**#Day1​ -Elastic Kubernetes Services [EKS]**

EKS is a managed service also known as Kubernetes as a Service. Its managed by public cloud i.e. AWS.

EKS leverages below services of AWS:

EC2 (Elastic Compute Cloud), EBS (Elastic Block Storage), ELB (Elastic Load Balancer), EFS (Elastic File Service), VPC (Virtual Private Cloud), IAM (Identity & Access Management)

EKS can be created using WebUI/CLI/API -(Terraform)

CLI - there are 2 methods:

❖ 1) AWS EKS

❖ 2) eksctl

We need to install eksctl program first of all from following url

<https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html>

Using eksctl we can specify no. of worker nodes also the resources in worker nodes such as RAM,CPU etc.

Recommended method is eksctl using which we can create node groups & create a cluster.

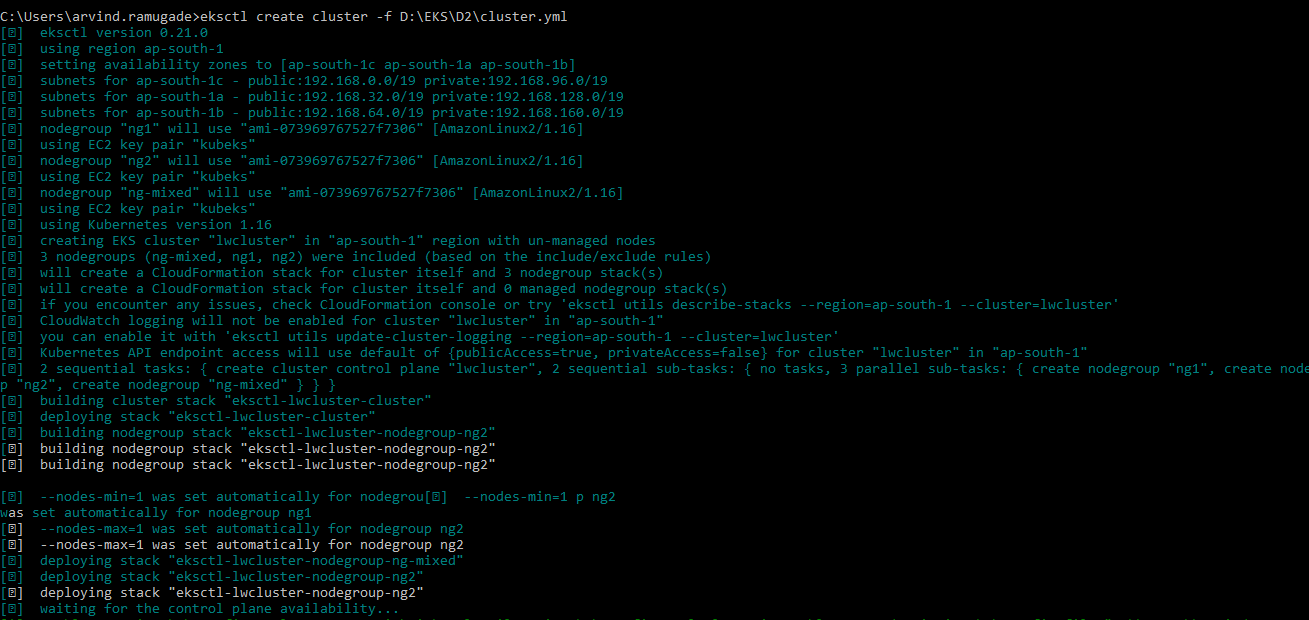
EKS uses cloud formation service in the background, which creates a stack of resources.

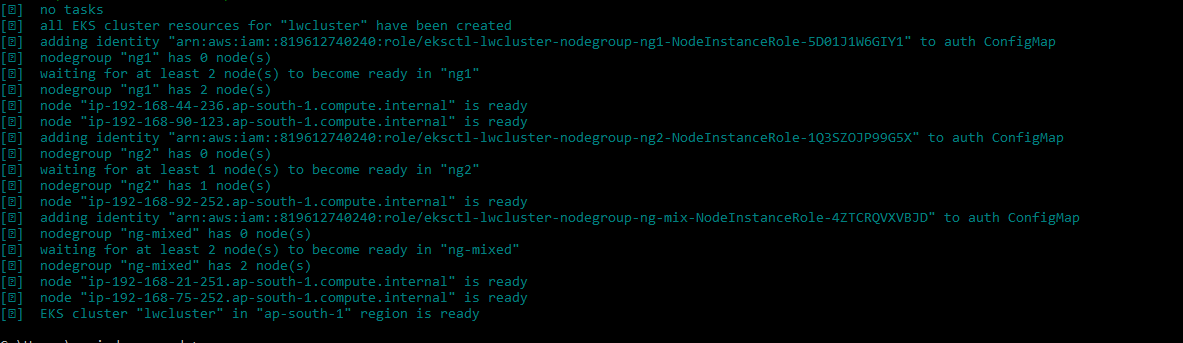
Below command used to create EKS cluster in Mumbai region (ap-south-1)

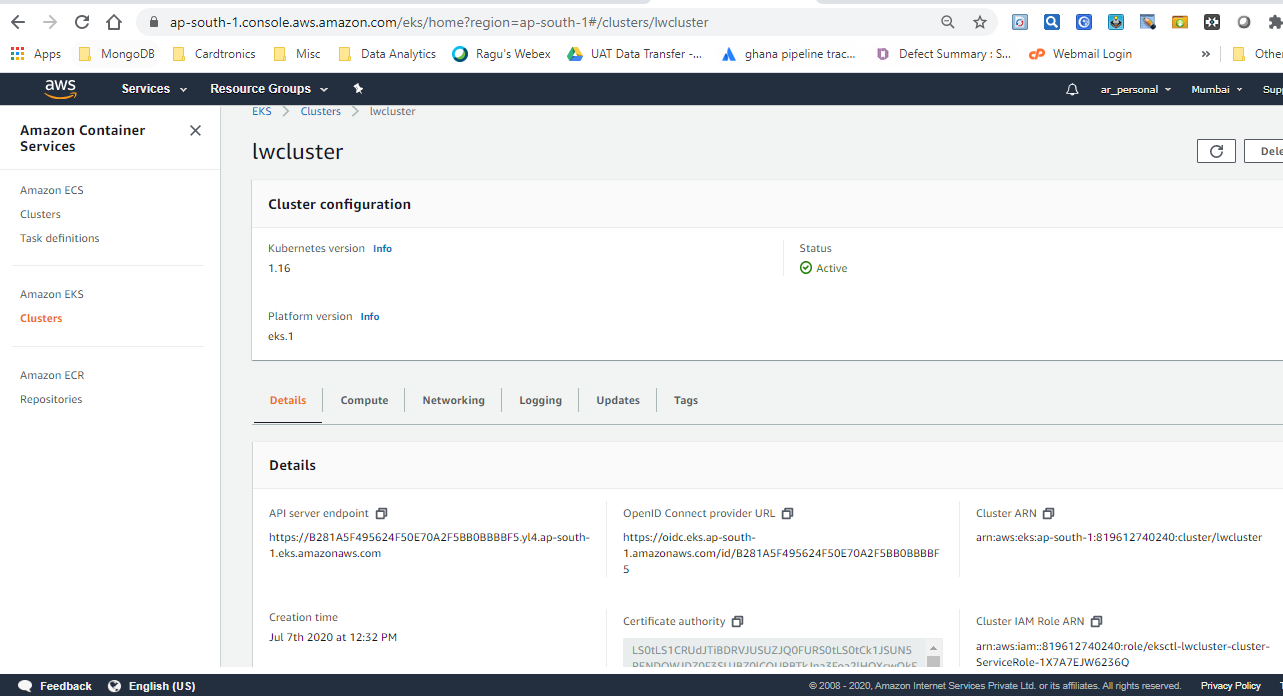
This will launch slave nodes with one master internally.

There will be 3 node groups such as ng1, ng2, ng-mixed all of which same AMI.

Also, VPC & Subnets will be created automatically.

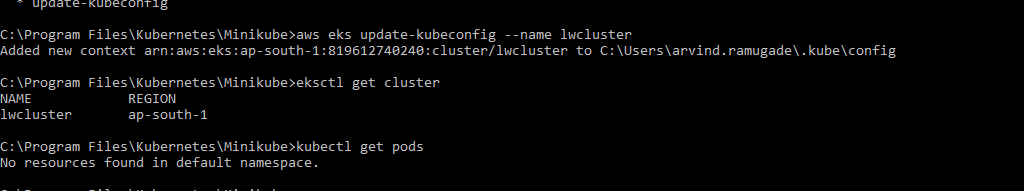






We can verify the cluster using below command.

We need to update kubeconfig file so as to use kubectl client command in the eks cluster



Kubernetes master node is critical & it has various services such as

❏ API Server

❏ Scheduler

❏ Controller

❏ etcd DB

**★ API Server**​ : which accepts images using kubectl client & send to scheduler​.

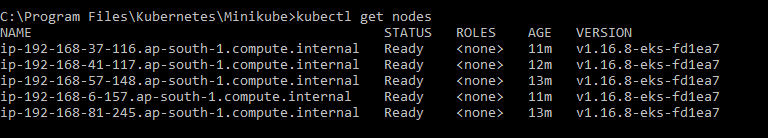
**★ Scheduler :**  contacts controller which in turn contact the kubelet program in the worker node  so as to ​launch the container using docker engine​.

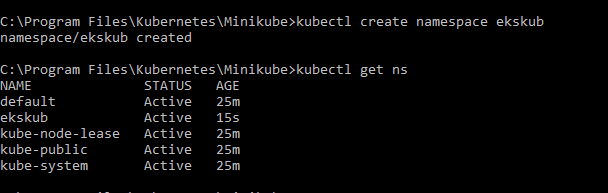
**★ Controller:** tracks all the nodes & works with scheduler to launch/terminate pods as part of the scaling process using replicas.

**★ Etcd**: Master has a DB known as etcd which stores metadata & config files.

The Kubelet program is required to be present on all the worker nodes.

Below CLI command will display all the nodes running in AWS.

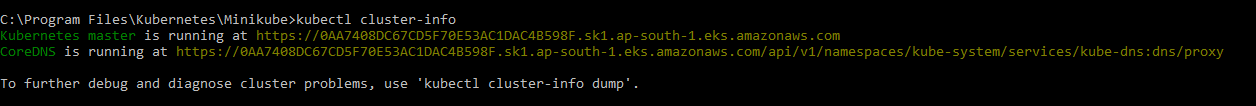


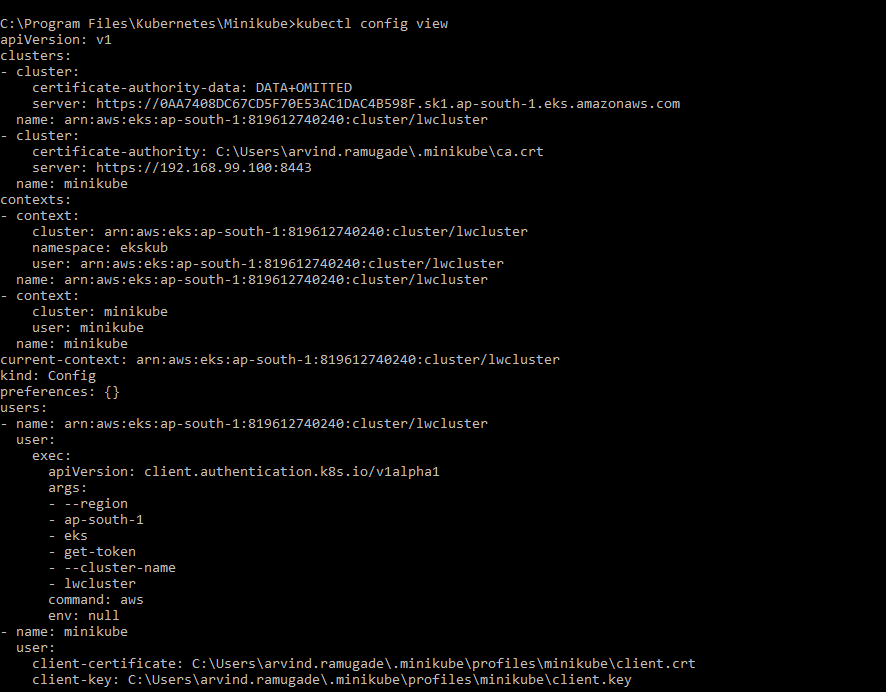


Kubectl uses default namespace. We can create our own namespace & associate its context with eks cluster using below command.

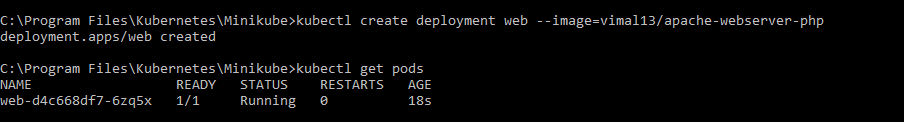


Use below command to get details of eks cluster

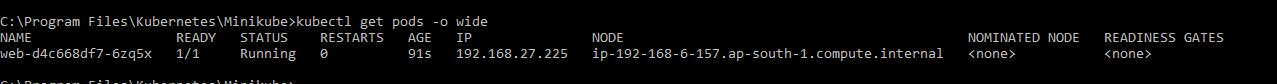


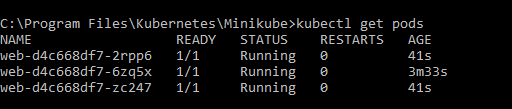


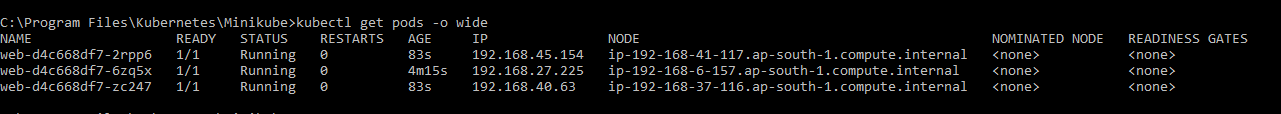
We can launch deployment inside the pod using below command



This command below shows information of pod along with its IP address.







EKS uses Load Balancer service to keep the cluster highly available.

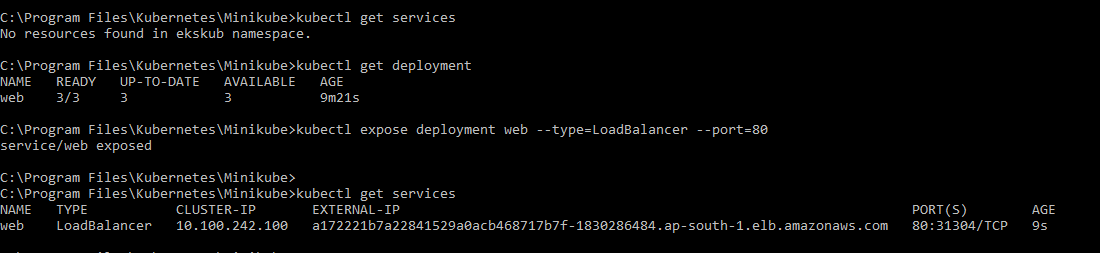
By default it uses classic load balancer which provides public IP to access pods & also provides load balancing. It distributes traffic among the pods such that load is evenly distributed for incoming requests.

Load balancer service in Kubernetes provides IP address to pods & also provides load balancing across all the pods.

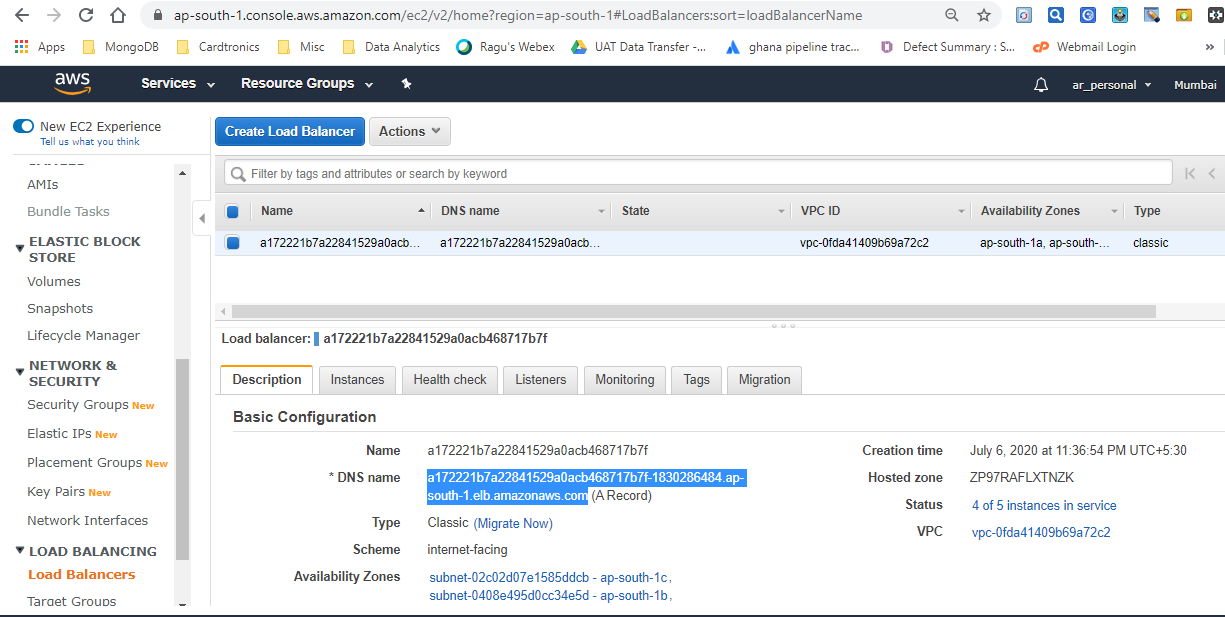
To expose the pod to outside world we can use below kubectl expose command.

Http requests listens on port 80. There are 3 types such as NodePort, LoadBalancer & ClusterIP.

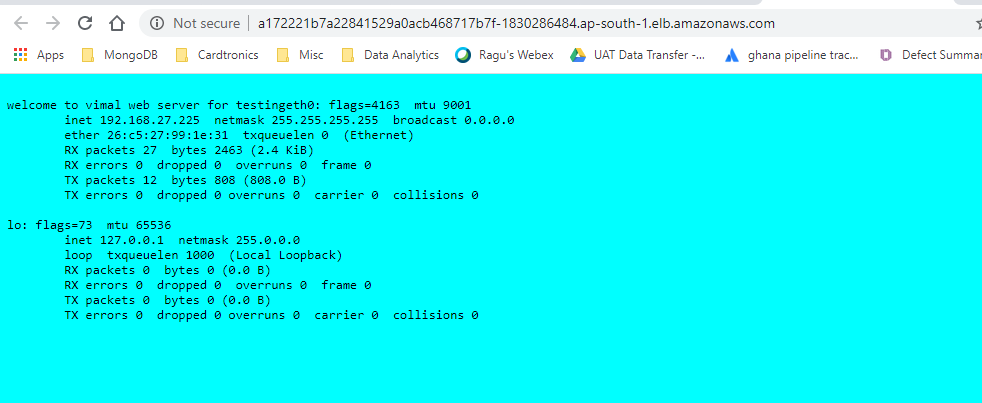
However, LoadBalancer type provides public IP to pods & provides load balancing. LoadBalancer capable of performing health of nodes & directs traffic only to healthy nodes.



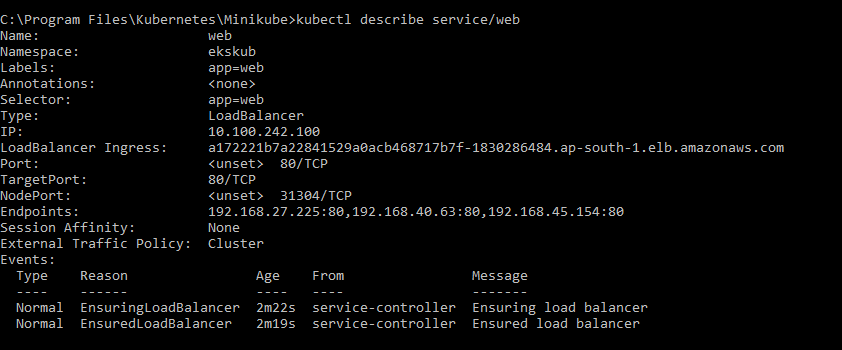
In AWS Console we can see the Load Balancer which is public facing.



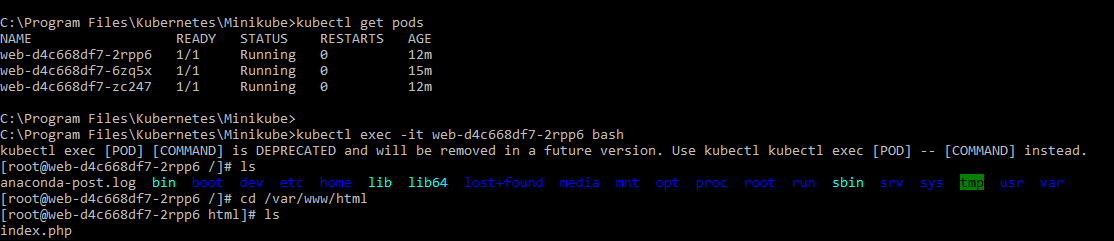
After clicking on DNS name following page gets displayed. These are the contents of /var/www/html/index.php



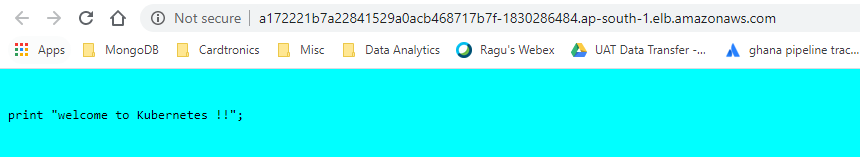
Below is the description of LoadBalancer.



Using below command we can log in to pod & copy code to /var/www/html/





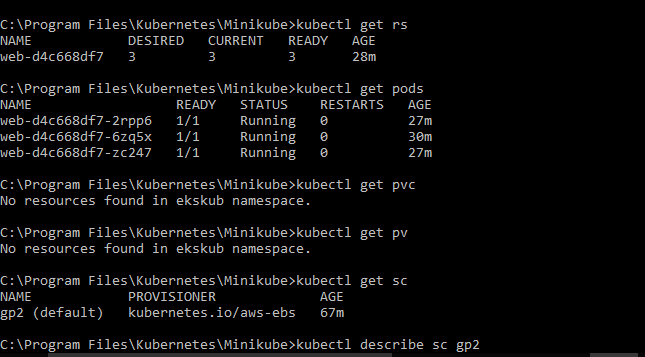


Persistent storage is required in order to preserve the data.

This is useful in scenarios where content of the web page is changed. To overcome this we can use EBS volume.

Pod requests PVC to claim volume. PVC in turn gets this volume from PV.

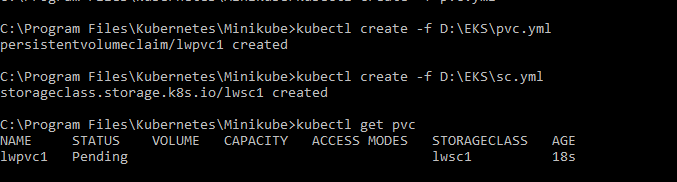
PV gets the volume from storage class provisioned by EBS.





By default the storage inside /var/www/html is ephemeral . If the pod goes down all the contents inside the folder will be lost. To avoid this we use PVC (Persistent Volume Claim).

PVC requests PV for storage. PV in turn provides storage internally from storage class of AWS Cloud. This ensures that the data is stored in PVC even if the Pod goes down/terminated.

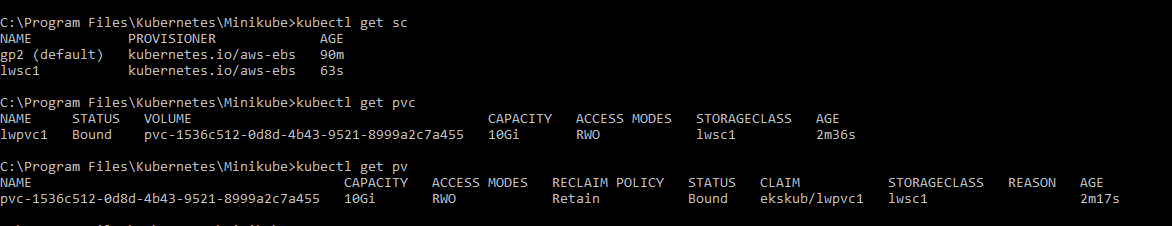


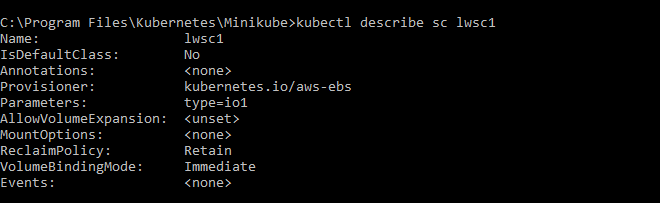
PVC status is in pending initially as we still need to provide mount path & storage volume which pod needs from PVC. This can be done by editing the pvc code & specifying these details.

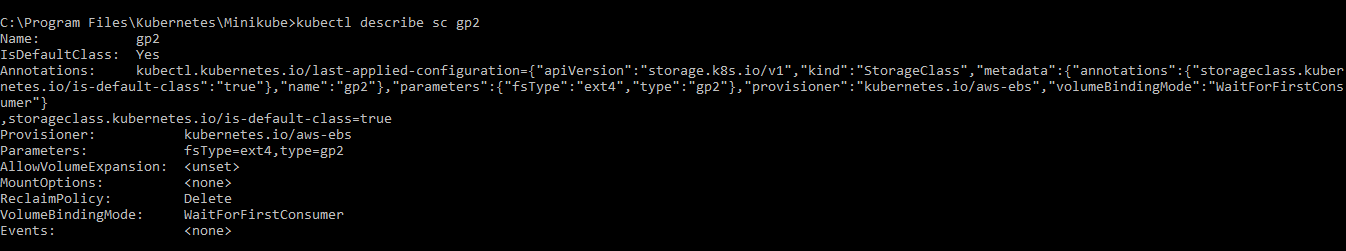
Default storage class is gp2.

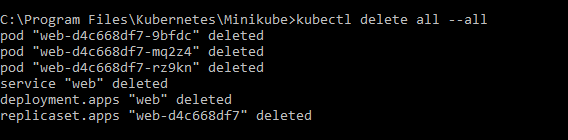
However, we can change it to Provisioned IOPS (io1) using annotations in the storage class description.

Using reclaim policy (delete/retain) we can either retain EBS volume or delete it.   When we delete the EKS Cluster, Cloud formation services also gets deleted



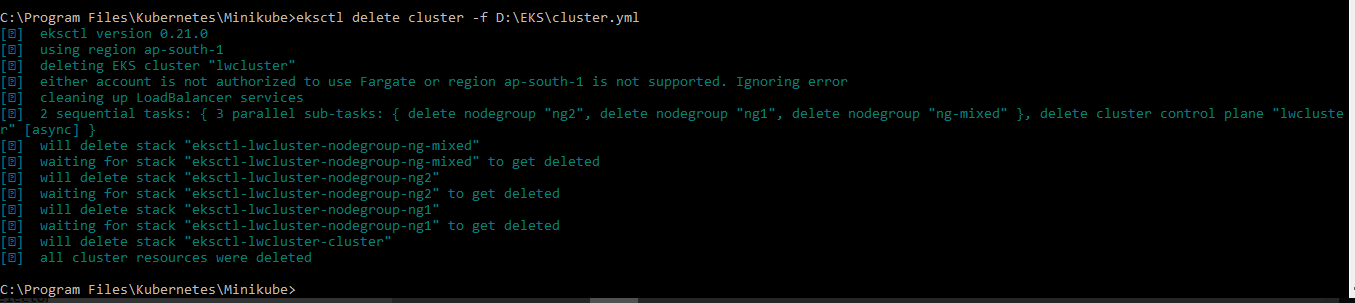


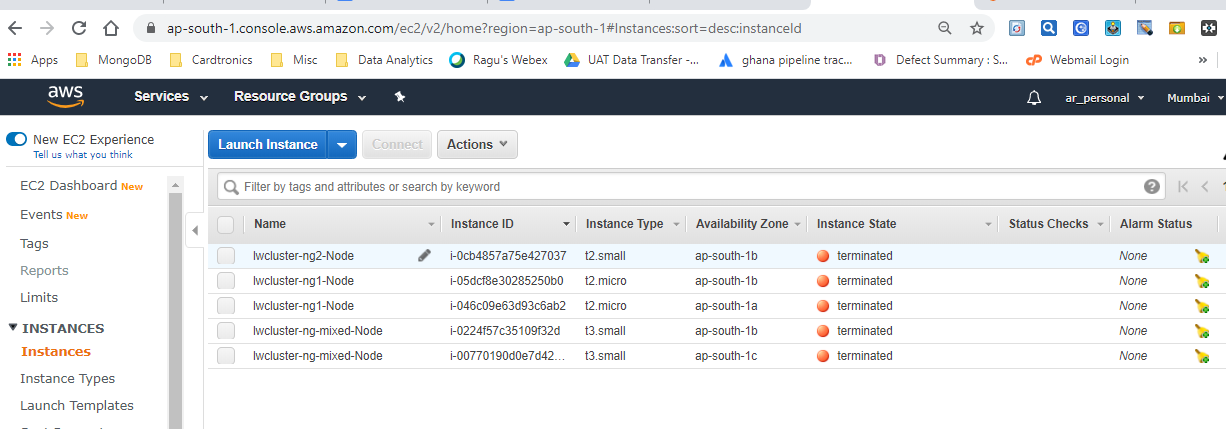


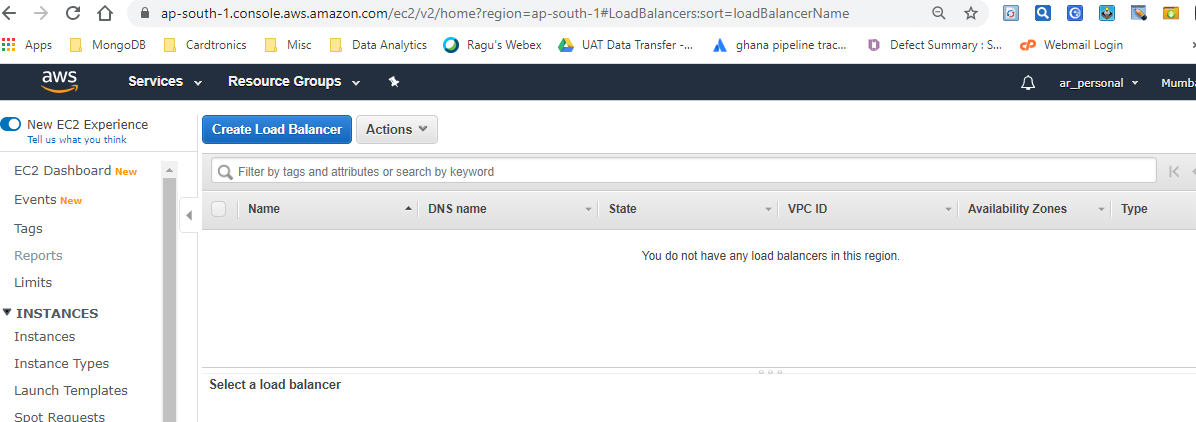


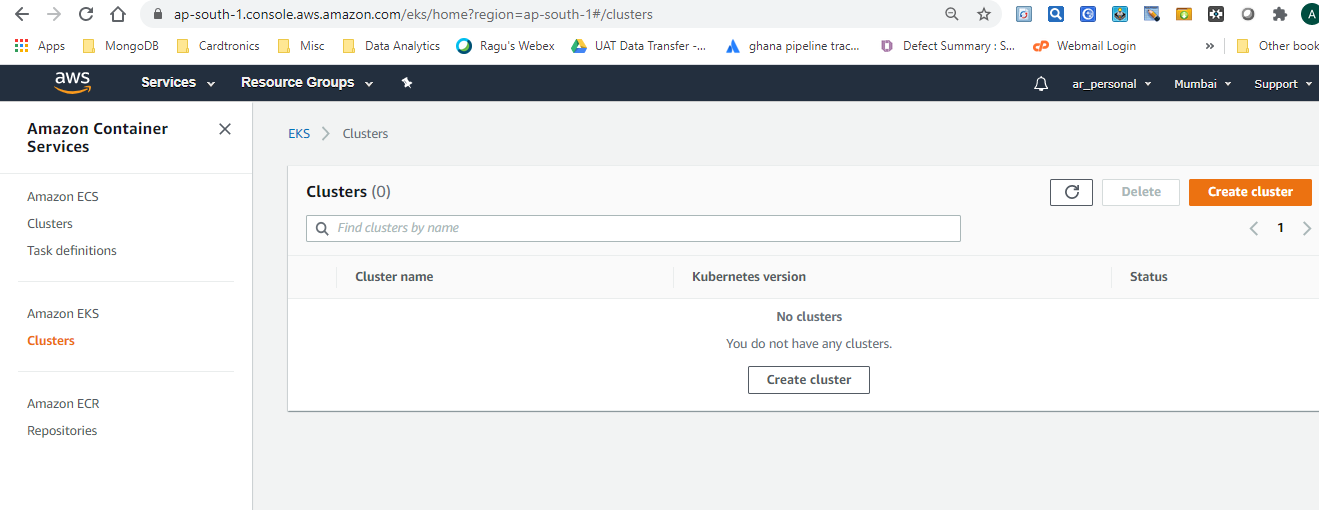
We can finally delete the eks cluster using below command

This will terminate all the instances, delete LoadBalancer as can be seen below.

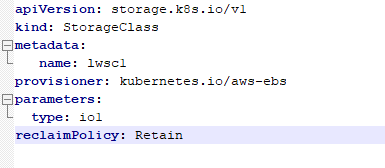


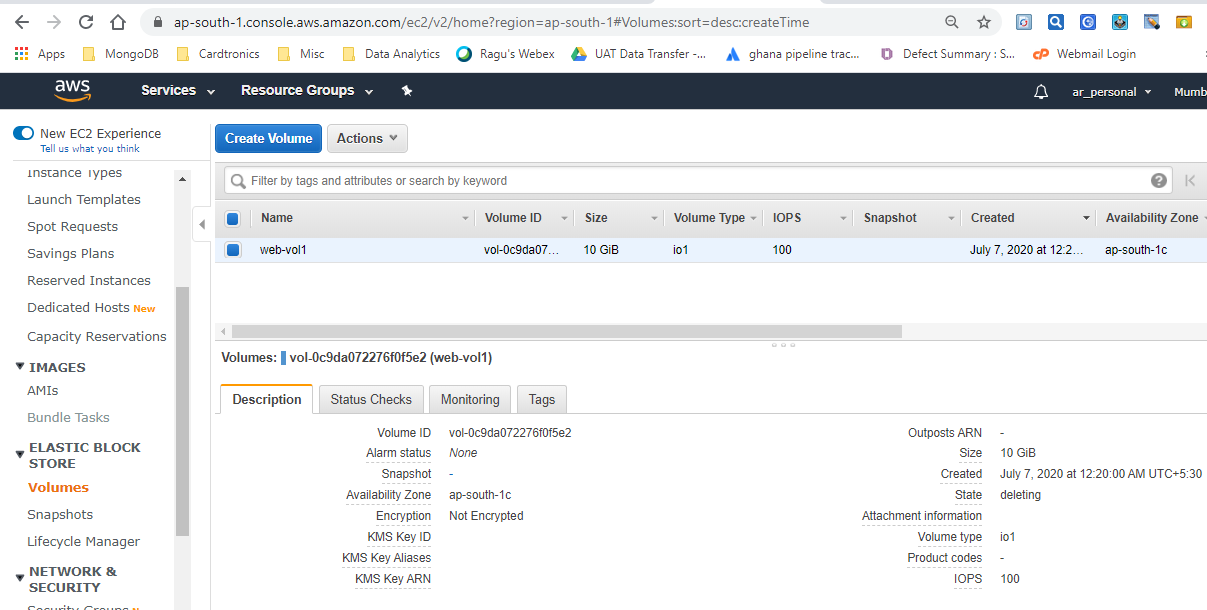


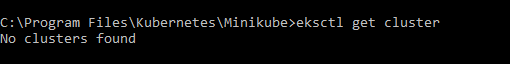




Since reclaim policy was set to Retain in sc.yml , EBS volume is not getting removed even after deleting the Kubernetes cluster. We have to remove the EBS volume manually as if we keep is as it is AWS will charge for the provisioned EBS volume.







***Summary***