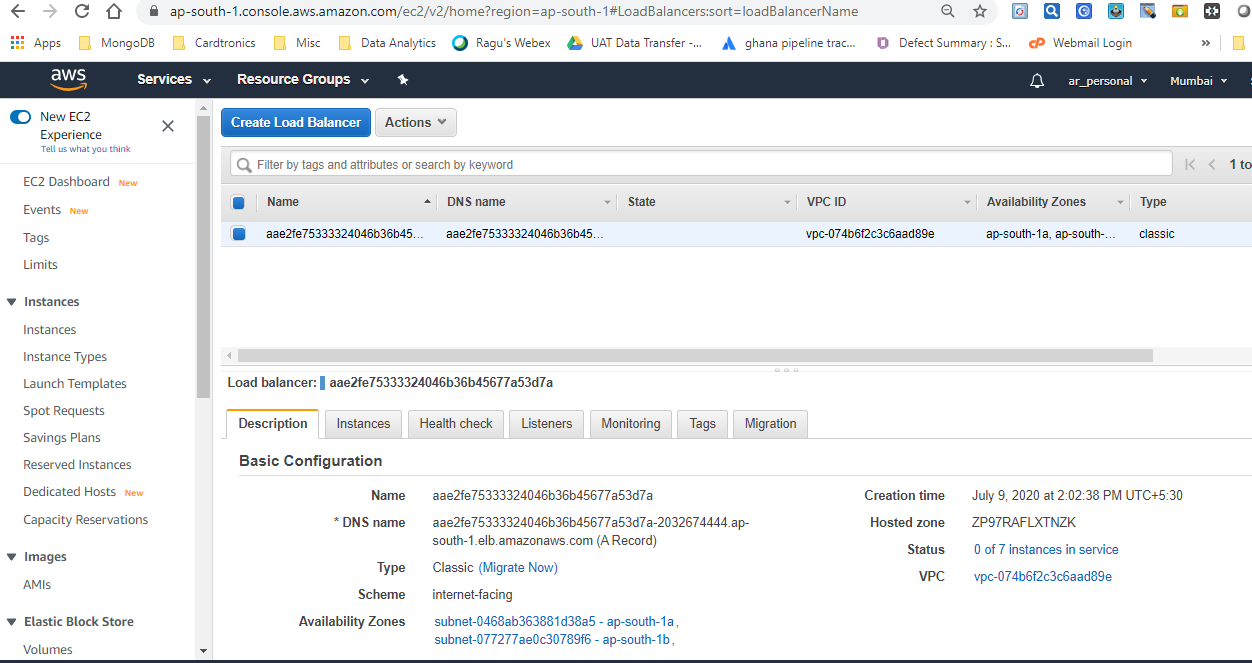


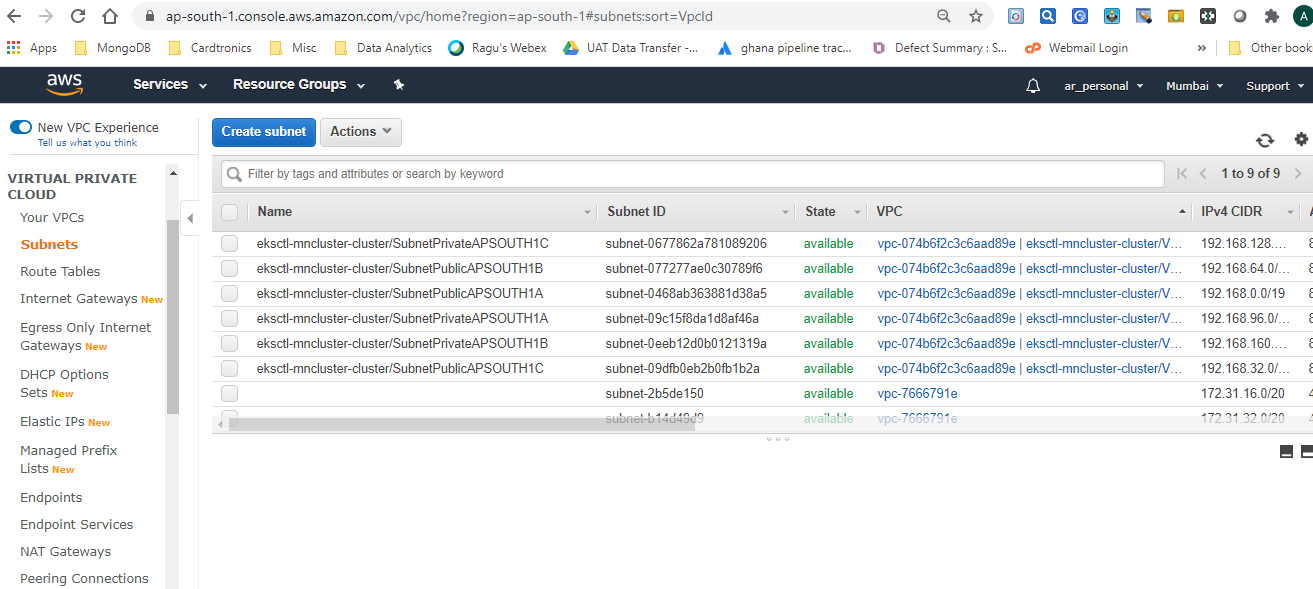
EKS Supports LoadBalancer service which automatically balances traffic on slave nodes.

Kubernetes service type is LoadBalancer & by default it uses Classic Load Balancer which are public facing.



Using annotations we can manage LoadBalancer configuration.

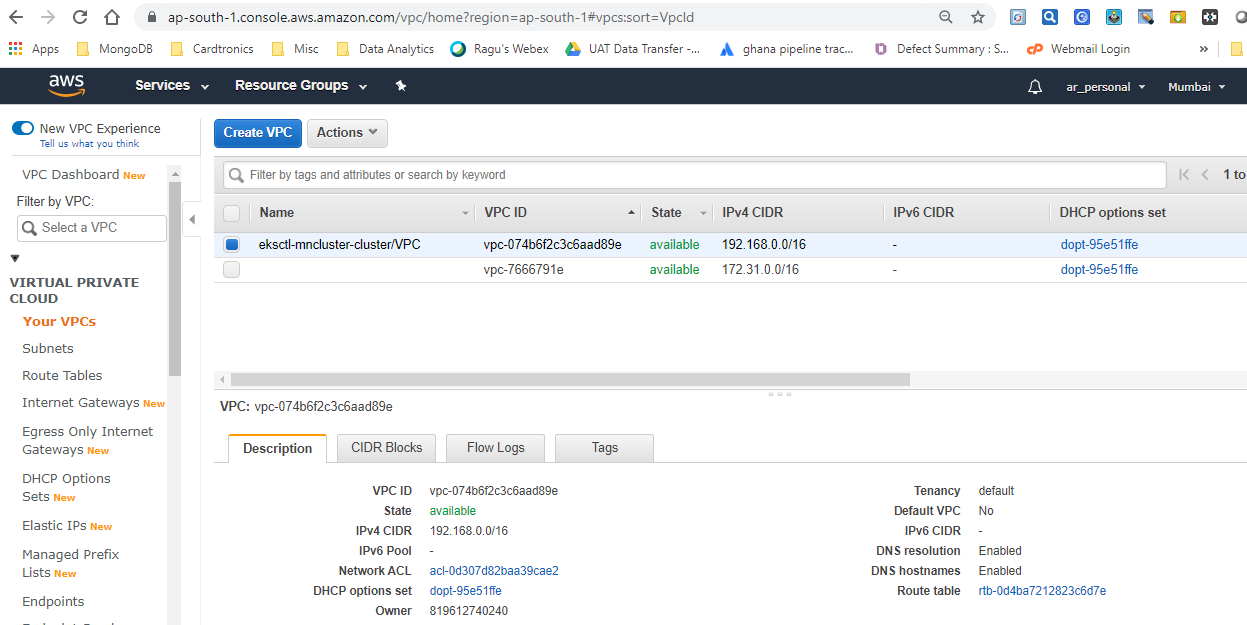


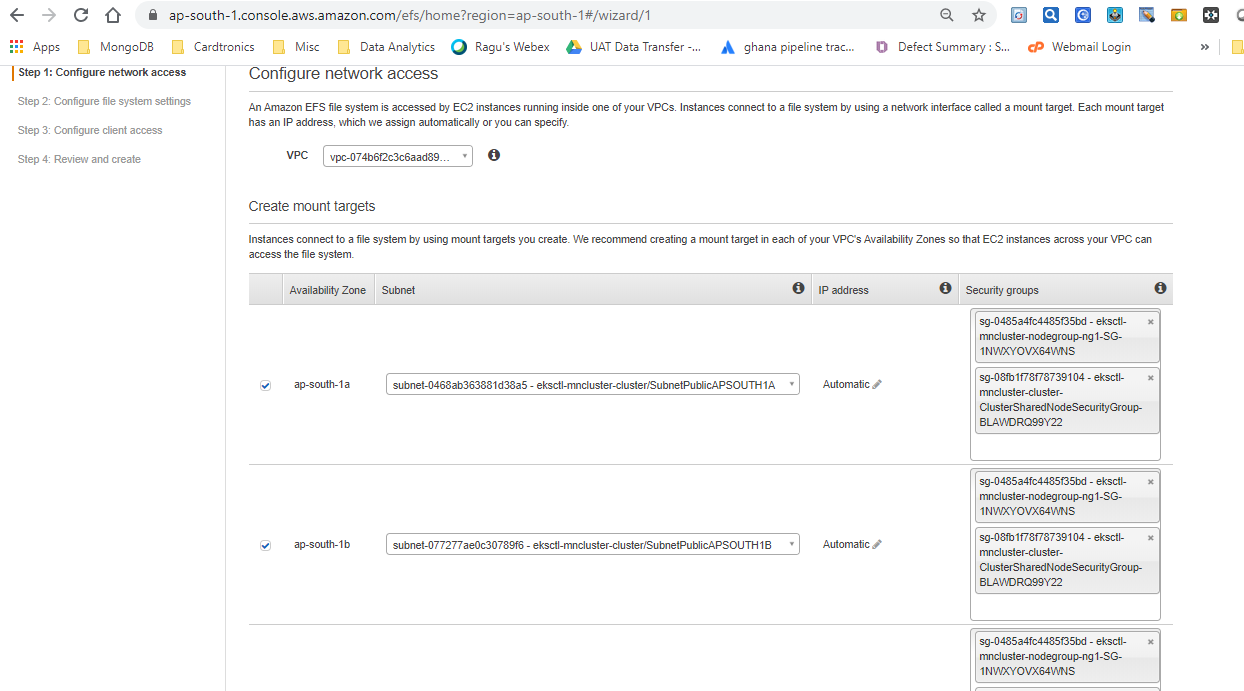


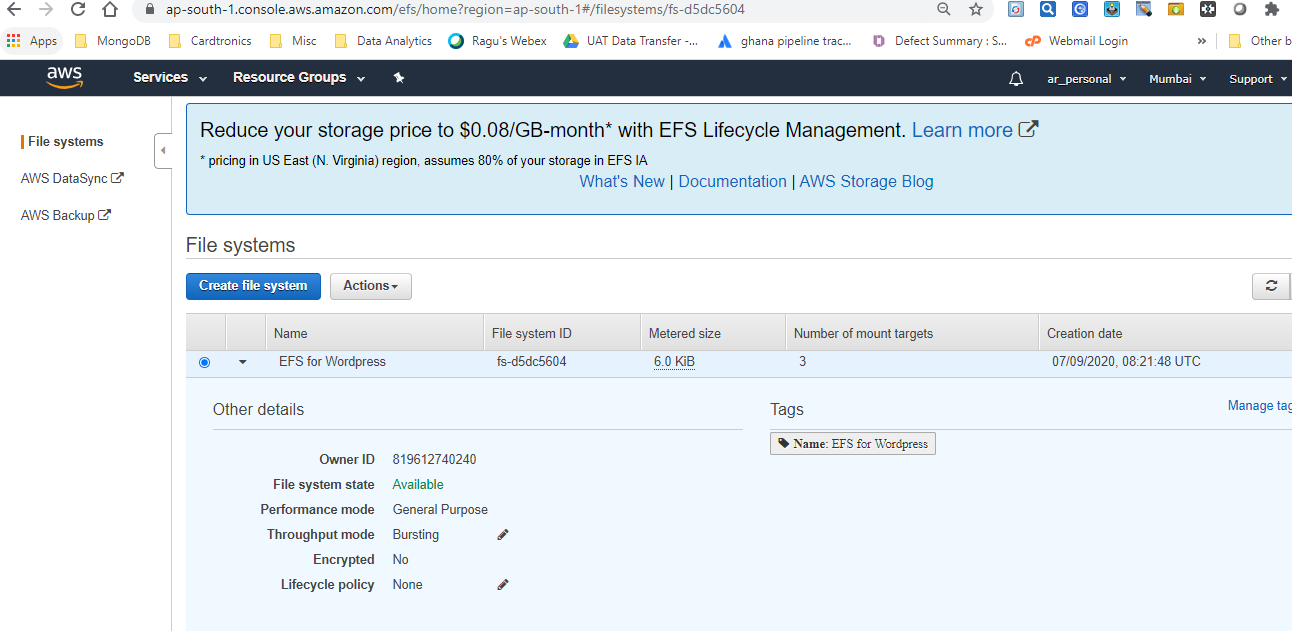
**EBS challenge:**

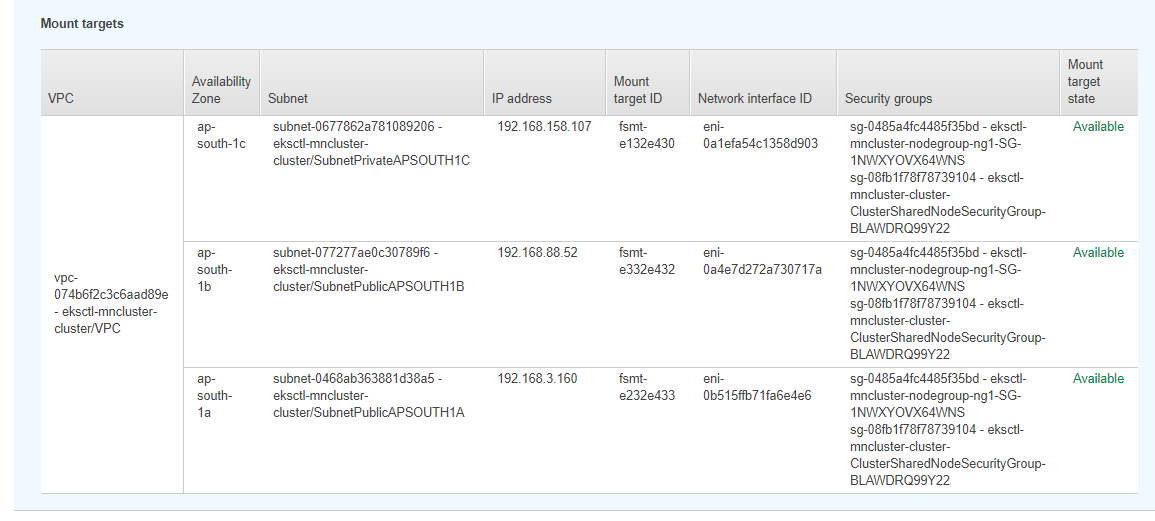
With EBS we can attach only one EBS volume at a time with the EC2 instance.​EBS volume can't be used to connect to EC2 instance launched in different availability zones.​This is a problem in a multi node cluster set up where multiple slave nodes can't be attached to centralized storage.

This can be resolved by using **EFS service**. **EFS is a centralized NFS storage** & it spans subnets in VPC. However, the Security group should be assigned to all the subsets while creating EFS.​In EFS multiple nodes can concurrently access the storage without any performance/operational overhead.

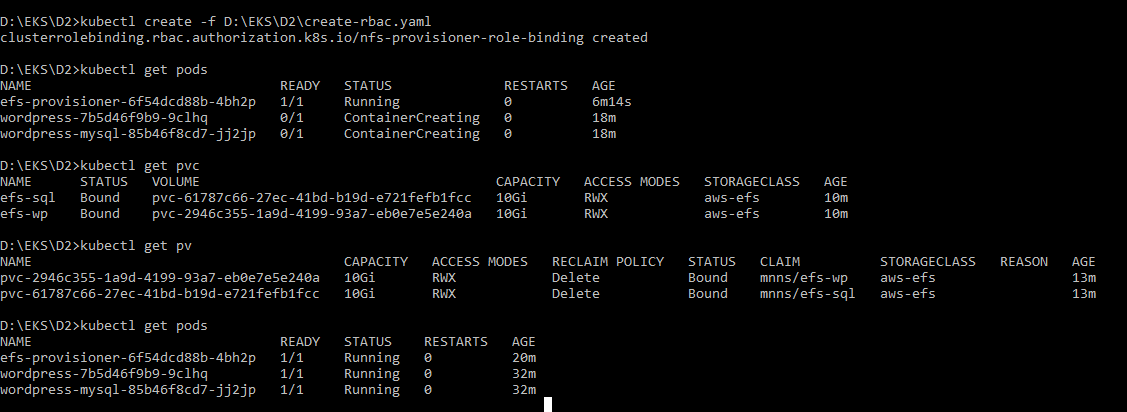




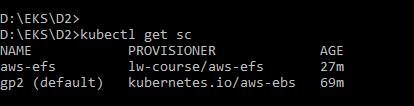


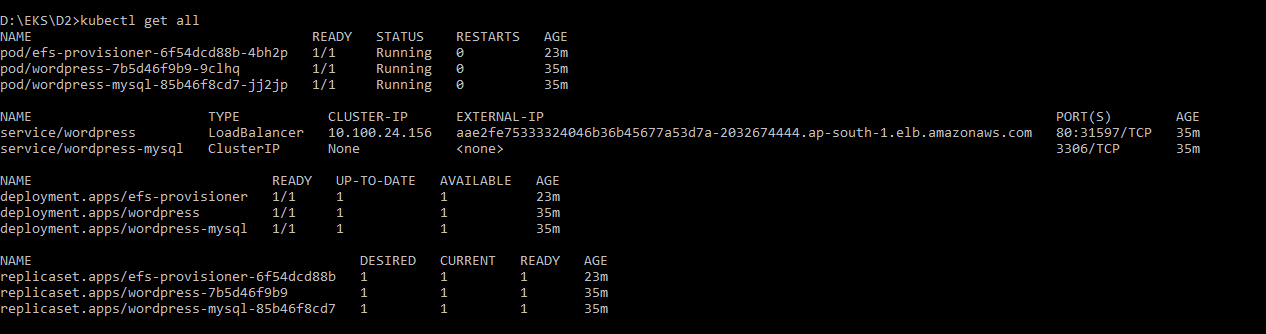




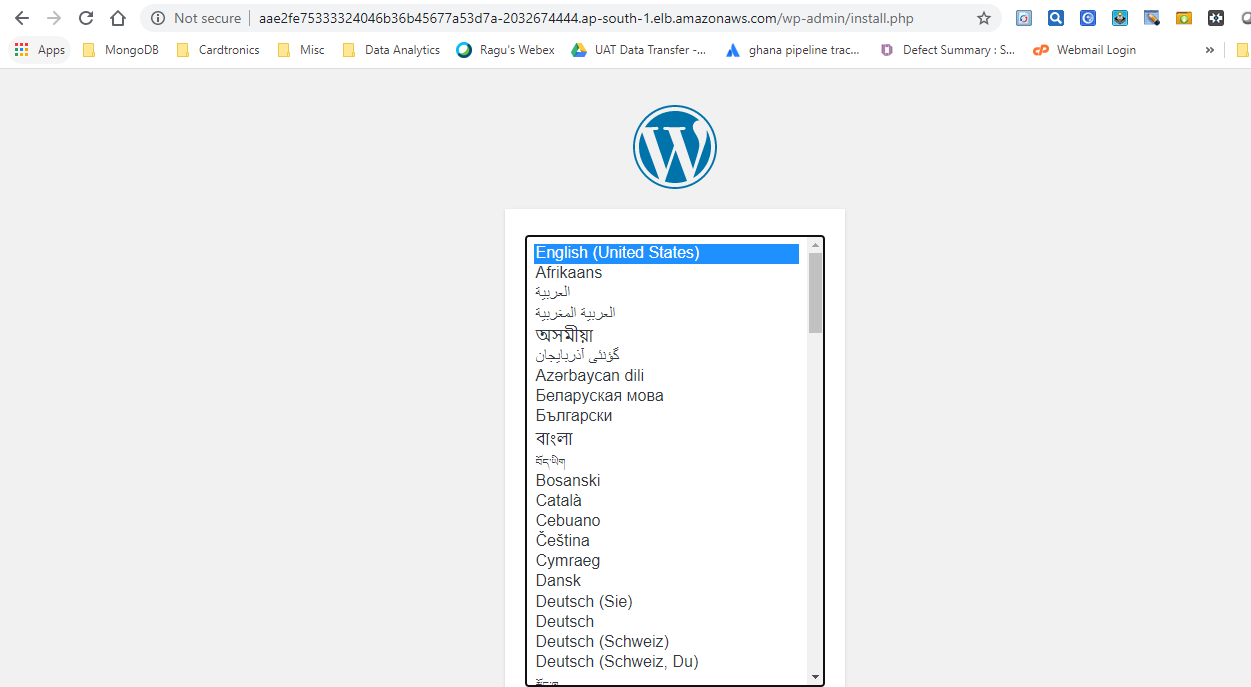


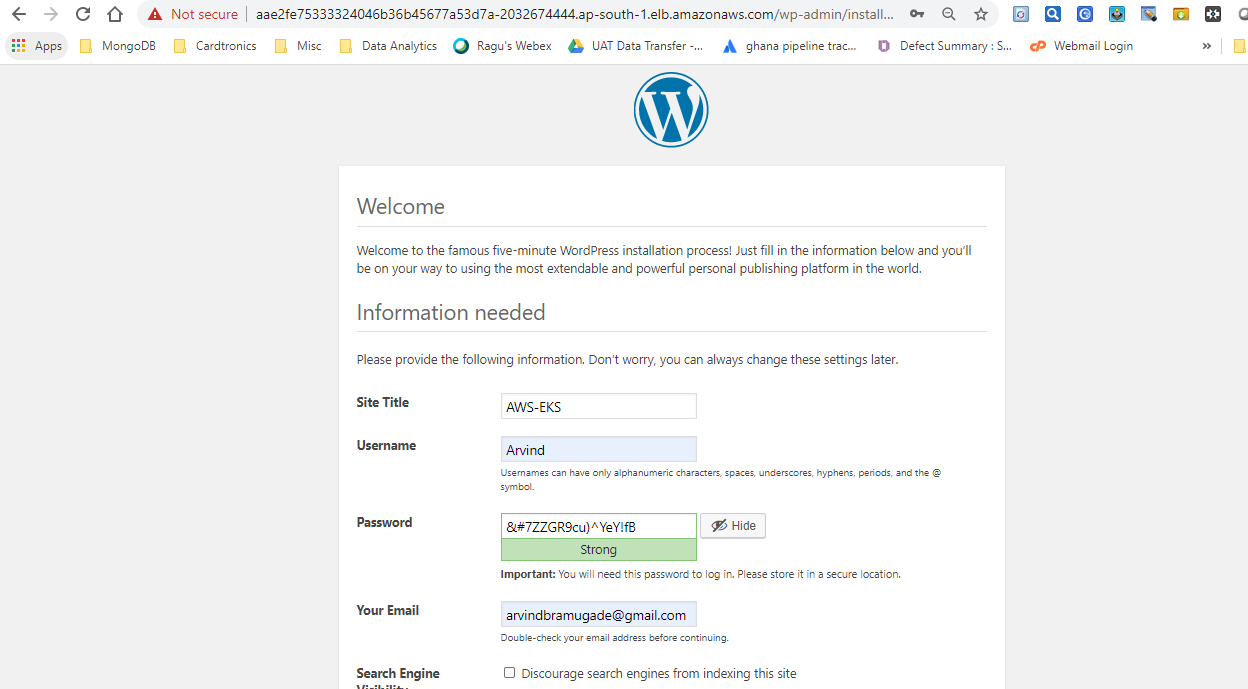


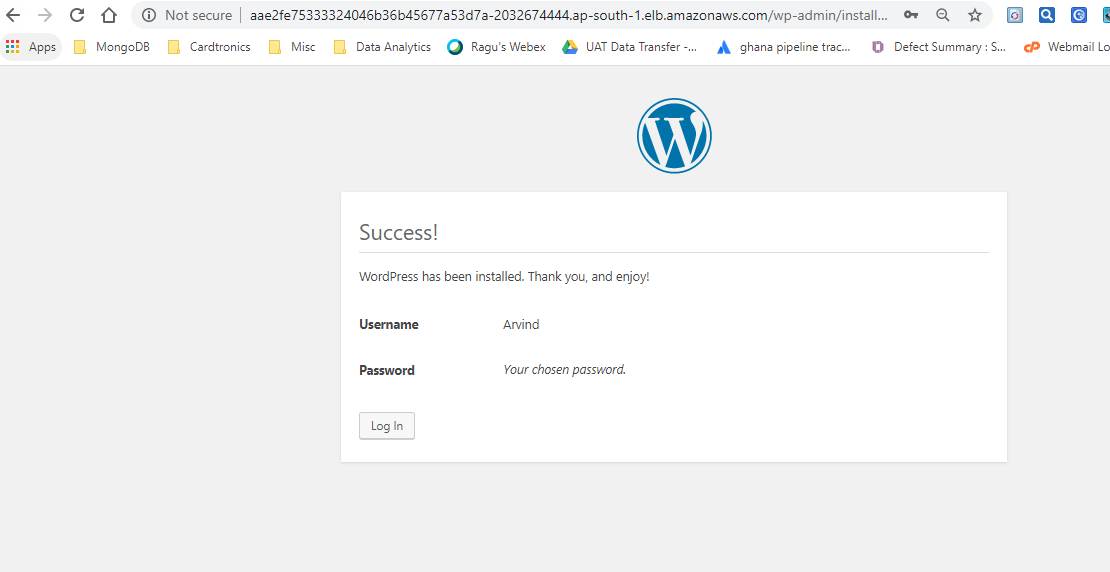


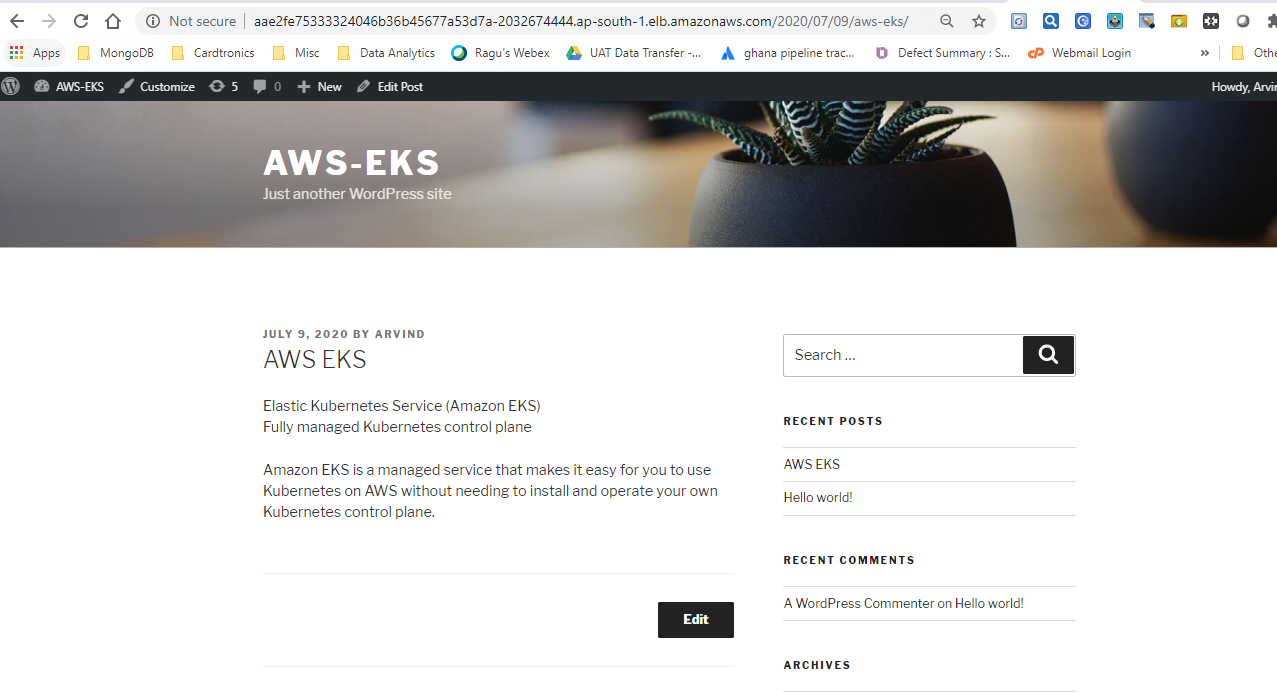


WordPress Application

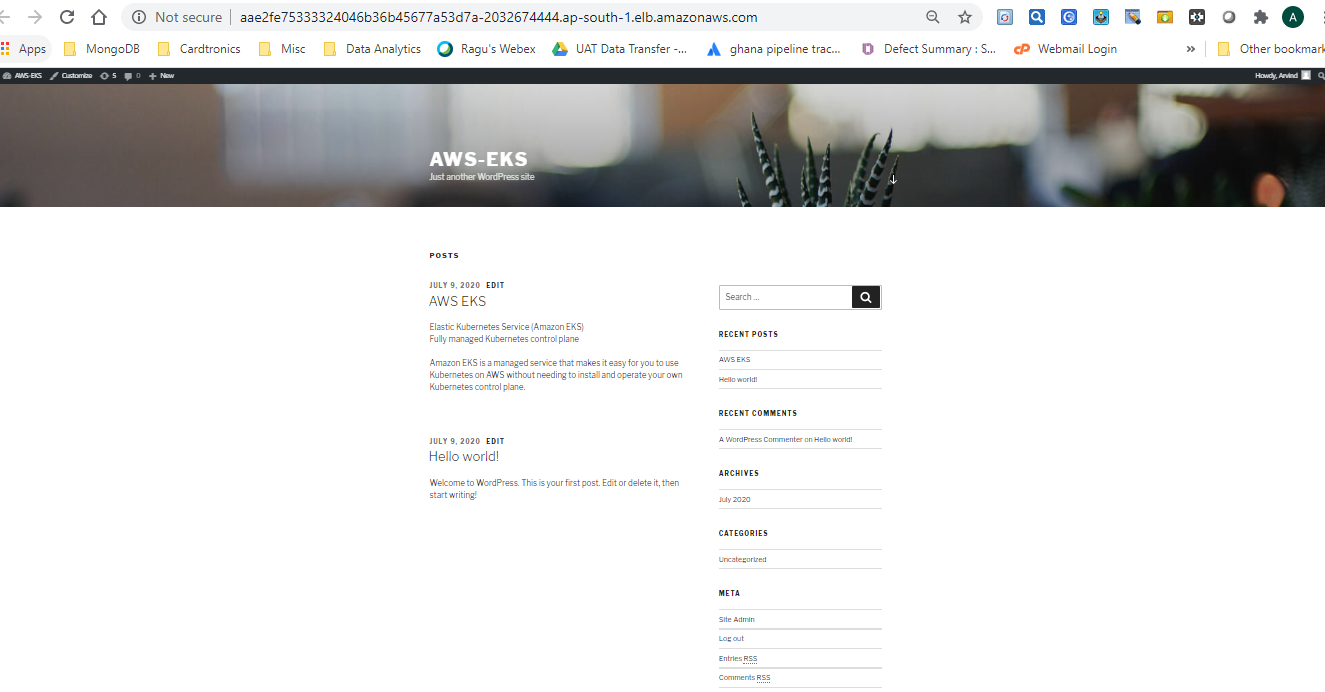


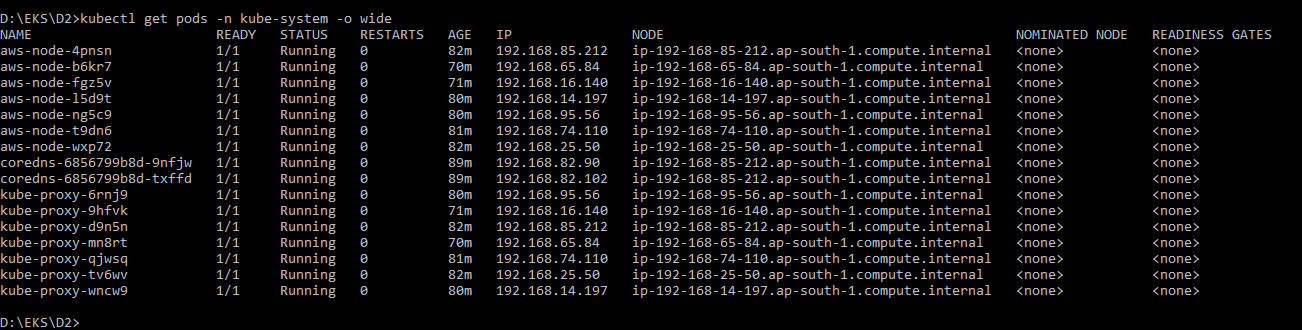






After clicking on LoadBalancer following page will be opened





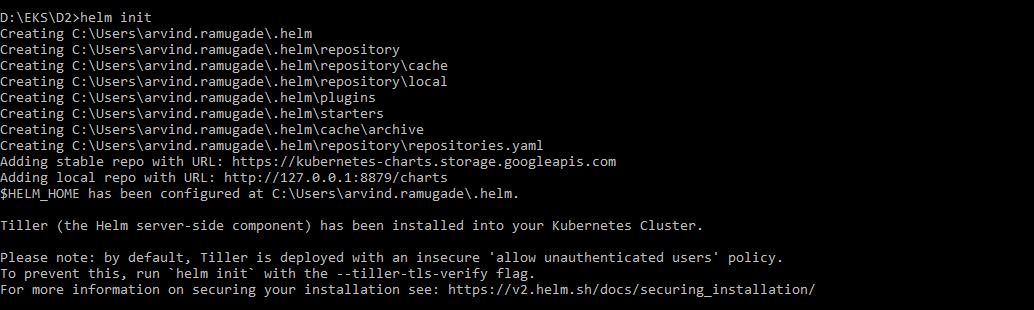
**Helm**

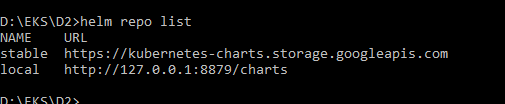
Helm is a client side program that provides the k8s software packages where we can launch the whole application in the kubernetes cluster.

**Tiller**

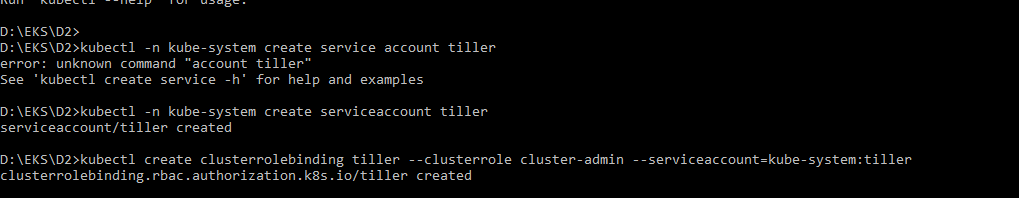
Tiller is a server side program to help the helm to set up the whole infrastructure.

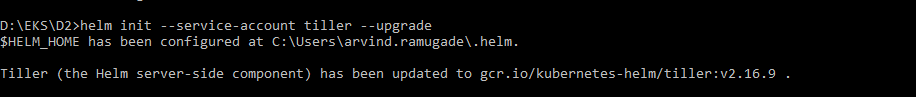
Initializing Helm



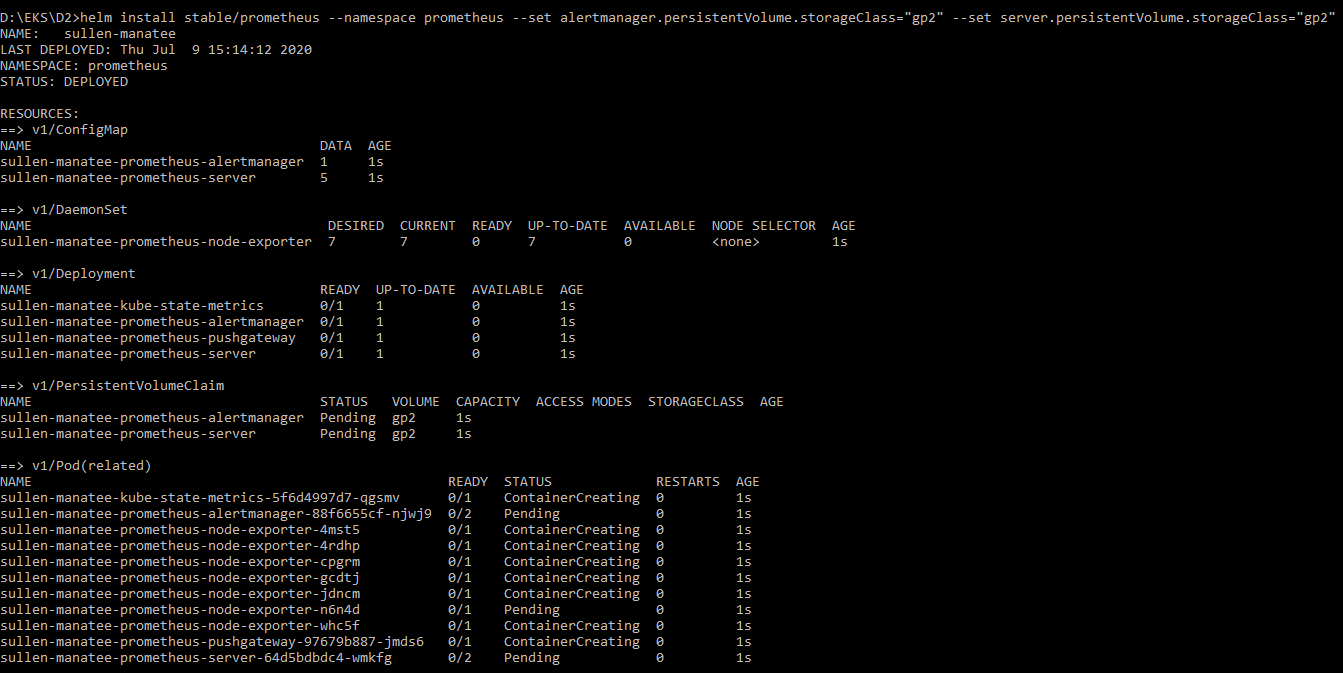
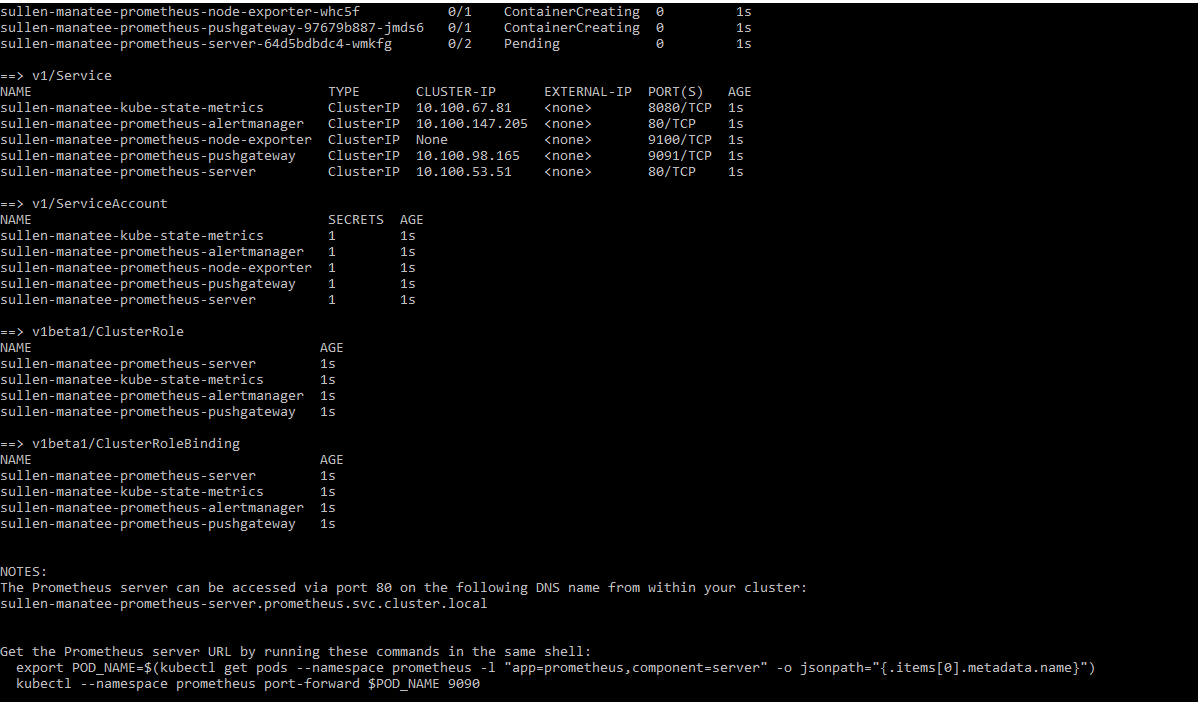


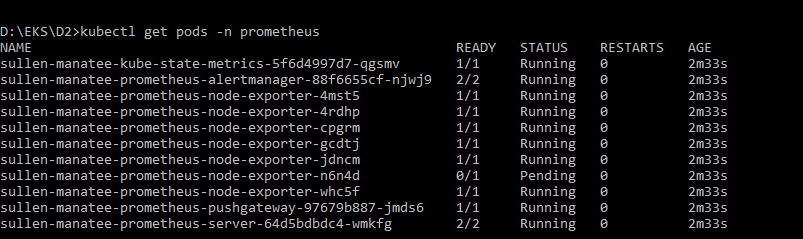
Creating Tiller Service



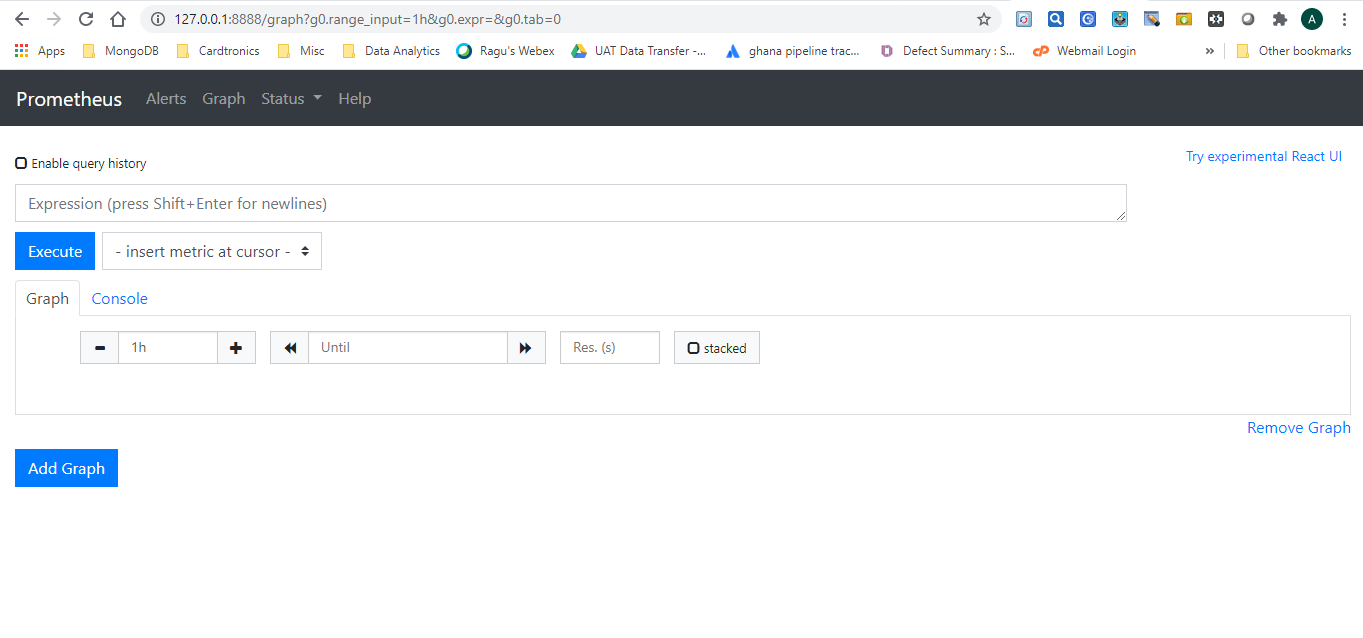


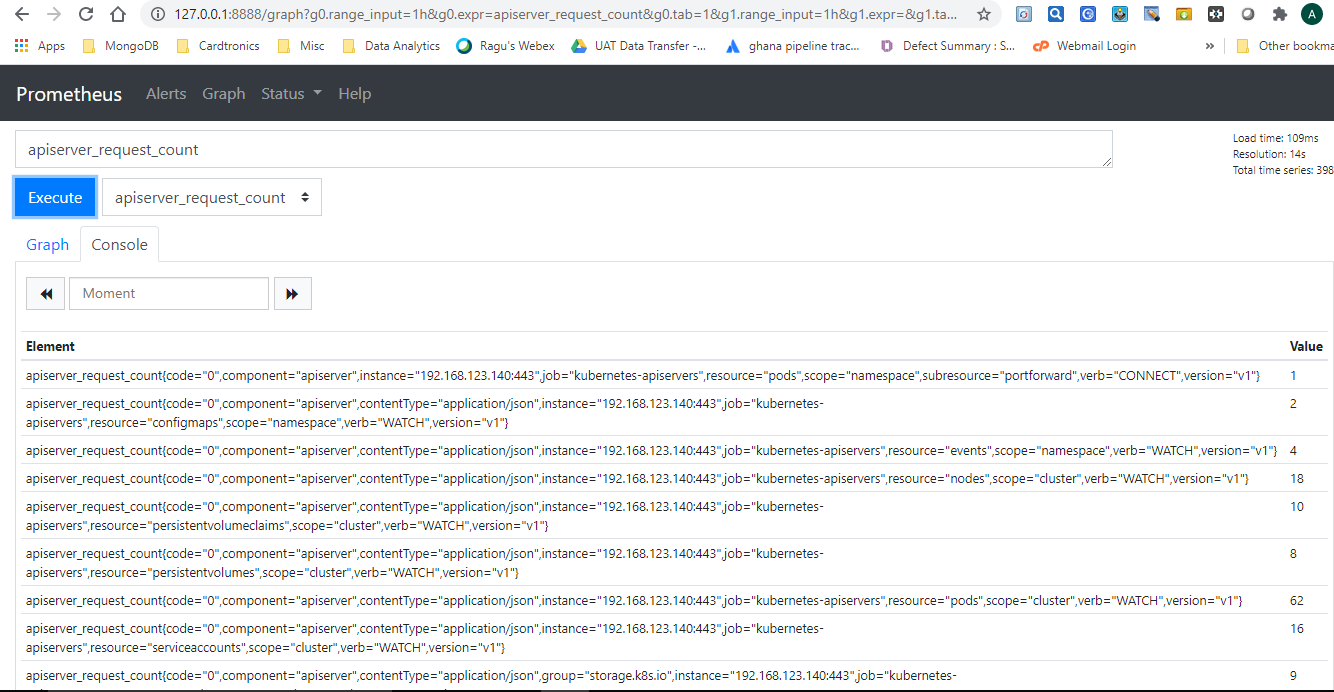
**Installation of Prometheus using helm**

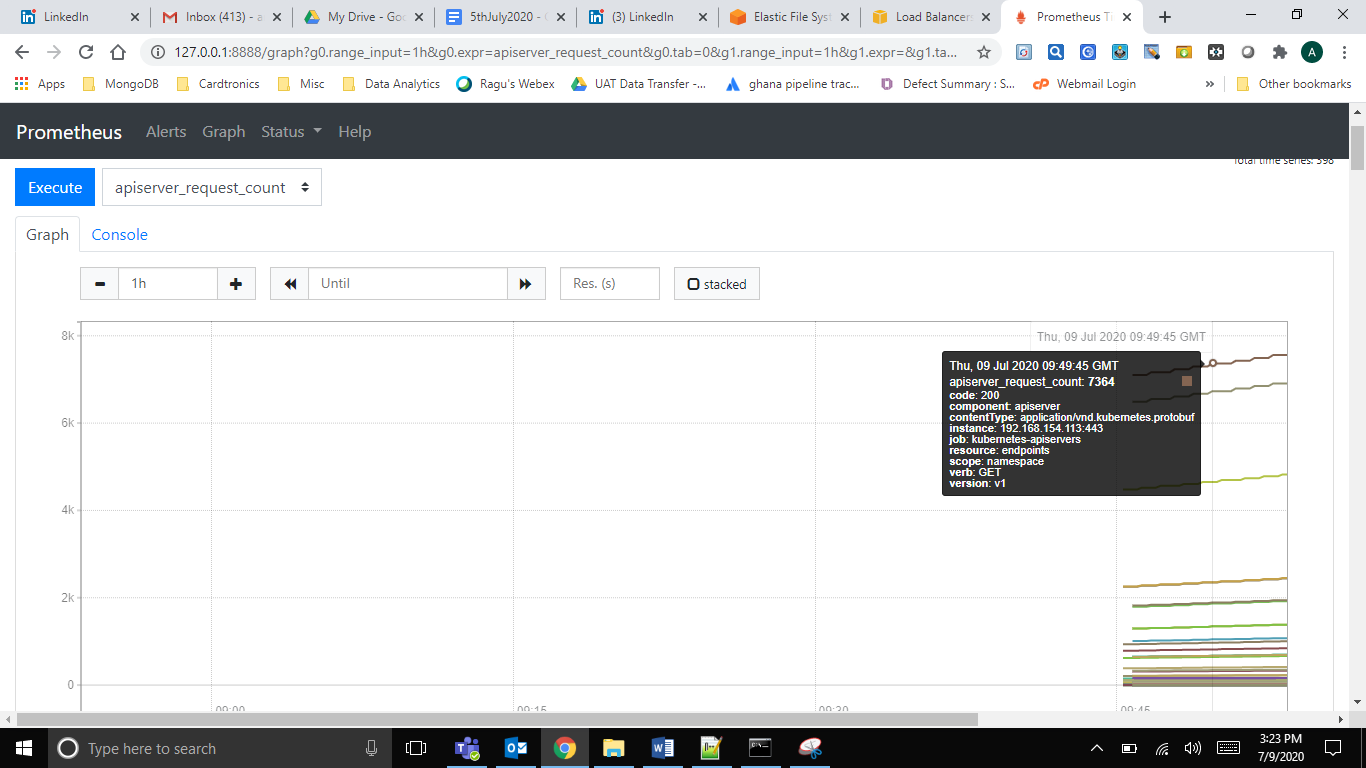
  

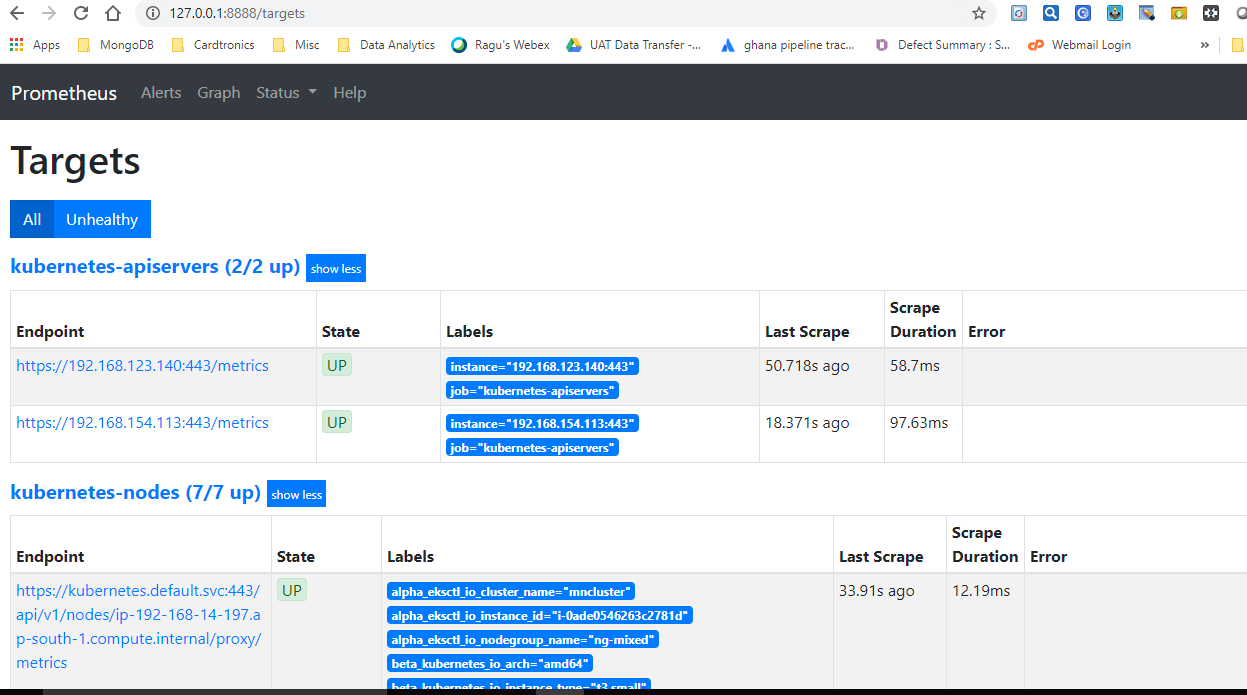


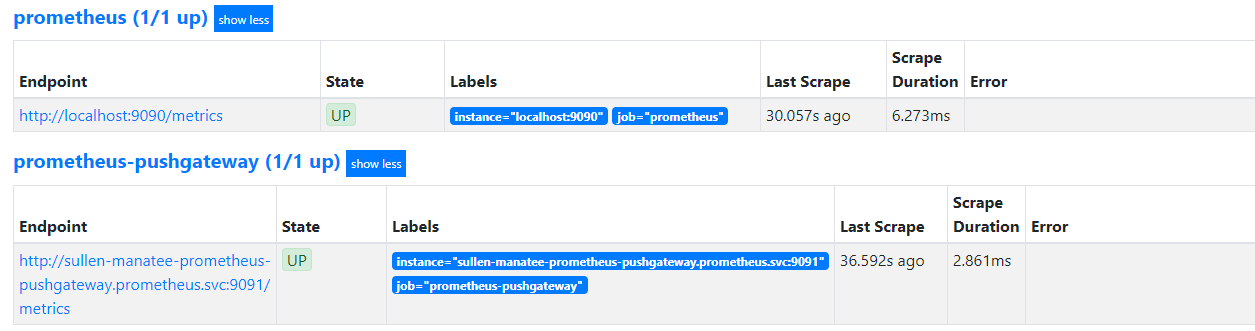




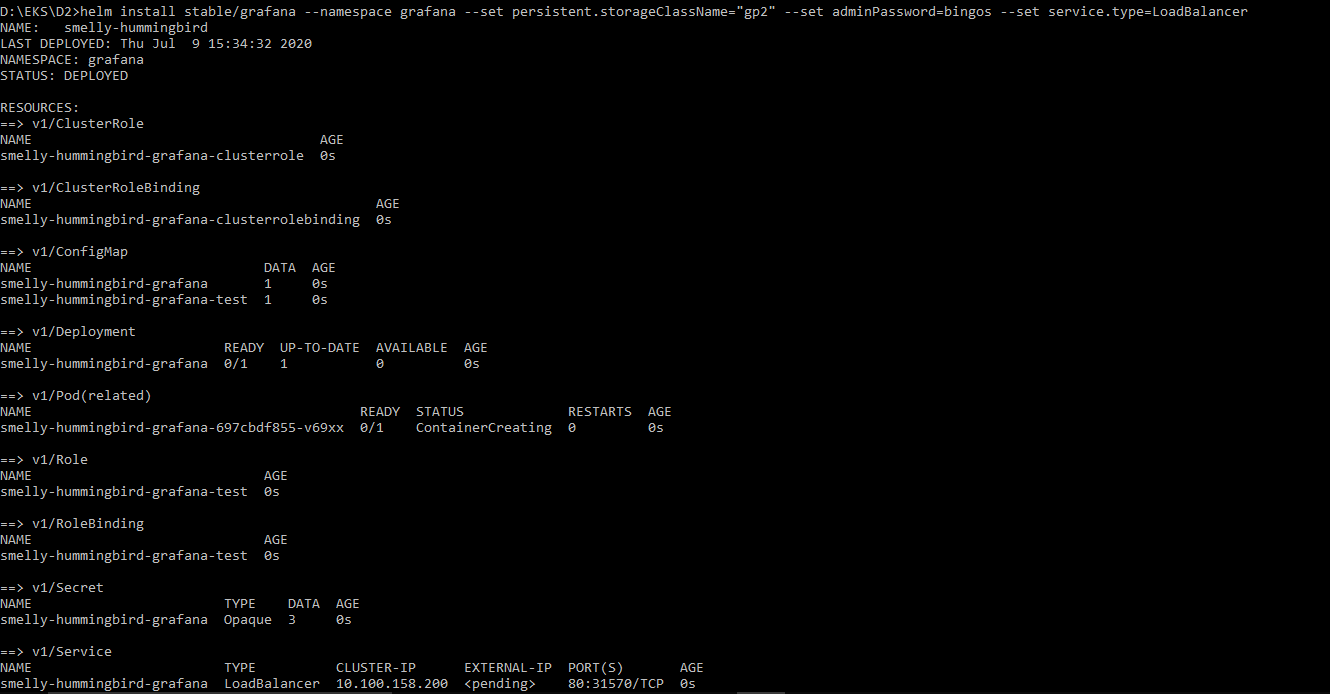


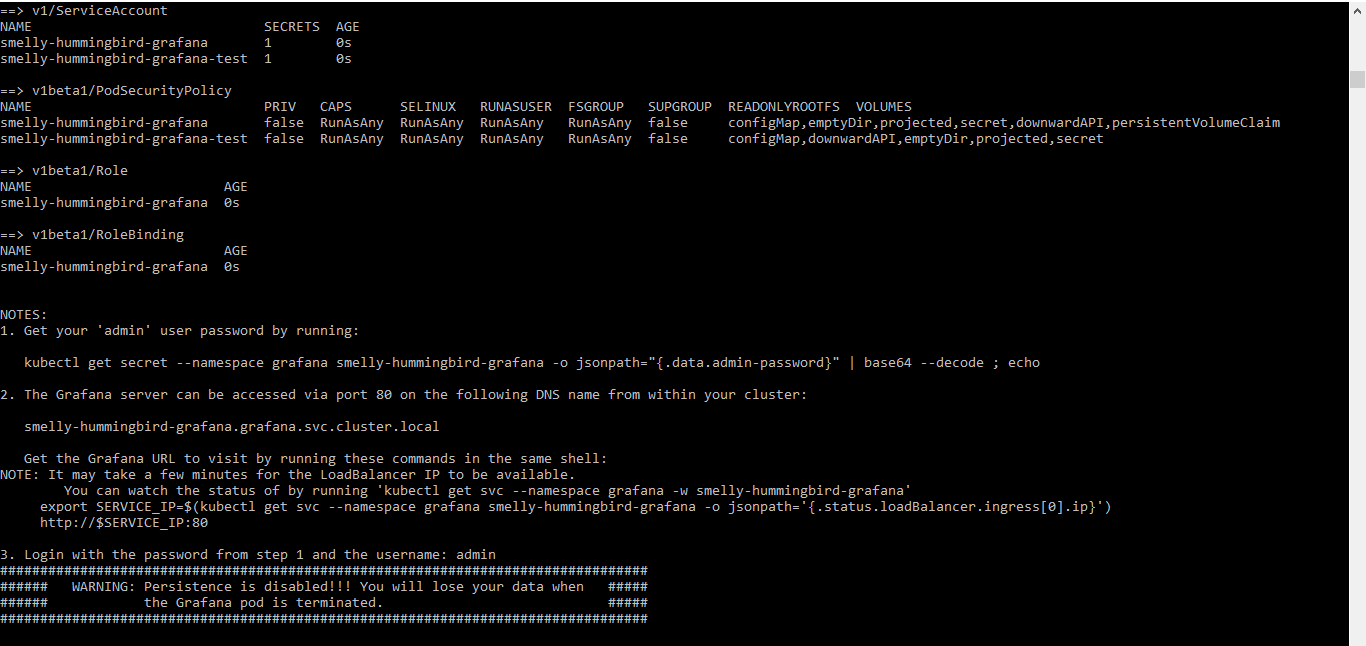


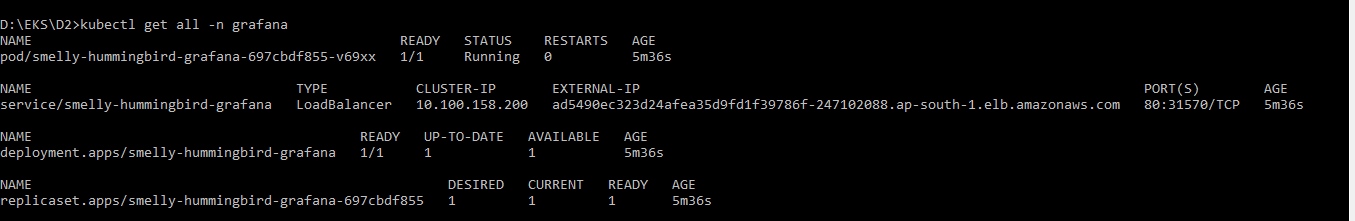


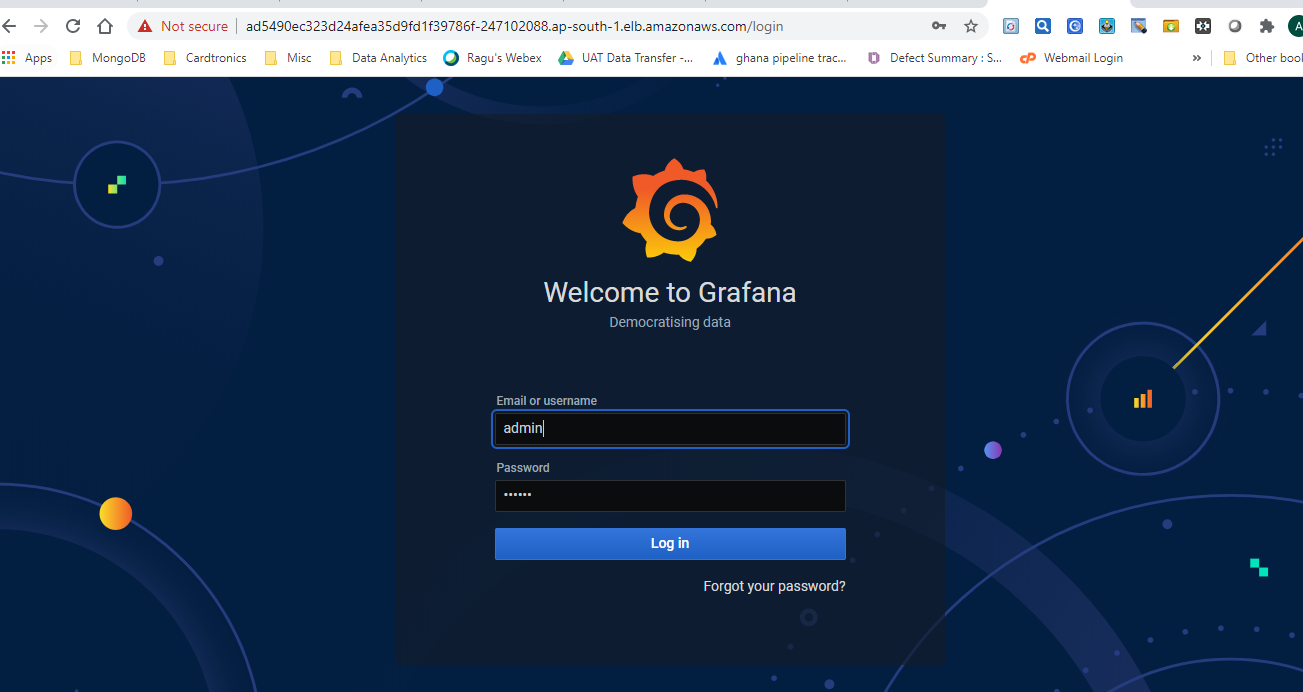


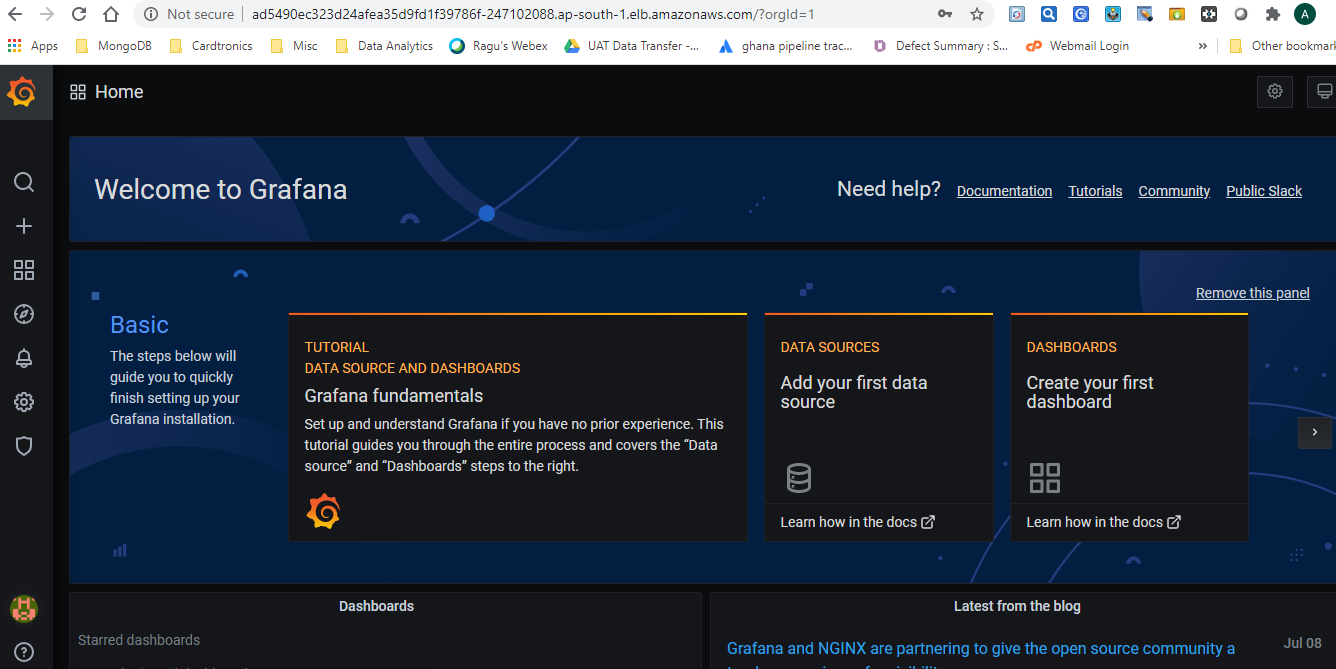
**Install Grafana.**



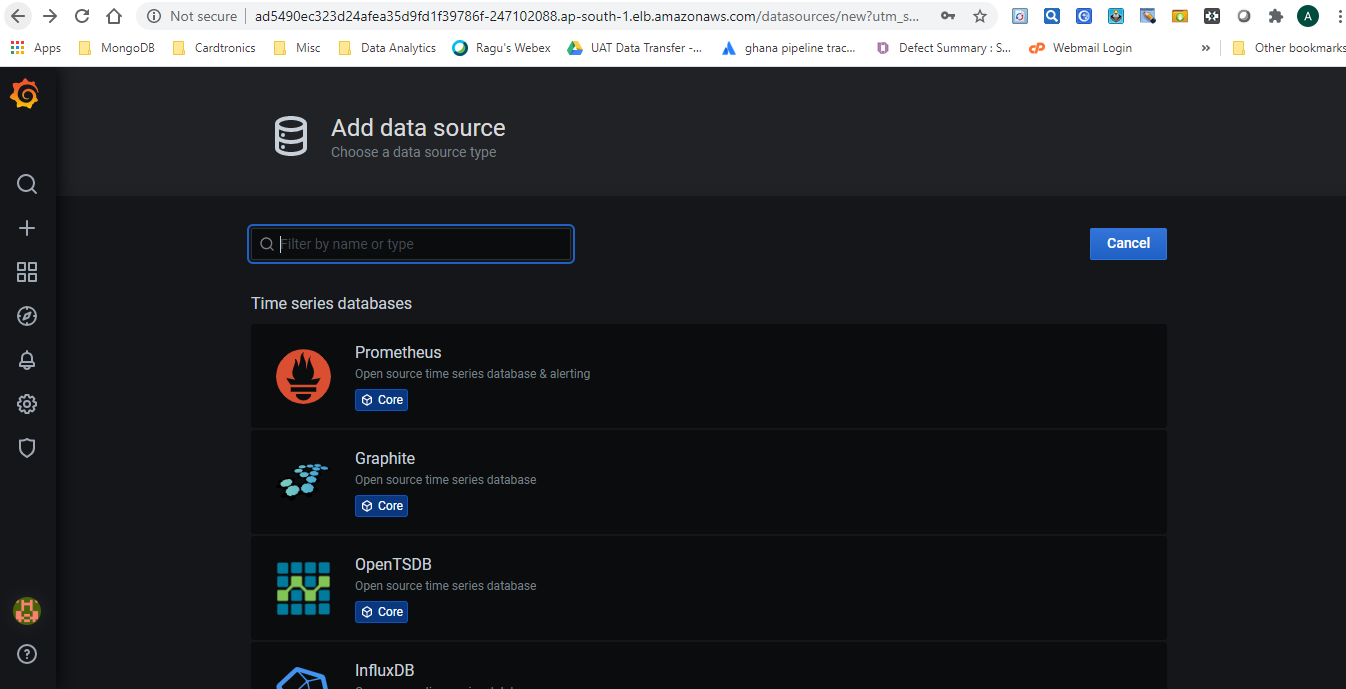


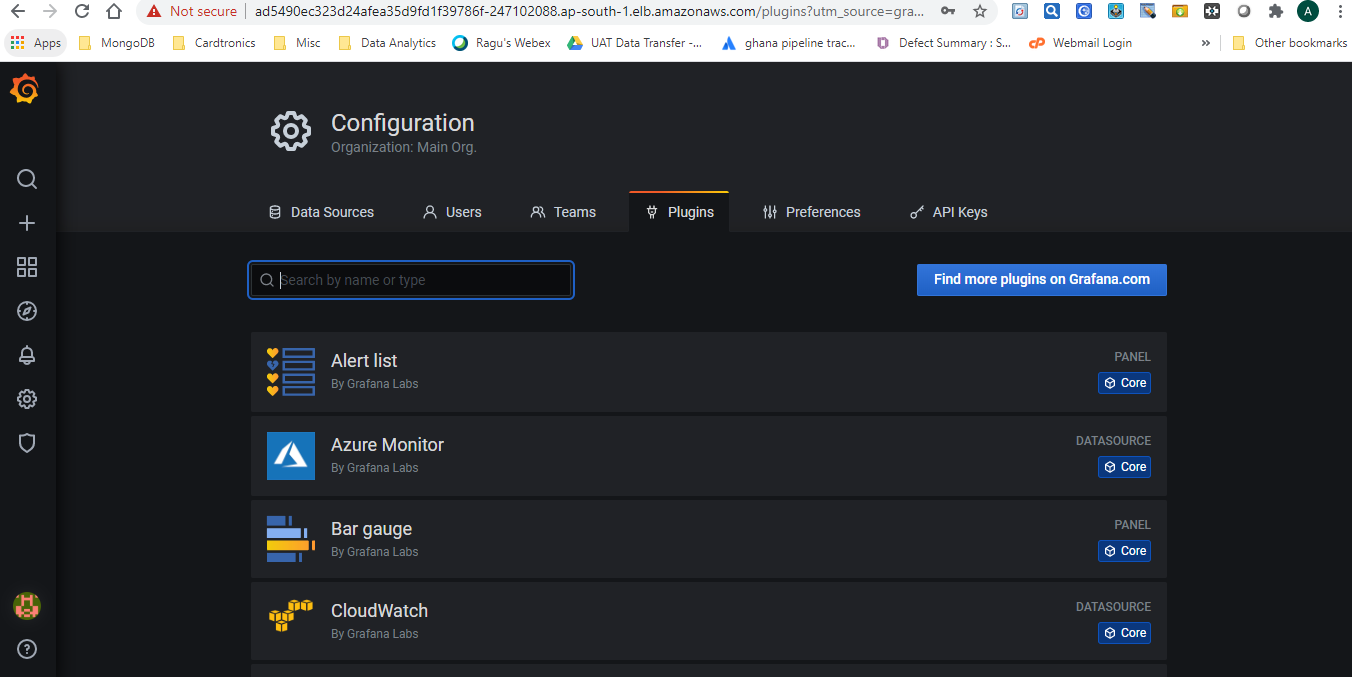


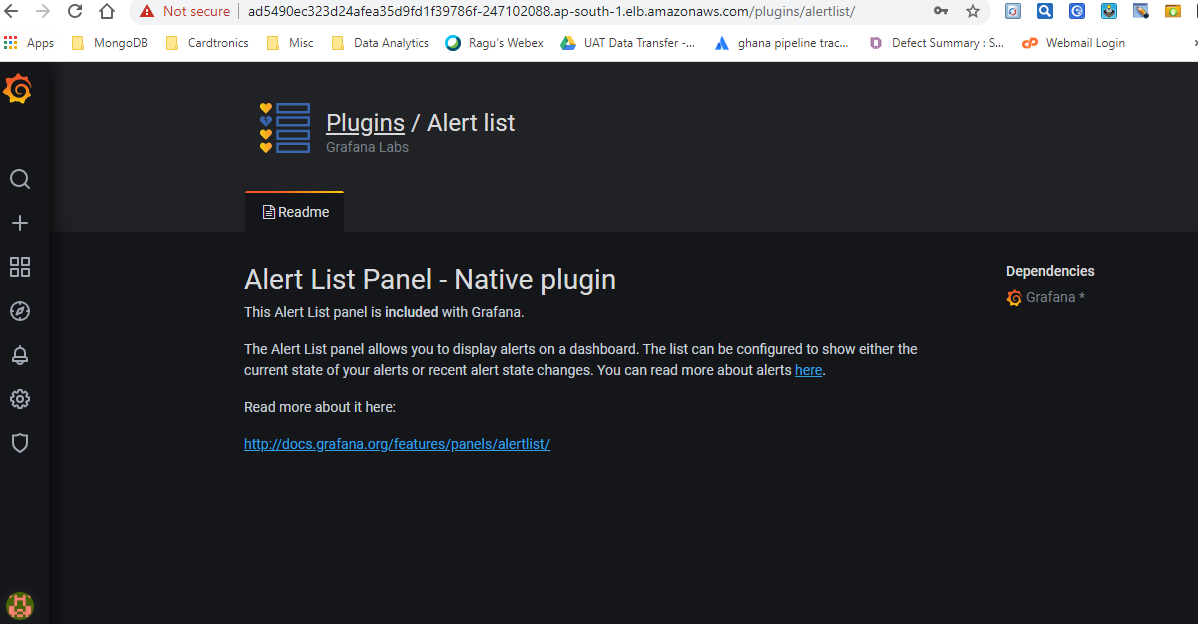




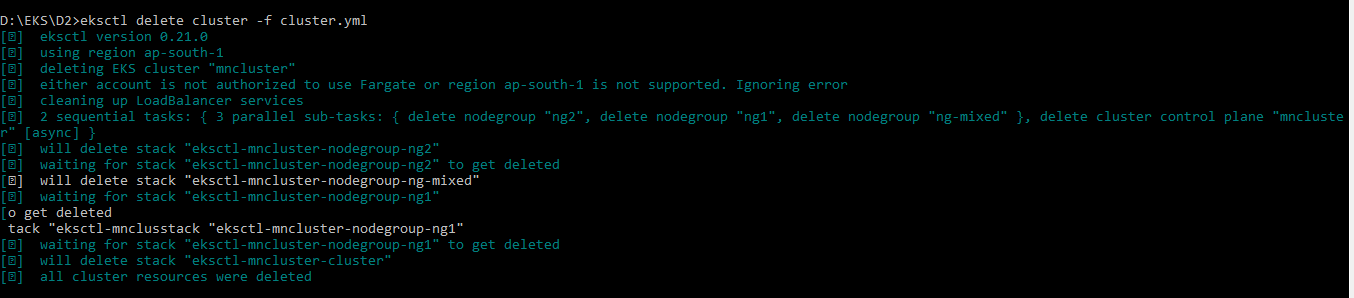
Select Prometheus Data Source

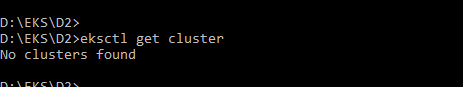




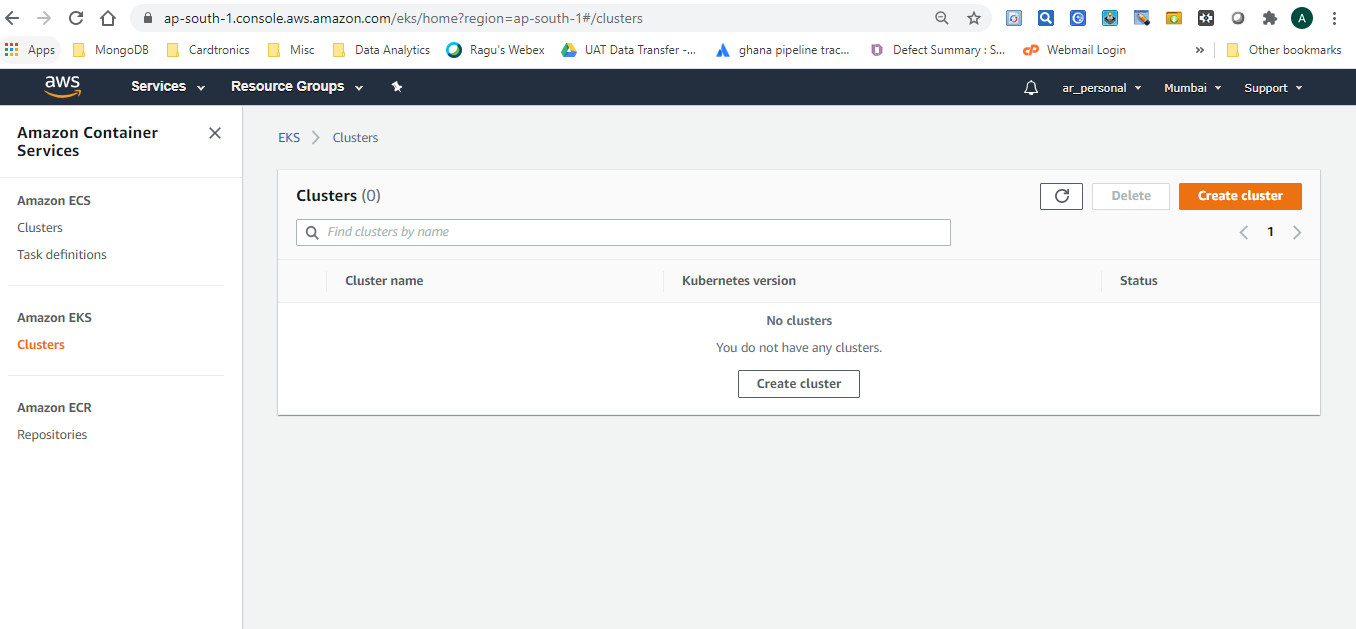


To delete entire EKS Cluster using below command

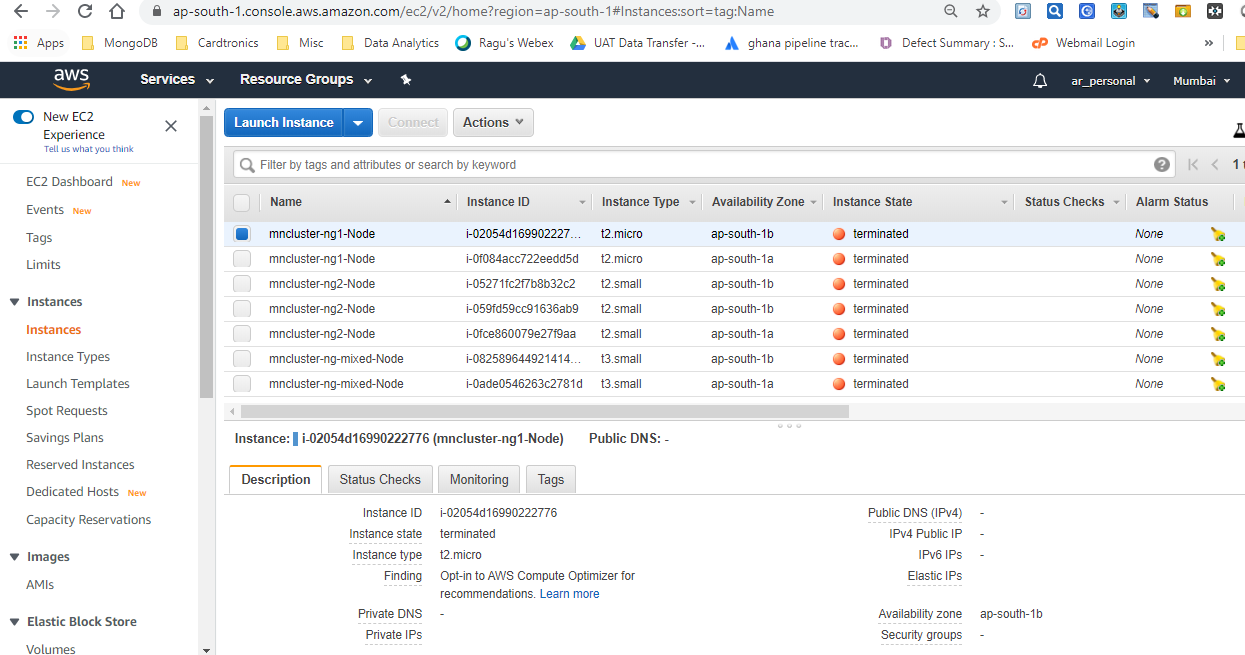




Verify deletion of cluster in AWS Console

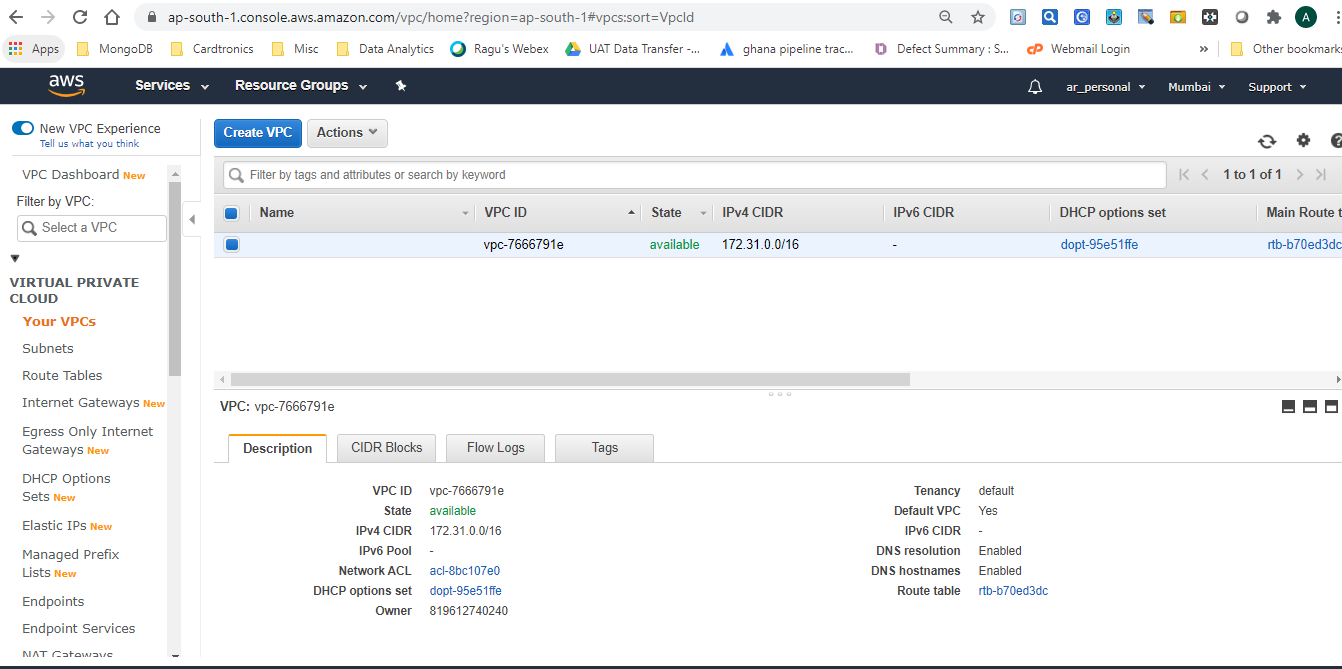


Verify that all EC2 instances are terminated.

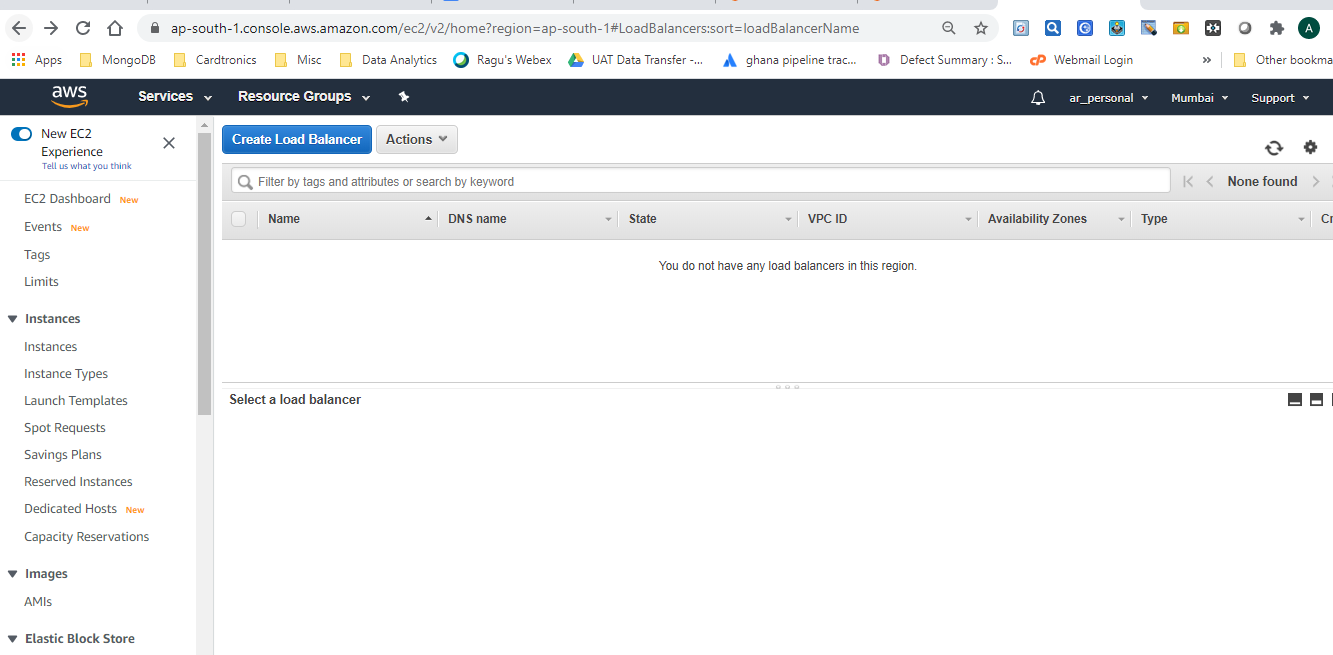


Verify that the VPC which was created by EKS Cluster also gets removed

Below is the default VPC which is created by AWS in a region.



Load Balancer also gets removed as part of cluster deletion.



We can remove EFS volume finally which is not required.

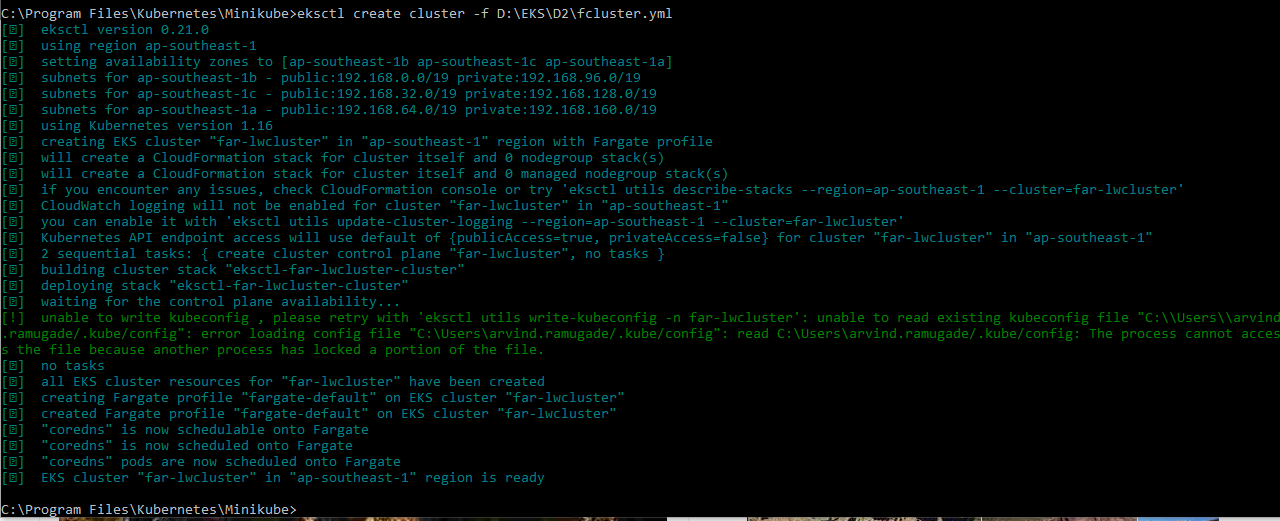


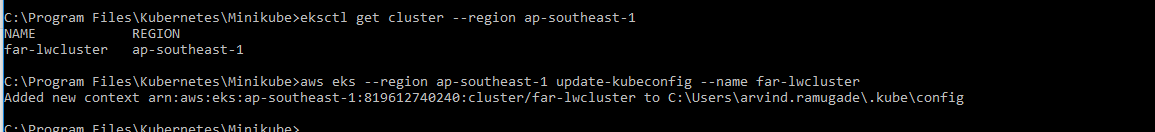
**AWS Fargate**

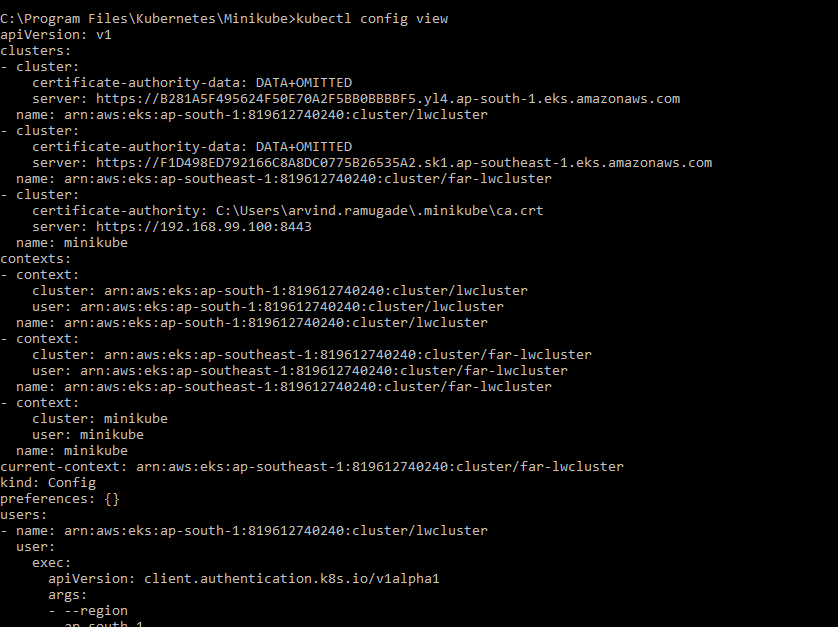
Fargate is a server-less architecture & it integrates with EKS.

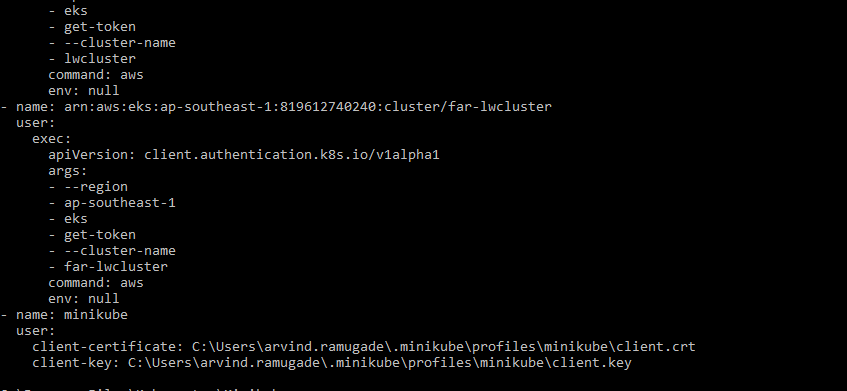
It's a black box which dynamically manages slave nodes without worrying about capacity such as CPU, RAM etc.

It launches slaves at run time & manages internally. We can't see those slaves in the AWS console though.











To delete Fargate cluster use below command.

