#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.

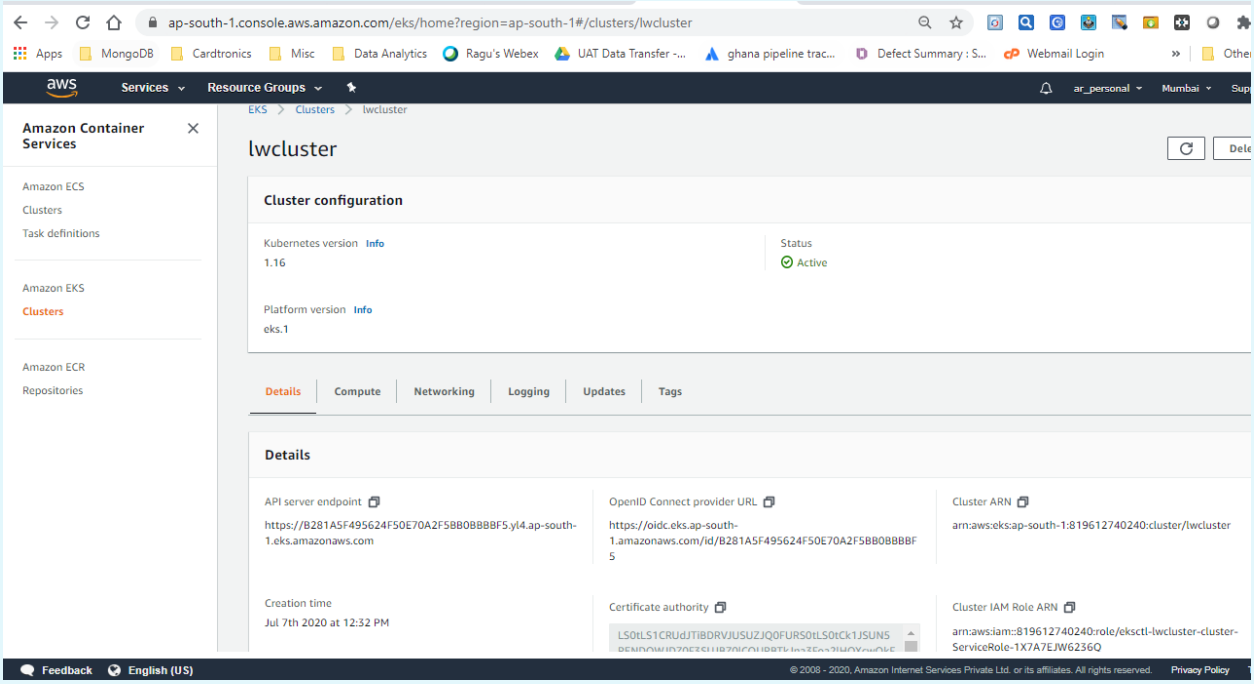
Below command used to create EKS cluster in Mumbai region (ap-south-1)
This will launch slave nodes with one master internally.

There will be 3 node groups such as ng1, ng2, ng-mixed all of which same AMI.
Also, VPC & Subnets will be created automatically.

```
C:\Users\arvind.ramugade>eksctl create cluster -f D:\EKS\D2\cluster.yml
[ℹ]  eksctl version 0.21.0
[ℹ]  using region ap-south-1
[ℹ]  setting availability zones to [ap-south-1c ap-south-1a ap-south-1b]
[ℹ]  subnets for ap-south-1c - public:192.168.0.0/19 private:192.168.96.0/19
[ℹ]  subnets for ap-south-1a - public:192.168.32.0/19 private:192.168.128.0/19
[ℹ]  subnets for ap-south-1b - public:192.168.64.0/19 private:192.168.160.0/19
[ℹ]  nodegroup "ng1" will use "ami-073969767527f7306" [AmazonLinux2/1.16]
[ℹ]  using EC2 key pair "kubeks"
[ℹ]  nodegroup "ng2" will use "ami-073969767527f7306" [AmazonLinux2/1.16]
[ℹ]  using EC2 key pair "kubeks"
[ℹ]  nodegroup "ng-mixed" will use "ami-073969767527f7306" [AmazonLinux2/1.16]
[ℹ]  using EC2 key pair "kubeks"
[ℹ]  using Kubernetes version 1.16
[ℹ]  creating EKS cluster "lwcluster" in "ap-south-1" region with un-managed nodes
[ℹ]  3 nodegroups (ng-mixed, ng1, ng2) were included (based on the include/exclude rules)
[ℹ]  will create a CloudFormation stack for cluster itself and 3 nodegroup stack(s)
[ℹ]  will create a CloudFormation stack for cluster itself and 0 managed nodegroup stack(s)
[ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=lwcluster'
[ℹ]  CloudWatch logging will not be enabled for cluster "lwcluster" in "ap-south-1"
[ℹ]  you can enable it with 'eksctl utils update-cluster-logging --region=ap-south-1 --cluster=lwcluster'
[ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "lwcluster" in "ap-south-1"
[ℹ]  2 sequential tasks: { create cluster control plane "lwcluster", 2 sequential sub-tasks: { no tasks, 3 parallel sub-tasks: { create nodegroup "ng1", create node
p "ng2", create nodegroup "ng-mixed" } } }
[ℹ]  building cluster stack "eksctl-lwcluster-cluster"
[ℹ]  deploying stack "eksctl-lwcluster-cluster"
[ℹ]  building nodegroup stack "eksctl-lwcluster-nodegroup-ng2"
[ℹ]  building nodegroup stack "eksctl-lwcluster-nodegroup-ng2"
[ℹ]  building nodegroup stack "eksctl-lwcluster-nodegroup-ng2"

[ℹ]  --nodes-min=1 was set automatically for nodegrou[ℹ]  --nodes-min=1 p ng2
was set automatically for nodegroup ng1
[ℹ]  --nodes-max=1 was set automatically for nodegroup ng2
[ℹ]  --nodes-max=1 was set automatically for nodegroup ng2
[ℹ]  deploying stack "eksctl-lwcluster-nodegroup-ng-mixed"
[ℹ]  deploying stack "eksctl-lwcluster-nodegroup-ng2"
[ℹ]  deploying stack "eksctl-lwcluster-nodegroup-ng2"
[ℹ]  waiting for the control plane availability...
```

```
[ℹ]  no tasks
[ℹ]  all EKS cluster resources for "lwcluster" have been created
[ℹ]  adding identity "arn:aws:iam::819612740240:role/eksctl-lwcluster-nodegroup-ng1-NodeInstanceRole-5D01J1W6GIY1" to auth ConfigMap
[ℹ]  nodegroup "ng1" has 0 node(s)
[ℹ]  waiting for at least 2 node(s) to become ready in "ng1"
[ℹ]  nodegroup "ng1" has 2 node(s)
[ℹ]  node "ip-192-168-44-236.ap-south-1.compute.internal" is ready
[ℹ]  node "ip-192-168-90-123.ap-south-1.compute.internal" is ready
[ℹ]  adding identity "arn:aws:iam::819612740240:role/eksctl-lwcluster-nodegroup-ng2-NodeInstanceRole-1Q3SZOJP99G5X" to auth ConfigMap
[ℹ]  nodegroup "ng2" has 0 node(s)
[ℹ]  waiting for at least 1 node(s) to become ready in "ng2"
[ℹ]  nodegroup "ng2" has 1 node(s)
[ℹ]  node "ip-192-168-92-252.ap-south-1.compute.internal" is ready
[ℹ]  adding identity "arn:aws:iam::819612740240:role/eksctl-lwcluster-nodegroup-ng-mix-NodeInstanceRole-4ZTCRQVXVBJD" to auth ConfigMap
[ℹ]  nodegroup "ng-mixed" has 0 node(s)
[ℹ]  waiting for at least 2 node(s) to become ready in "ng-mixed"
[ℹ]  nodegroup "ng-mixed" has 2 node(s)
[ℹ]  node "ip-192-168-21-251.ap-south-1.compute.internal" is ready
[ℹ]  node "ip-192-168-75-252.ap-south-1.compute.internal" is ready
[ℹ]  EKS cluster "lwcluster" in "ap-south-1" region is ready
```



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

```
C:\Program Files\Kubernetes\Minikube>aws eks update-kubeconfig --name lwcluster
Added new context arn:aws:eks:ap-south-1:819612740240:cluster/lwcluster to C:\Users\arvind.ramugade\.kube\config

C:\Program Files\Kubernetes\Minikube>eksctl get cluster
NAME            REGION
lwcluster       ap-south-1

C:\Program Files\Kubernetes\Minikube>kubectl get pods
No resources found in default namespace.
```

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.

```
C:\Program Files\Kubernetes\Minikube>kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
ip-192-168-37-116.ap-south-1.compute.internal   Ready    <none>   11m   v1.16.8-eks-fd1ea7
ip-192-168-41-117.ap-south-1.compute.internal   Ready    <none>   12m   v1.16.8-eks-fd1ea7
ip-192-168-57-148.ap-south-1.compute.internal   Ready    <none>   13m   v1.16.8-eks-fd1ea7
ip-192-168-6-157.ap-south-1.compute.internal    Ready    <none>   11m   v1.16.8-eks-fd1ea7
ip-192-168-81-245.ap-south-1.compute.internal   Ready    <none>   13m   v1.16.8-eks-fd1ea7
```

```
C:\Program Files\Kubernetes\Minikube>kubectl create namespace ekskub
namespace/ekskub created

C:\Program Files\Kubernetes\Minikube>kubectl get ns
NAME              STATUS   AGE
default           Active   25m
ekskub            Active   15s
kube-node-lease   Active   25m
kube-public       Active   25m
kube-system       Active   25m
```

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

```
C:\Program Files\Kubernetes\Minikube>kubectl config set-context --current --namespace=ekskub
Context "arn:aws:eks:ap-south-1:819612740240:cluster/lwcluster" modified.
```

Use below command to get details of eks cluster

```
C:\Program Files\Kubernetes\Minikube>kubectl cluster-info
Kubernetes master is running at https://0AA7408DC67CD5F70E53AC1DAC4B598F.sk1.ap-south-1.eks.amazonaws.com
CoreDNS is running at https://0AA7408DC67CD5F70E53AC1DAC4B598F.sk1.ap-south-1.eks.amazonaws.com/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

```
C:\Program Files\Kubernetes\Minikube>kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://0AA7408DC67CD5F70E53AC1DAC4B598F.sk1.ap-south-1.eks.amazonaws.com
  name: arn:aws:eks:ap-south-1:819612740240:cluster/lwcluster
- cluster:
    certificate-authority: C:\Users\arvind.ramugade\.minikube\ca.crt
    server: https://192.168.99.100:8443
  name: minikube
contexts:
- context:
    cluster: arn:aws:eks:ap-south-1:819612740240:cluster/lwcluster
    namespace: ekskub
    user: arn:aws:eks:ap-south-1:819612740240:cluster/lwcluster
  name: arn:aws:eks:ap-south-1:819612740240:cluster/lwcluster
- context:
    cluster: minikube
    user: minikube
  name: minikube
current-context: arn:aws:eks:ap-south-1:819612740240:cluster/lwcluster
kind: Config
preferences: {}
users:
- name: arn:aws:eks:ap-south-1:819612740240:cluster/lwcluster
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      args:
      - --region
      - ap-south-1
      - eks
      - get-token
      - --cluster-name
      - lwcluster
      command: aws
      env: null
- name: minikube
  user:
    client-certificate: C:\Users\arvind.ramugade\.minikube\profiles\minikube\client.crt
    client-key: C:\Users\arvind.ramugade\.minikube\profiles\minikube\client.key
```

We can launch deployment inside the pod using below command

```
C:\Program Files\Kubernetes\Minikube>kubectl create deployment web --image=vimal13/apache-webserver-php
deployment.apps/web created

C:\Program Files\Kubernetes\Minikube>kubectl get pods
NAME                   READY   STATUS    RESTARTS   AGE
web-d4c668df7-6zq5x    1/1     Running   0          18s
```

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

```
C:\Program Files\Kubernetes\Minikube>kubectl get pods -o wide
NAME                   READY   STATUS    RESTARTS   AGE   IP              NODE                                        NOMINATED NODE   READINESS GATES
web-d4c668df7-6zq5x    1/1     Running   0          91s   192.168.27.225  ip-192-168-6-157.ap-south-1.compute.internal   <none>           <none>
```

```
C:\Program Files\Kubernetes\Minikube>kubectl get pods
NAME                   READY   STATUS    RESTARTS   AGE
web-d4c668df7-2rpp6    1/1     Running   0          41s
web-d4c668df7-6zq5x    1/1     Running   0          3m33s
web-d4c668df7-zc247    1/1     Running   0          41s
```

```
C:\Program Files\Kubernetes\Minikube>kubectl get pods -o wide
NAME                   READY   STATUS    RESTARTS   AGE     IP               NODE                                          NOMINATED NODE   READINESS GATES
web-d4c668df7-2rpp6    1/1     Running   0          83s     192.168.45.154   ip-192-168-41-117.ap-south-1.compute.internal   <none>           <none>
web-d4c668df7-6zq5x    1/1     Running   0          4m15s   192.168.27.225   ip-192-168-6-157.ap-south-1.compute.internal    <none>           <none>
web-d4c668df7-zc247    1/1     Running   0          83s     192.168.40.63    ip-192-168-37-116.ap-south-1.compute.internal   <none>           <none>
```

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.

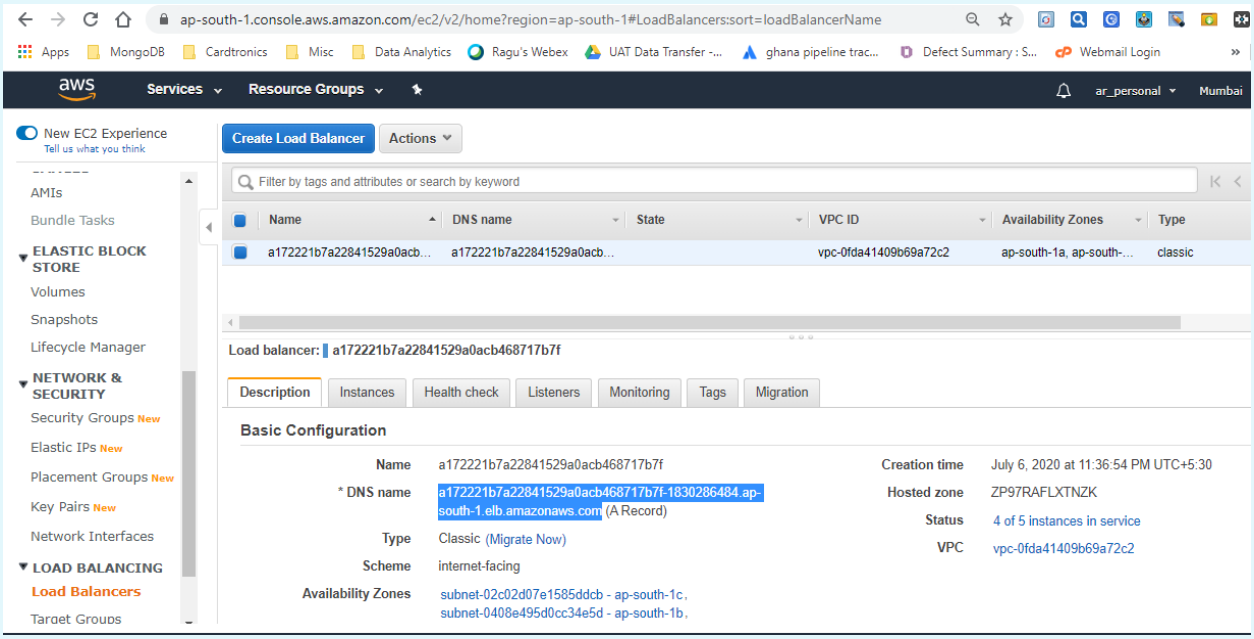

```
C:\Program Files\Kubernetes\Minikube>kubectl get services
No resources found in ekskub namespace.

C:\Program Files\Kubernetes\Minikube>kubectl get deployment
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
web    3/3     3            3           9m21s

C:\Program Files\Kubernetes\Minikube>kubectl expose deployment web --type=LoadBalancer --port=80
service/web exposed

C:\Program Files\Kubernetes\Minikube>
C:\Program Files\Kubernetes\Minikube>kubectl get services
NAME   TYPE           CLUSTER-IP       EXTERNAL-IP                                                                   PORT(S)        AGE
web    LoadBalancer   10.100.242.100   a172221b7a22841529a0acb468717b7f-1830286484.ap-south-1.elb.amazonaws.com   80:31304/TCP   9s
```

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.

```
C:\Program Files\Kubernetes\Minikube>kubectl describe service/web
Name:                     web
Namespace:                ekskub
Labels:                   app=web
Annotations:              <none>
Selector:                 app=web
Type:                     LoadBalancer
IP:                       10.100.242.100
LoadBalancer Ingress:     a172221b7a22841529a0acb468717b7f-1830286484.ap-south-1.elb.amazonaws.com
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31304/TCP
Endpoints:                192.168.27.225:80,192.168.40.63:80,192.168.45.154:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason                Age    From                Message
  ----    ------                ----   ----                -------
  Normal  EnsuringLoadBalancer  2m22s  service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer   2m19s  service-controller  Ensured load balancer
```

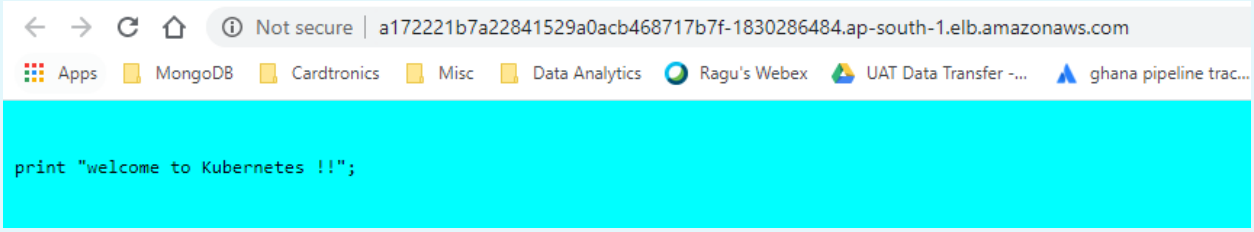
Using below command we can log in to pod & copy code to /var/www/html/

```
C:\Program Files\Kubernetes\Minikube>kubectl get pods
NAME                   READY   STATUS    RESTARTS   AGE
web-d4c668df7-2rpp6    1/1     Running   0          12m
web-d4c668df7-6zq5x    1/1     Running   0          15m
web-d4c668df7-zc247    1/1     Running   0          12m

C:\Program Files\Kubernetes\Minikube>
C:\Program Files\Kubernetes\Minikube>kubectl exec -it web-d4c668df7-2rpp6 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl kubectl exec [POD] -- [COMMAND] instead.
[root@web-d4c668df7-2rpp6 /]# ls
anaconda-post.log  bin  boot  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@web-d4c668df7-2rpp6 /]# cd /var/www/html
[root@web-d4c668df7-2rpp6 html]# ls
index.php
```

```
C:\Program Files\Kubernetes\Minikube>kubectl cp index.php web-d4c668df7-2rpp6:/var/www/html/index.php

C:\Program Files\Kubernetes\Minikube>
```



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.

```
C:\Program Files\Kubernetes\Minikube>kubectl get rs
NAME            DESIRED   CURRENT   READY   AGE
web-d4c668df7   3         3         3       28m

C:\Program Files\Kubernetes\Minikube>kubectl get pods
NAME                   READY   STATUS    RESTARTS   AGE
web-d4c668df7-2rpp6    1/1     Running   0          27m
web-d4c668df7-6zq5x    1/1     Running   0          30m
web-d4c668df7-zc247    1/1     Running   0          27m

C:\Program Files\Kubernetes\Minikube>kubectl get pvc
No resources found in ekskub namespace.

C:\Program Files\Kubernetes\Minikube>kubectl get pv
No resources found in ekskub namespace.

C:\Program Files\Kubernetes\Minikube>kubectl get sc
NAME            PROVISIONER             AGE
gp2 (default)   kubernetes.io/aws-ebs   67m

C:\Program Files\Kubernetes\Minikube>kubectl describe sc gp2
```



```
C:\Program Files\Kubernetes\Minikube>kubectl describe sc gp2
Name:           gp2
IsDefaultClass:  Yes
Annotations:     kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"storage.k8s.io/v1","kind":"StorageClass","metadata":{"annotations":{"storageclass.kube
netes.io/is-default-class":"true"},"name":"gp2"},"parameters":{"fsType":"ext4","type":"gp2"},"provisioner":"kubernetes.io/aws-ebs","volumeBindingMode":"WaitForFirstCons
umer"}
,storageclass.kubernetes.io/is-default-class=true
Provisioner:           kubernetes.io/aws-ebs
Parameters:            fsType=ext4,type=gp2
AllowVolumeExpansion:  <unset>
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
Events:                <none>
```

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.

```
C:\Program Files\Kubernetes\Minikube>kubectl create -f D:\EKS\pvc.yml
persistentvolumeclaim/lwpvc1 created

C:\Program Files\Kubernetes\Minikube>kubectl create -f D:\EKS\sc.yml
storageclass.storage.k8s.io/lwsc1 created

C:\Program Files\Kubernetes\Minikube>kubectl get pvc
NAME     STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
lwpvc1   Pending                                      lwsc1          18s
```

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

```
C:\Program Files\Kubernetes\Minikube>kubectl get sc
NAME            PROVISIONER             AGE
gp2 (default)   kubernetes.io/aws-ebs   90m
lwsc1           kubernetes.io/aws-ebs   63s

C:\Program Files\Kubernetes\Minikube>kubectl get pvc
NAME     STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
lwpvc1   Bound    pvc-1536c512-0d8d-4b43-9521-8999a2c7a455   10Gi       RWO            lwsc1          2m36s

C:\Program Files\Kubernetes\Minikube>kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM           STORAGECLASS   REASON   AGE
pvc-1536c512-0d8d-4b43-9521-8999a2c7a455   10Gi       RWO            Retain           Bound    ekskub/lwpvc1   lwsc1                   2m17s
```

```
C:\Program Files\Kubernetes\Minikube>kubectl describe sc lwsc1
Name:                  lwsc1
IsDefaultClass:        No
Annotations:           <none>
Provisioner:           kubernetes.io/aws-ebs
Parameters:            type=io1
AllowVolumeExpansion:  <unset>
MountOptions:          <none>
ReclaimPolicy:         Retain
VolumeBindingMode:     Immediate
Events:                <none>
```

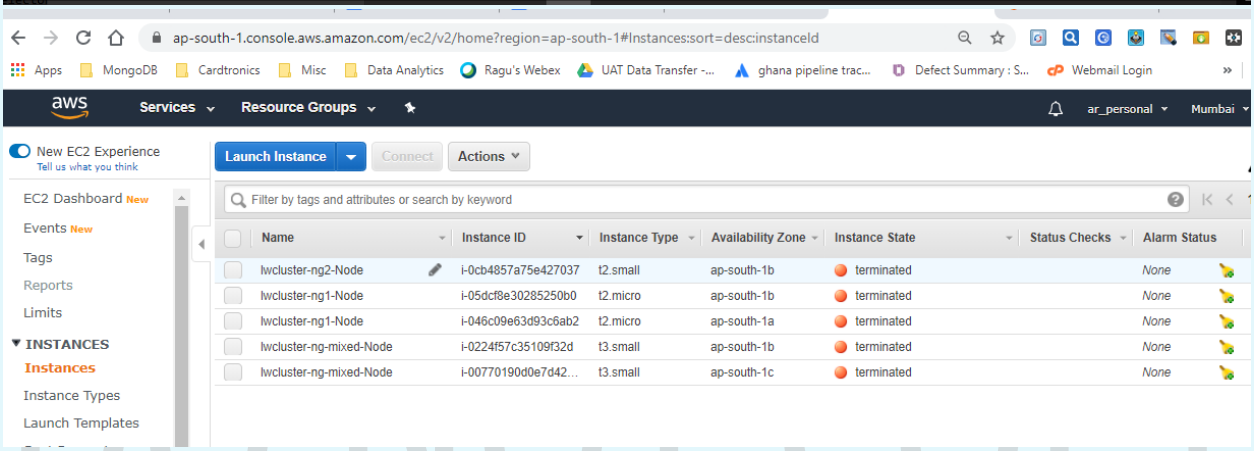
```
C:\Program Files\Kubernetes\Minikube>kubectl describe sc gp2
Name:            gp2
IsDefaultClass:  Yes
Annotations:     kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"storage.k8s.io/v1","kind":"StorageClass","metadata":{"annotations":{"storageclass.kuber
netes.io/is-default-class":"true"},"name":"gp2"},"parameters":{"fsType":"ext4","type":"gp2"},"provisioner":"kubernetes.io/aws-ebs","volumeBindingMode":"WaitForFirstCons
umer"}
,storageclass.kubernetes.io/is-default-class=true
Provisioner:           kubernetes.io/aws-ebs
Parameters:            fsType=ext4,type=gp2
AllowVolumeExpansion:  <unset>
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
Events:                <none>
```

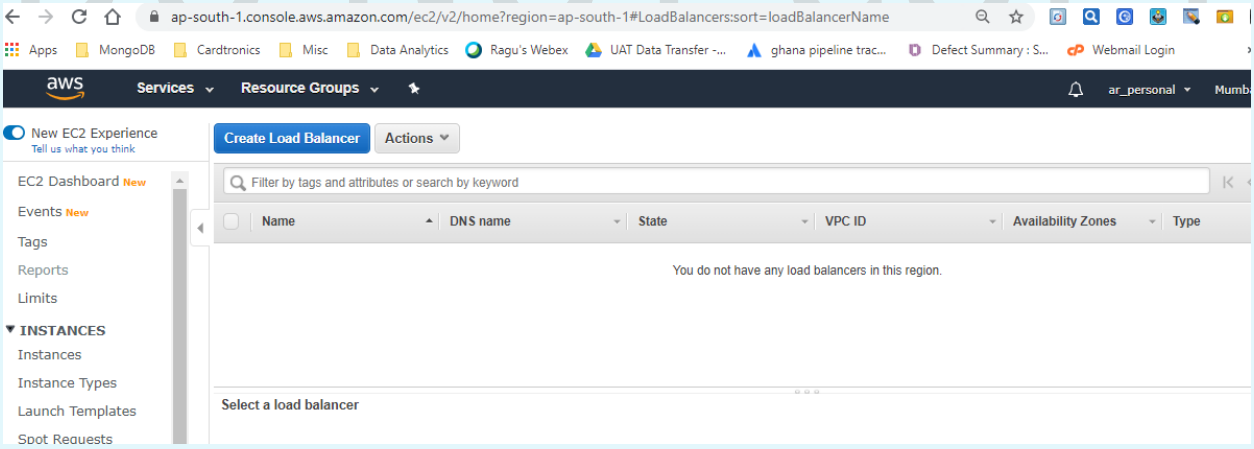
```
C:\Program Files\Kubernetes\Minikube>kubectl delete all --all
pod "web-d4c668df7-9bfdc" deleted
pod "web-d4c668df7-mq2z4" deleted
pod "web-d4c668df7-rz9kn" deleted
service "web" deleted
deployment.apps "web" deleted
replicaset.apps "web-d4c668df7" deleted
```

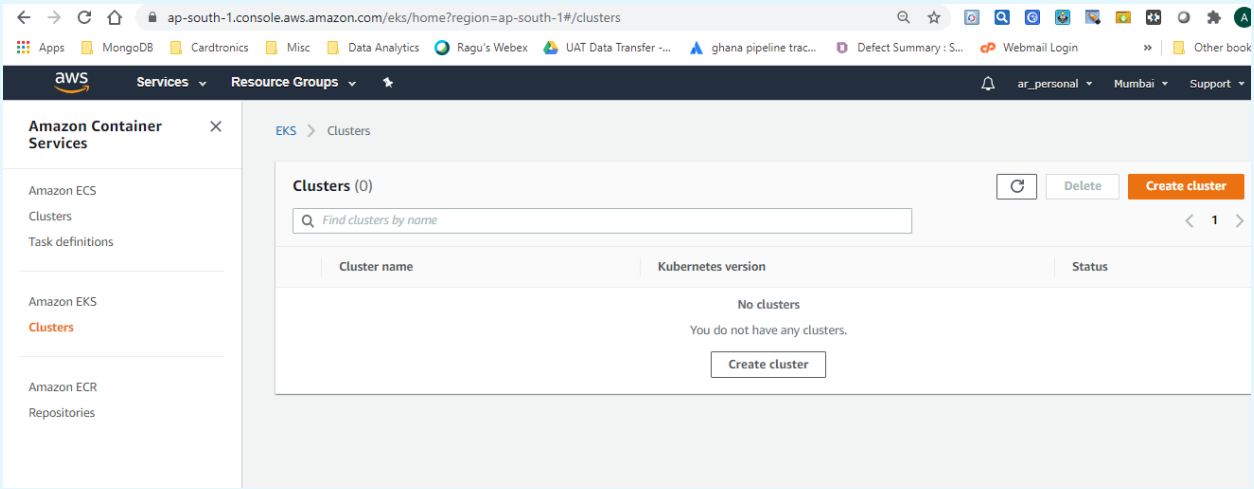
We can finally delete the eks cluster using below command
This will terminate all the instances, delete LoadBalancer as can be seen below.

```
C:\Program Files\Kubernetes\Minikube>eksctl delete cluster -f D:\EKS\cluster.yml
[ℹ]  eksctl version 0.21.0
[ℹ]  using region ap-south-1
[ℹ]  deleting EKS cluster "lwcluster"
[ℹ]  either account is not authorized to use Fargate or region ap-south-1 is not supported. Ignoring error
[ℹ]  cleaning up LoadBalancer services
[ℹ]  2 sequential tasks: { 3 parallel sub-tasks: { delete nodegroup "ng2", delete nodegroup "ng1", delete nodegroup "ng-mixed" }, delete cluster control plane "lwcluste
r" [async] }
[ℹ]  will delete stack "eksctl-lwcluster-nodegroup-ng-mixed"
[ℹ]  waiting for stack "eksctl-lwcluster-nodegroup-ng-mixed" to get deleted
[ℹ]  will delete stack "eksctl-lwcluster-nodegroup-ng2"
[ℹ]  waiting for stack "eksctl-lwcluster-nodegroup-ng2" to get deleted
[ℹ]  will delete stack "eksctl-lwcluster-nodegroup-ng1"
[ℹ]  waiting for stack "eksctl-lwcluster-nodegroup-ng1" to get deleted
[ℹ]  will delete stack "eksctl-lwcluster-cluster"
[ℹ]  all cluster resources were deleted

C:\Program Files\Kubernetes\Minikube>
```



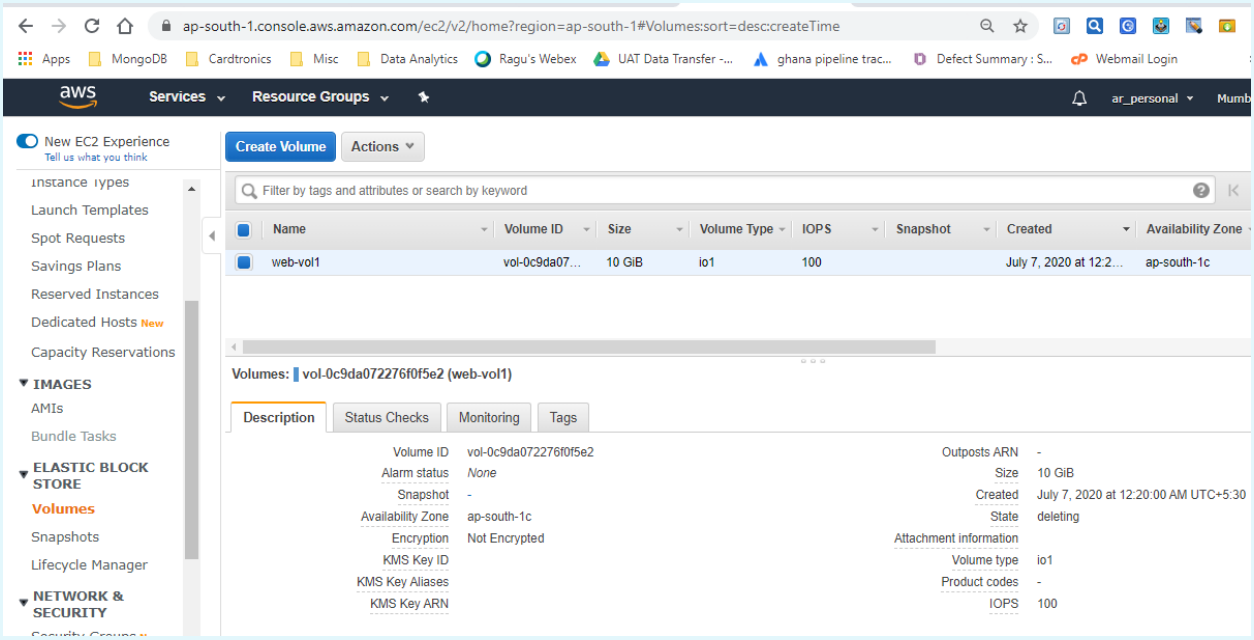




Since reclaim policy was set to Retain in sc.yml , EBS volume is not getting removed even after deleting the Kubernetes cluster. We have to remove the EBS volume manually as if we keep is as it is AWS will charge for the provisioned EBS volume.

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: lwscl
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
reclaimPolicy: Retain
```

COPY



```
C:\Program Files\Kubernetes\Minikube>eksctl get cluster
No clusters found
```

Summary