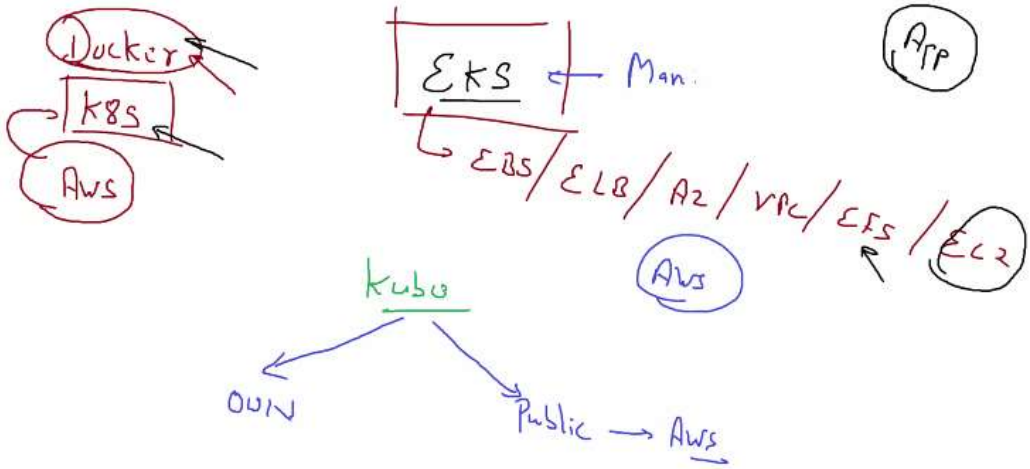
Day 1

02 July 2020 15:08

Actually we have the Docker, K8s, and AWS which are great services we have
And might have a lot of uses and use case where they are used

But we have EKS the Elastic Kubernetes as Service which is the K8s managed on the AWS Cloud

- Which gives us the all the power and features of the (Public) Cloud
- And will be managed by the AWS people (the developers)



- And it is the form of Managed K8s on the Cloud (as the managing of anything is very important)
    - Like for managing the Docker we use the K8s
    - We are using the EKS which is the Managed K8s

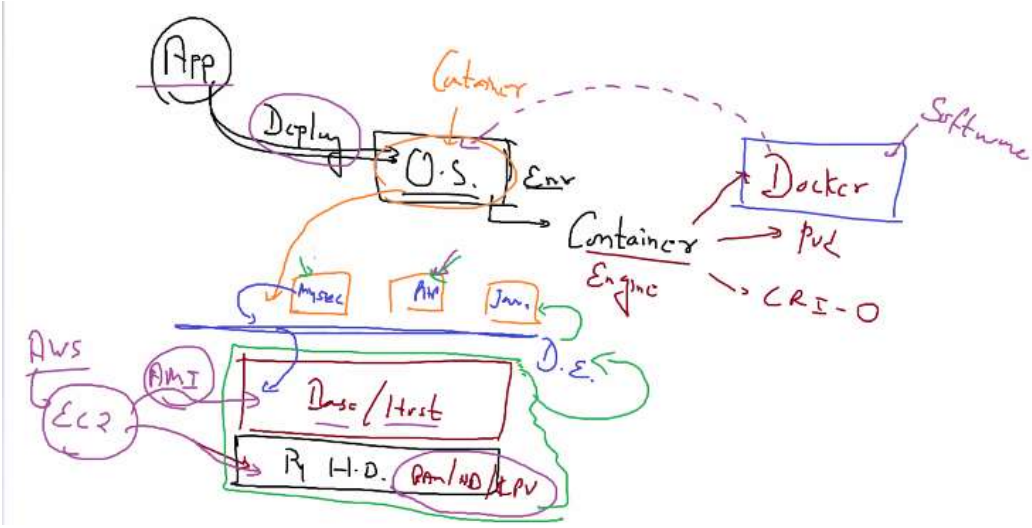
And also in this training we will be using and integrating the EKS with the multiple services
And this is a mature one better than what we can do in our own setup of K8s

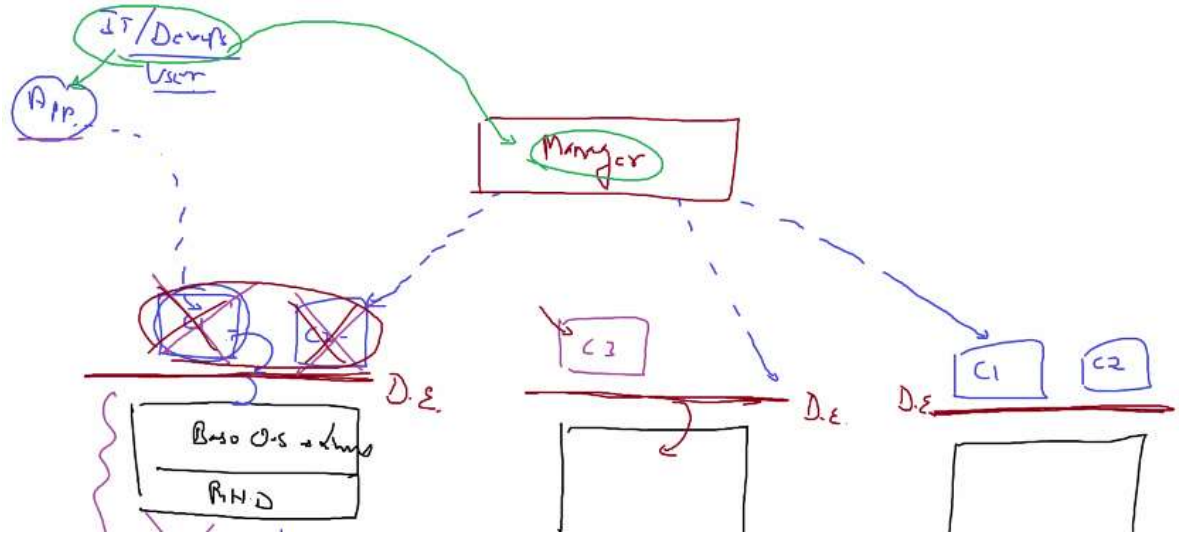
- even if we are thops in K8s,
- we can't rely on the Base system,
- and if we want our base resources or Base System to be mature
- (each and every part of it) then we have the service EKS

==========

Some Basic:

- And for anything we need to have an OS
    - As we might need to run an App or launch or run a program
- And for this we might use the Containers (which is most used now-a-days as within 1 sec we get the OS and also the isolation)

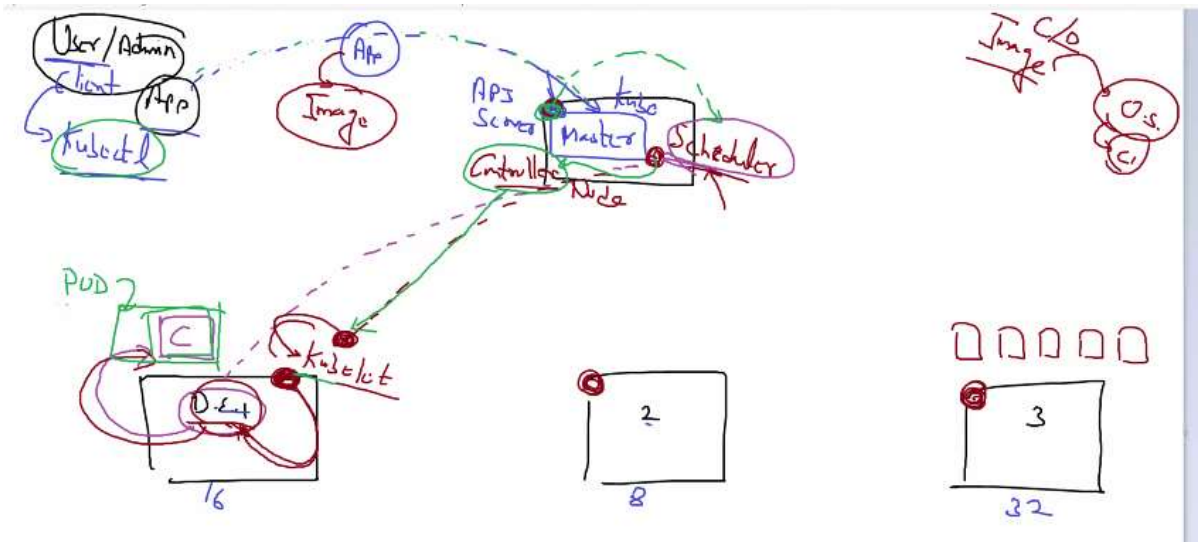
-


- And one of the Container technology is the Docker
- And to launch a container we need the Base Hardware on top of these we can launch it
- But for that we need to have those
- And these can be bought or we can use the CC (like AWS) for the resources
- And for the resources in AWS we have the service called as EC2 using which we can launch an OS or instance
- And then in the Docker there might be issues where the Container might deleted or closed and for this we run the same container in replicas called as Clustering
    - Where we have the multiple Nodes and in those we have multiple containers to be launched
- But we might have the problem of managing cluster and this is done using the K8s
    - Means like to relaunch it if it has deleted
    - This is called as Fault Tolerance
    - And many others
-


- So for all these we need a program (to make this automatic) which will manage the noes, containers and others etc.,
- And this program is called as Container Orchestration Engine
- And one of the many which is famous is the K8s
- And the Managing node is called as master and the others are called as Slaves and these are called Master Slave Architecture

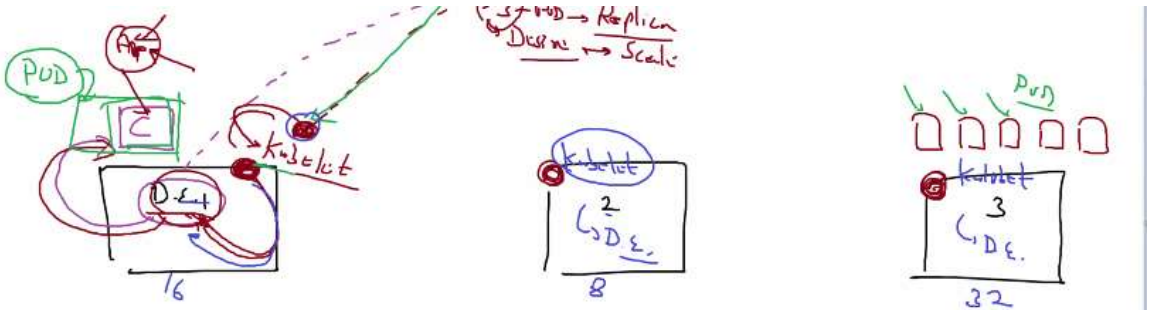
- And for connecting to the master and also to use it we need a program to which we can connect and this program here in K8s is called as API Server
- And the API server will be contacting to the Scheduler
- And the Scheduler is the one that will decide on which Node to use or launch the containers
    - And the one who will launch the containers is the Docker but the Scheduler is the one to decide which node to be used
- And also the K8s or Scheduler don't ko now how to contact to the Docker so we need a program that will contact to the Docker and this is called as Kubelet (here)

-



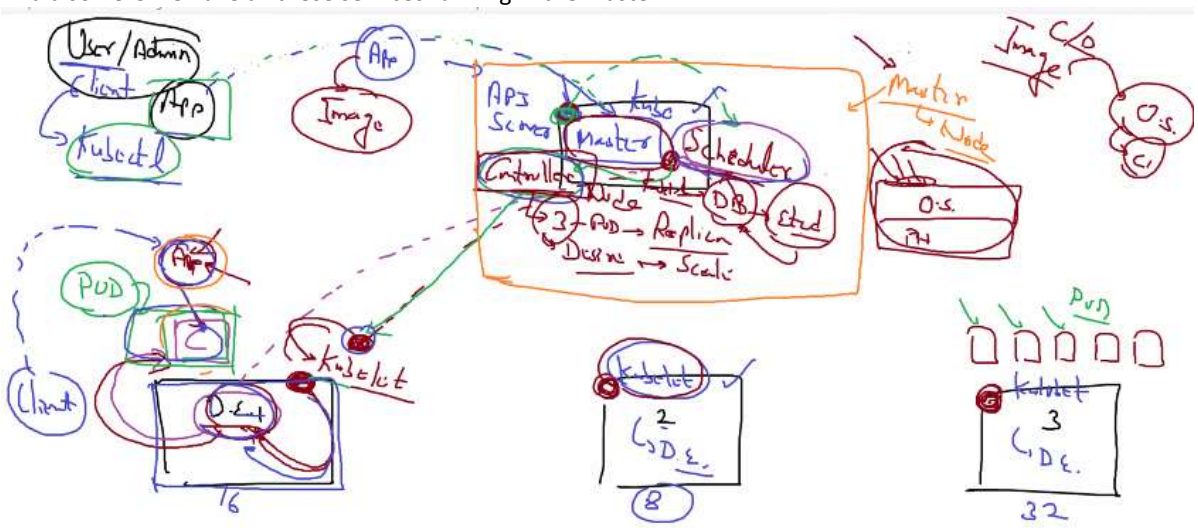
- And as the Kubelet is in K8s we wrap the Container in a box called as POD
- And in all the nodes we need the Kubelet and the container Engine we need to use
- And between the Kubelet and the Scheduler we have an another program called Controller that will manage or tell all the information about the node and also manage the Kubelet for us

-



- And in some cases we have to store some values and for this we have a Database and for this we have a program called as Etcd
- And the controller will also manage the replicas also

And also here we have all these services running in the master

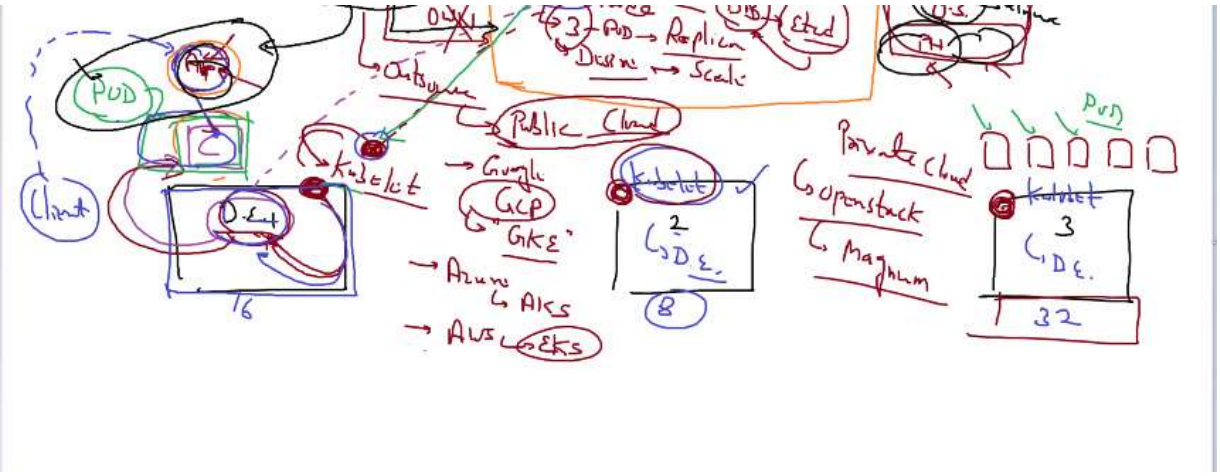


And for this if we configure or somebody configure the Master node (with all the best Practices and technologies)
And actually the configuration of the master is always the same
And also the main thing for the company is the App
And for this we have the many companies that will set up the master for us and also we need the resources for this we might have to arrange or this can also be done by the same company

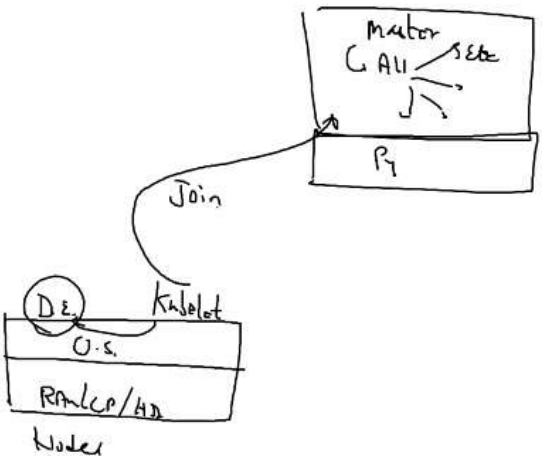
And here the Company is the CC companies like

- AWS
- Google
- Azure (Microsoft)

And these have this service provided with their own name
In which they will also create the master Node and also



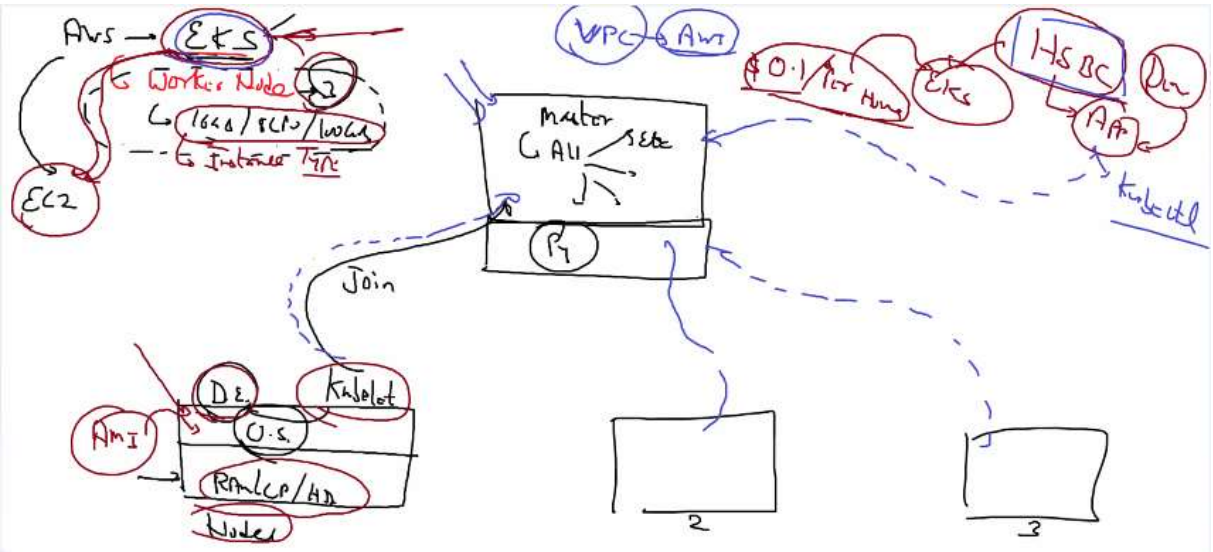
Also apart from the above we also need to even create and manage the entire cluster



And all this we have the service in AWS cloud as the EKS service

- In which we can tell the no. of worker nodes we need
- The configuration we can tell what is the RAM and other resources we want for the slaves

And the charge for this cluster creation is the 0.1$/ hour
And this is only for the Cluster creation and management
And for the other resources we are using in AWS they will be charged differently



And the beauty is that in one click the entire this is launched
And also this is cheap and also we don't need to manage the resources we are using and also it is launched using the best practices

And for the company this is the very big relief or a benefit to use this kind of service

==========

And in the Industry the best and most important thing is the APP
And for them all this managing will be done with ease and in one click by the EKS

==========

And for this we need to have the AWS account

aws.amazon.com › eks › pricing

Amazon EKS Pricing - Managed Kubernetes Service

You pay $0.10 per hour for each Amazon **EKS** cluster that you create. You can use a single Amazon **EKS** cluster to run multiple applications by taking advantage of Kubernetes namespaces and IAM security policies. ... See detailed **pricing** information on the Amazon EC2 **pricing** page.

You visited this page on 3/7/20.

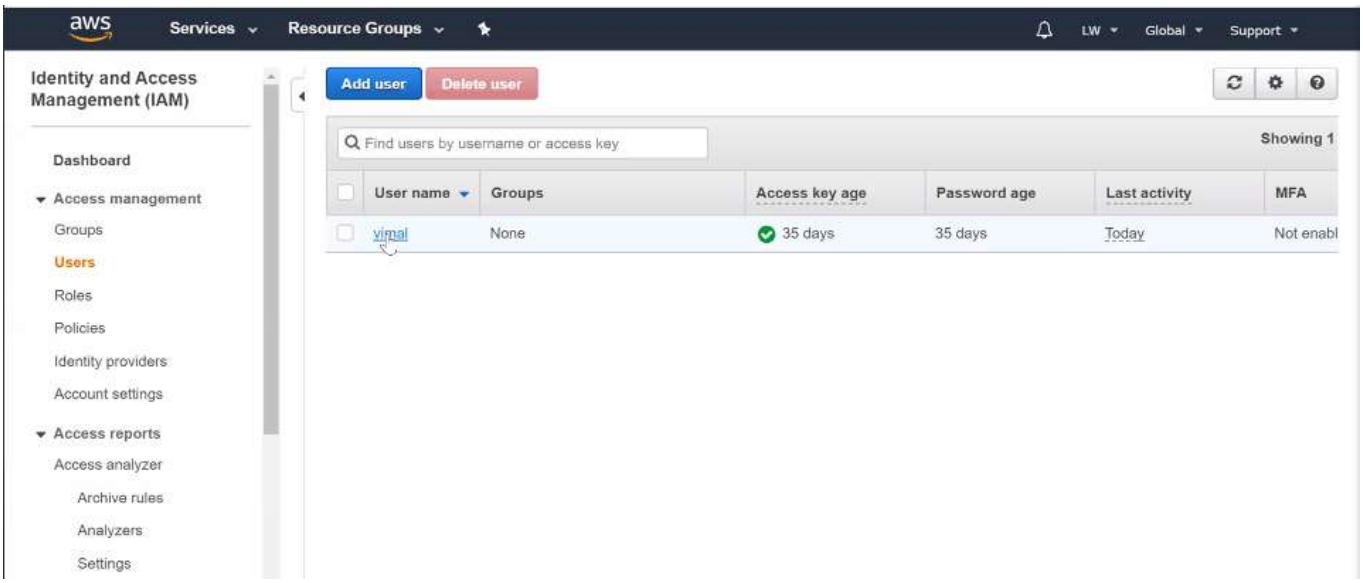
And they will charge for the per hour and also for per second which they have changed and this is a benefit

==========

And to use the AWS we have many ways to use it

- Web UI
- CLI

And for the CLI we need to install the AWS CLI and check for the command if it does not work then we have to set the path then it works



And for using we need to login and for this we need the

- Username - Access Key
- Password - Secret Key

And for this we have or can create an account

And also while creating we need to specify the Role or Power we want to give it and for this we have to set the Admin for all the power if we want to give

```
C:\Users\Vimal Daga>aws  configure
AWS Access Key ID [****************EIGQ]:
AWS Secret Access Key [****************92ua]:
Default region name [ap-south-1]:
Default output format [None]:
```

And to login we use the aws configure and then it will ask the username password default region and format

And then we have logged in perfectly

==========

```
al Daga>aws   eks  list-clusters
```

And this is used to check the cluster we have (EKS)

```
C:\Users\chand>aws eks list-clusters
{
    "clusters": []
}
```

And to create the cluster we can use create-cluster

But the problem is that we can't do much customization with this command

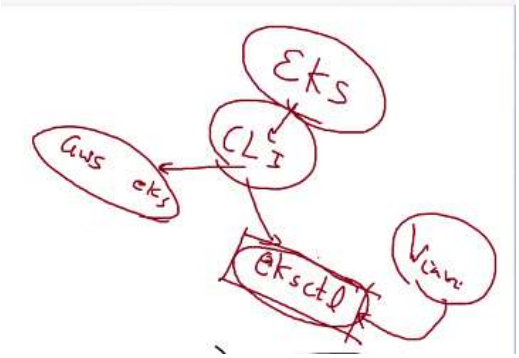
As we don’t have some much options for that

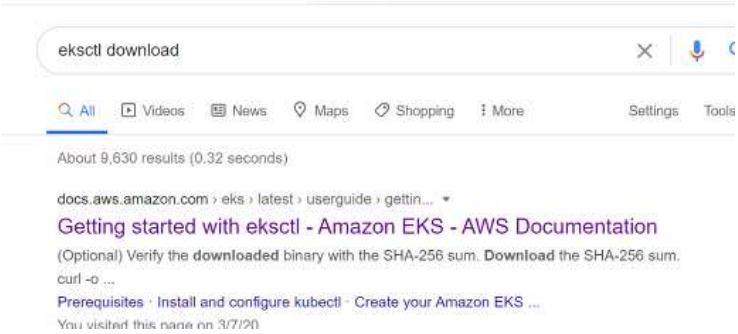
And the EKS is such a great service many companies might be hard to generate such customization

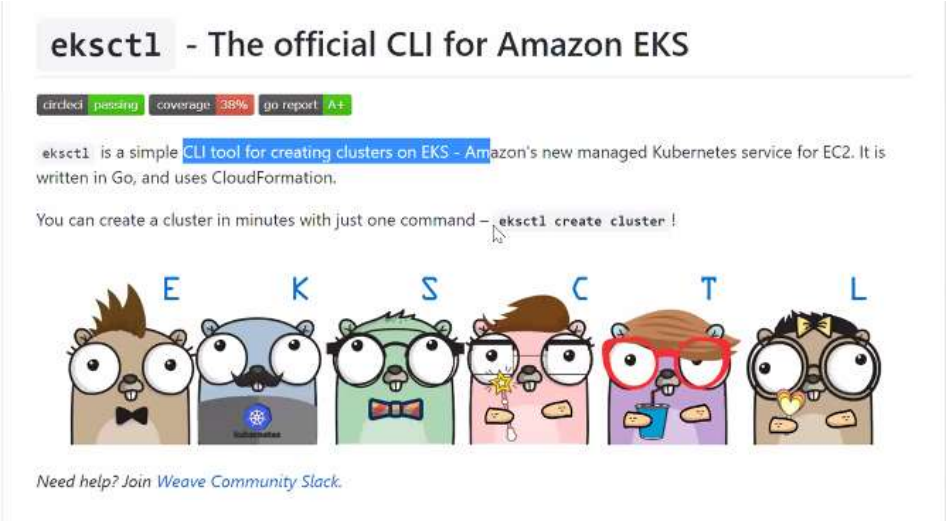
And from the CLI mode we have 2 ways now

- One is AWS typical aws eks command

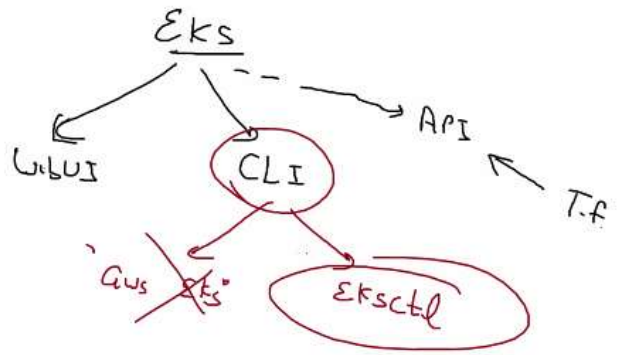
- Or we have some other commands from some other companies (Veave here) is called as EKSCTL (EKS control)
    - And this is a very powerful command
    - That using this we can launch the cluster in one click with all the customizations we want







And this software we can download it and then extract it and then set the path of the location where we have copied it



And this command is specialised for the EKS

```
C:\Users\Vimal Daga>eksctl  version
0.21.0

C:\Users\Vimal Daga>eksctl   get cluster
No clusters found
```

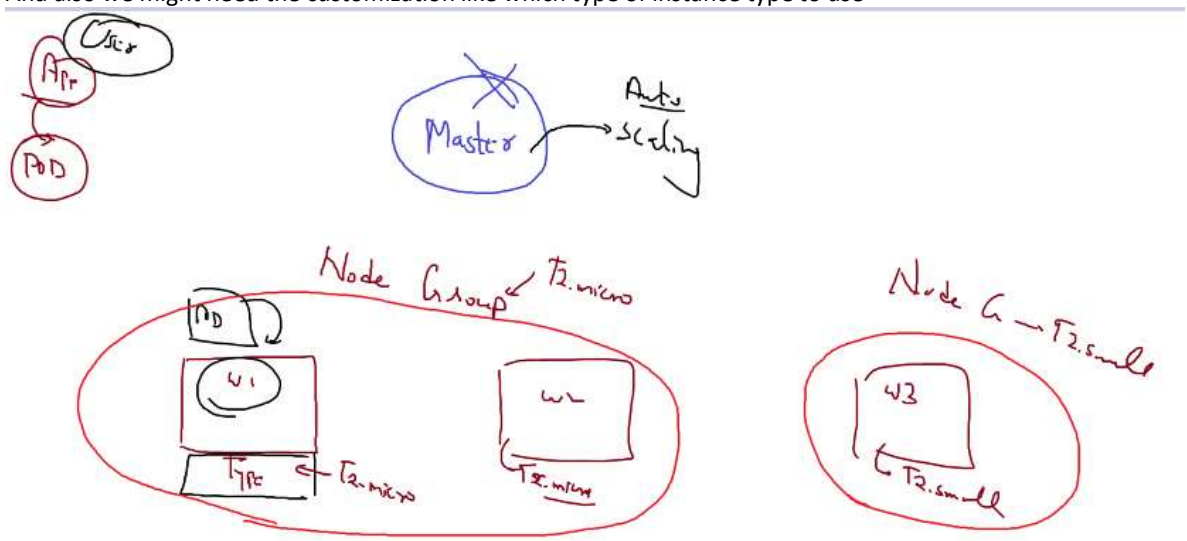
- And here to check the no. of cluster we have we use the | eks get cluster |

```
C:\Users\Vimal Daga>aws configure
AWS Access Key ID [****************EIGQ]:
AWS Secret Access Key [****************92ua]:
Default region name [ap-south-1]:
Default output format [None]:
```

And for this they use the login and password and they are taking it from the aws configure command only
And that's why we have to configure it before the install or using this

And this will also perform the auto scaling (the EKS service) and that is that if the cluster is not able to handle the load then they will automatically scale it

And also we might need the customization like which type of instance type to use



And this is called as Node Group

- And here if we are using two types of instance types then we are having 2 Node groups

And for all this we have to use some script and has to be created using the YAML language (it is a declarative language)

```
nodegroups:
ng1:
        2
        t2.micro
ng2:
        1
        t2.small
```

As per the above we need the 2 node group and also we have to specify which service to use
And also the no. of Nodes of such we need and this is called as capacity

And also has to specify the region where we want to launch

```
kind: ClusterConfig

region: ap-south=1

nodegroups:
ng1:
        capacity: 2
        type: t2.micro
ng2:
        cap: 1
        type: t2.small
```

And also about what are we doing and this is called as the Kind

And this is not the right syntax but this is the information which we have to provide

And also we have to give name to the cluster we are launching and this is called as metadata
And the final syntax looks like the

```
kind: ClusterConfig

metadata:
  name: lwcluster
  region: ap-south=1

nodeGroups:
   - name: ng1
     desiredCapacity: 2
     instanceType: t2.micro
   - name: ng2
     desiredCapacity: 1
     instanceType: t2.small
```

And here it has to be ap-south-1
And here it will launch the cluster in one click as we have said it that the kind is ClusteConfig

And also we have to tell which version to use as this kind will have been come from this some type of program file
And this program file here is called as apiVersion

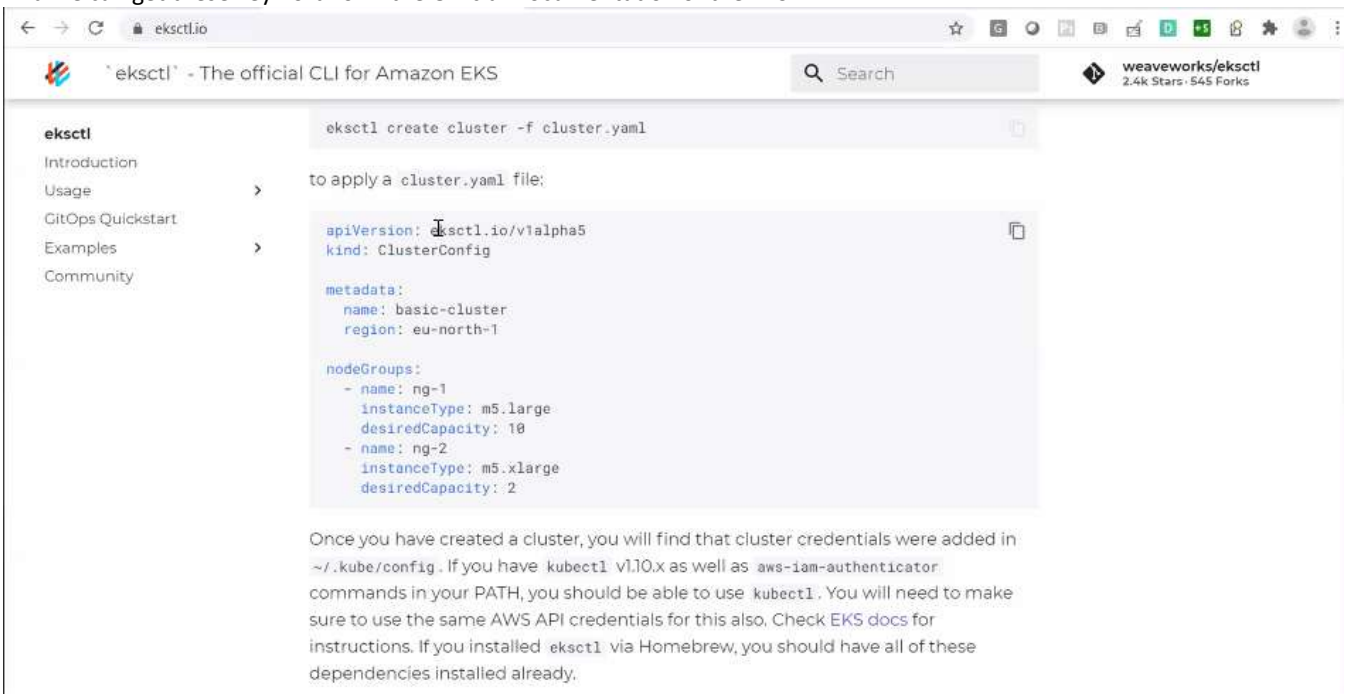
```
apiVersion: eksctl.io/v1alpha5
```

```yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: lwcluster
  region: ap-south-1

nodeGroups:
   - name: ng1
     desiredCapacity: 2
     instanceType: t2.micro
   - name: ng2
     desiredCapacity: 1
     instanceType: t2.small
```

And we can get these keyword form the Official Documentation of the EKS




```
Directory of C:\Users\Vimal Daga\Desktop\eks_class_code

04-07-2020  19:54    <DIR>          .
04-07-2020  19:54    <DIR>          ..
04-07-2020  20:02               262 cluster.yml
               1 File(s)            262 bytes
               2 Dir(s)   2,130,477,056 bytes free

C:\Users\Vimal Daga\Desktop\eks_class_code>eksctl   get cluster
No clusters found
```

And when we run the command then they will create the cluster for us

```
[ℹ]  nodegroup "ng2" will use "ami-073969767527f7306" [AmazonLinux2/1.16]
[ℹ]  using Kubernetes version 1.16
[ℹ]  creating EKS cluster "lwcluster" in "ap-south-1" region with un-managed nodes
[ℹ]  2 nodegroups (ng1, ng2) were included (based on the include/exclude rules)
[ℹ]  will create a CloudFormation stack for cluster itself and 2 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 --cluste
r=lwcluster'
[ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=fal
se} for cluster "lwcluster" in "ap-south-1"
[ℹ]  2 sequential tasks: { create cluster control plane "lwcluster", 2 sequential sub-tasks:
{ no tasks, 2 parallel sub-tasks: { create nodegroup "ng1", create nodegroup "ng2" } } }
[ℹ]  building cluster stack "eksctl-lwcluster-cluster"
[ℹ]  deploying stack "eksctl-lwcluster-cluster"
```

And this progr

apiVersic

And we can ge

Statu

eksc

eksctl

Introduction

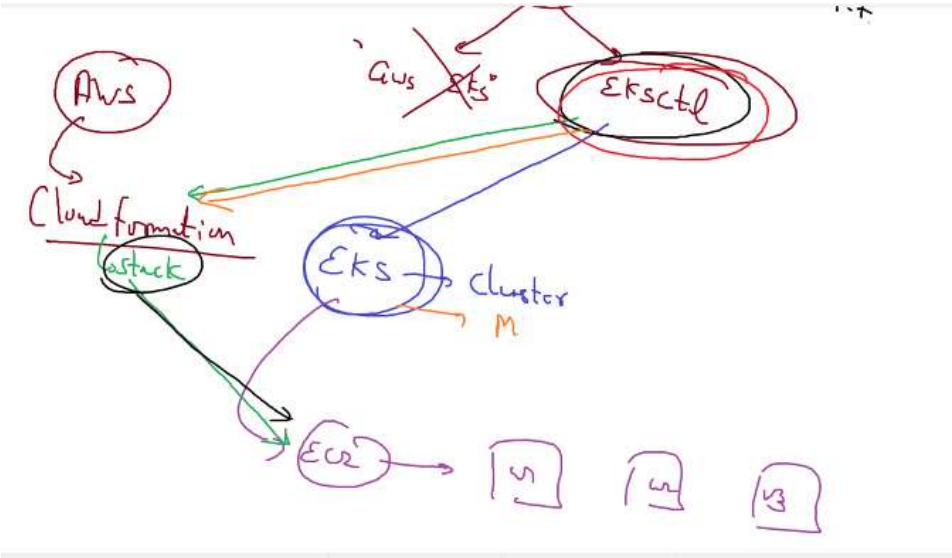
Usage

GitOps Quickst

Examples

Community

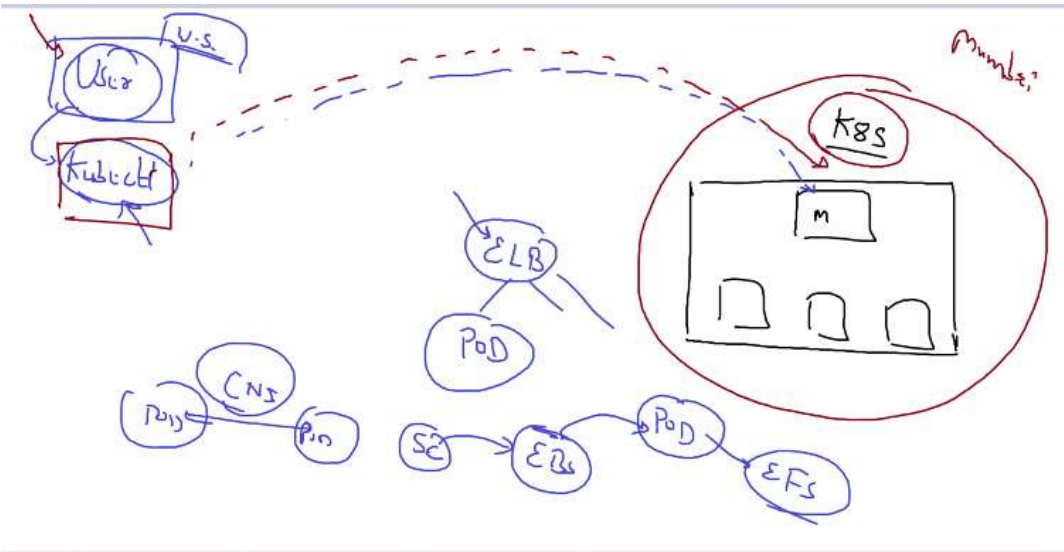
```
will create a CloudFormation stack for cluster its
will create a CloudFormation stack for cluster its
```



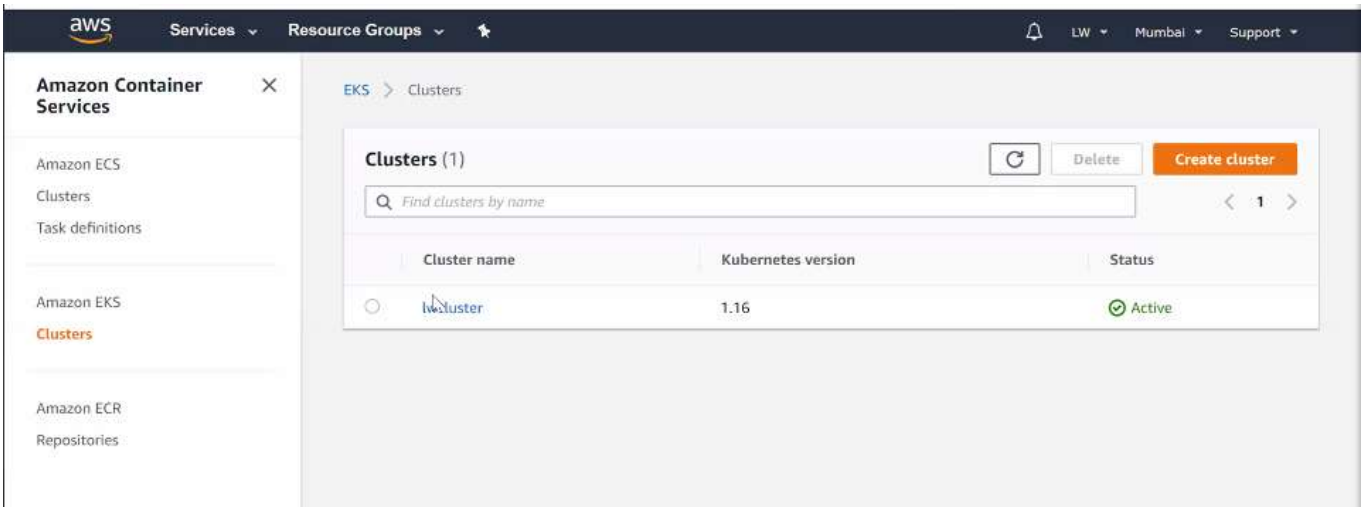
And this is doing by the EKSCTL automatically and this automatically doing is called as the Orchestration
And for this we have the service in AWS called as Cloud Formation
And the EKSCTL is internally creating a Cloud Formation program and this is the one that id doing the stuff for us

And here the Cloud Formation creation is called as Stack in Cloud

And after the cluster is created then we have to use it (and from here our stuff starts)
And for using we have many other things come in between
And using means we are using the K8s in EKS
And to use the CLI of the K8s we need to have the Kubectl program

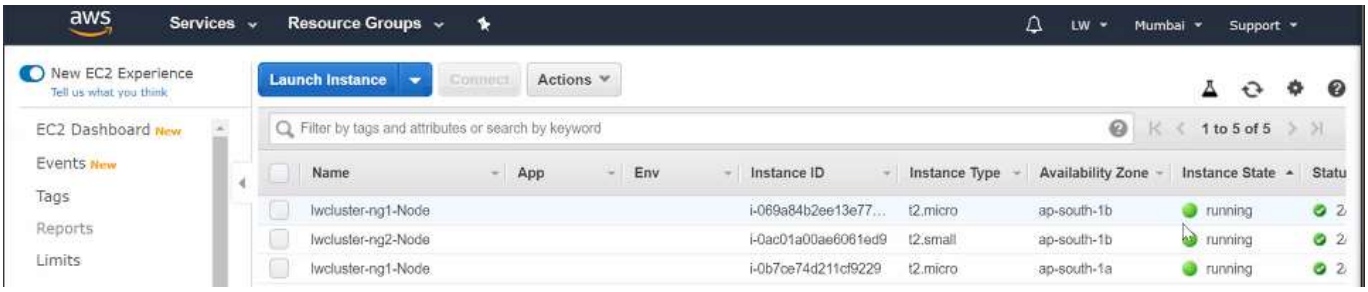


And we will be doing tons of integration and all this cannot be done in our laptop and can only be done using the AWS cloud or some other Cloud
And also the AWS is giving the feature of using its services as this is the one that is managing the




```
C:\Users\Vimal Daga\Desktop\eks_class_code>eksctl   get cluster
NAME            REGION
lwcluster       ap-south-1
```

now the cluster is created successfully



And in the EC2 we can see that we have 3 more nodes we had created

And here we have forgot to attach the Key pair
So without that we can't login into the node and see what is happening internally

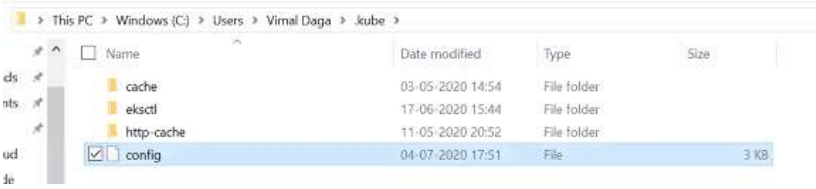
```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: lwcluster
  region: ap-south-1

nodeGroups:
   - name: ng1
     desiredCapacity: 2
     instanceType: t2.micro
     ssh:
        publicKeyName: mykey111222
   - name: ng2
     desiredCapacity: 1
     instanceType: t2.small
```

And internally if we want to attach the key we can use the following code in the file

And in the Kubectl which we have to use for the



And in our case we don't find this file
And we have to create it
And in this we just write the IP of our cluster and then username and password (that's it)

```
C:\Users\Vimal Daga>kubectl config view
apiVersion: v1
clusters: null
contexts: null
current-context: ""
kind: Config
preferences: {}
users: null
```

And if we don't have that file then if we run this command then we know about the details in the file

And we have an aws command that will create this file for us

```
C:\Users\Vimal Daga>aws  eks  update-kubeconfig
```

And for this we have to pass which cluster we have to update this file

```
C:\Users\Vimal Daga>aws  eks  update-kubeconfig    --name lwcluster
Added new context arn:aws:eks:ap-south-1:417149810339:cluster/lwcluster to C:\Users\Vimal Dag
a\.kube\config
```

And the contents in the file will be like

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5RENDQWJDZ
dmI5VXorZUdTVEtrcmIxQjBNS0l3TzFXd1ovNS8wNmJpTnpkbGV5RFNRc2NTTEFGTGFlZjVXWVd0Q1FmMEltC
    server: https://7BC589ADB090295994F768EF134BBFF4.sk1.ap-south-1.eks.amazonaws.com
  name: arn:aws:eks:ap-south-1:417149810339:cluster/lwcluster
contexts:
- context:
    cluster: arn:aws:eks:ap-south-1:417149810339:cluster/lwcluster
    user: arn:aws:eks:ap-south-1:417149810339:cluster/lwcluster
  name: arn:aws:eks:ap-south-1:417149810339:cluster/lwcluster
current-context: arn:aws:eks:ap-south-1:417149810339:cluster/lwcluster
kind: Config
preferences: {}
users:
- name: arn:aws:eks:ap-south-1:417149810339:cluster/lwcluster
  user:
    exec:
```

And this is the IP or URL for the cluster we have
And then we can use the K8s service now

```
C:\Users\Vimal Daga\Desktop\eks_class_code>
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pods
No resources found in default namespace.
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
ip-192-168-57-189.ap-south-1.compute.internal   Ready    <none>   21m   v1.16.8-eks-fd1ea7
ip-192-168-7-78.ap-south-1.compute.internal     Ready    <none>   21m   v1.16.8-eks-fd1ea7
ip-192-168-9-210.ap-south-1.compute.internal    Ready    <none>   20m   v1.16.8-eks-fd1ea7
```

And these are the nodes that are launched and also these are the IP's of the nodes
And these are the actually the EC2 IP's

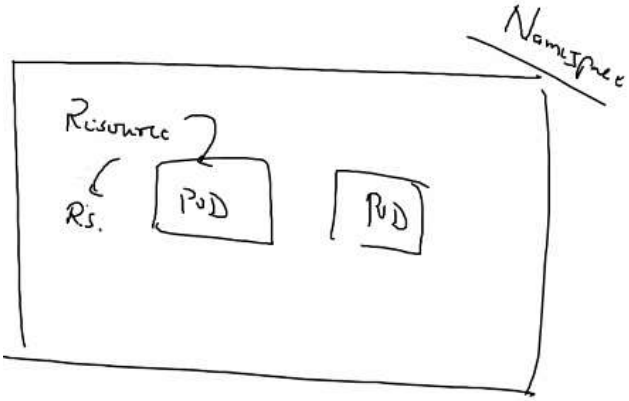
And if we want to see some extra information about the node we use the describe command

```
  Container Runtime Version:  docker://19.3.6
  Kubelet Version:            v1.16.8-eks-fd1ea7
  Kube-Proxy Version:         v1.16.8-eks-fd1ea7
ProviderID:                   aws:///ap-south-1b/i-069a84b2ee13e771a
Non-terminated Pods:          (3 in total)
  Namespace                   Name                       CPU Requests  CPU Limits  Memory Requests  Memory Lim
its  AGE
  ---------                   ----                       ------------  ----------  ---------------  ----------
---  ---
  kube-system                 aws-node-j8w9f             10m (1%)      0 (0%)      0 (0%)           0 (0%)
     22m
  kube-system                 coredns-6856799b8d-zqxzn   100m (10%)    0 (0%)      70Mi (11%)       170Mi (27%
)    28m
  kube-system                 kube-proxy-ck7pc           100m (10%)    0 (0%)      0 (0%)           0 (0%)
     22m
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource                   Requests    Limits
  --------                   --------    ------
  cpu                        210m (22%)  0 (0%)
  memory                     70Mi (11%)  170Mi (27%)
  ephemeral-storage          0 (0%)      0 (0%)
  hugepages-2Mi              0 (0%)      0 (0%)
  attachable-volumes-aws-ebs 0           0
Events:
  Type    Reason    Age   From                                                    Message
```

And here we see lots of information

==========

And also if we have the PODS in K8s we keep all the related one in some box called as namespace



And we can see the namespace (and many are created by the K8s for us) and by default we are using the deafult Namespace

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get  ns
NAME              STATUS   AGE
default           Active   32m
kube-node-lease   Active   32m
kube-public       Active   32m
kube-system       Active   32m
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl create namespace  lwns
namespace/lwns created

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get  ns
NAME              STATUS   AGE
default           Active   33m
kube-node-lease   Active   33m
kube-public       Active   33m
kube-system       Active   33m
lwns              Active   2s
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pods
No resources found in default namespace.
```

And if we want to change our default namespace then we will use the config file and then if we want to change the namespace then we have to add the context

```
>kubectl config view
```

This is an another way to view the file

And we can change the default NS using the following command

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl config  set-context --current --namespace=
lwns
Context "arn:aws:eks:ap-south-1:417149810339:cluster/lwcluster" modified.
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pods
No resources found in lwns namespace.
```

And till here is just the set up part and also about the use cases how the EKS is using the K8s

And now the fun part starts where we integrate features of services of AWS with the K8s (in EKS)
And if we want to see the cluster information then we can use the following command

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl cluster-info
Kubernetes master is running at https://7BC589ADB090295994F768EF134BBFF4.sk1.ap-south-1.eks
amazonaws.com
CoreDNS is running at https://7BC589ADB090295994F768EF134BBFF4.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'.
```

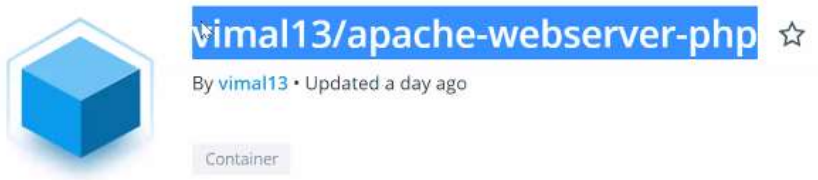
==========

After the Break:

==========

And now we have our cluster ready and now we can check for the practical and integration we want to do

And here we use some image that has the apache webserver configured like



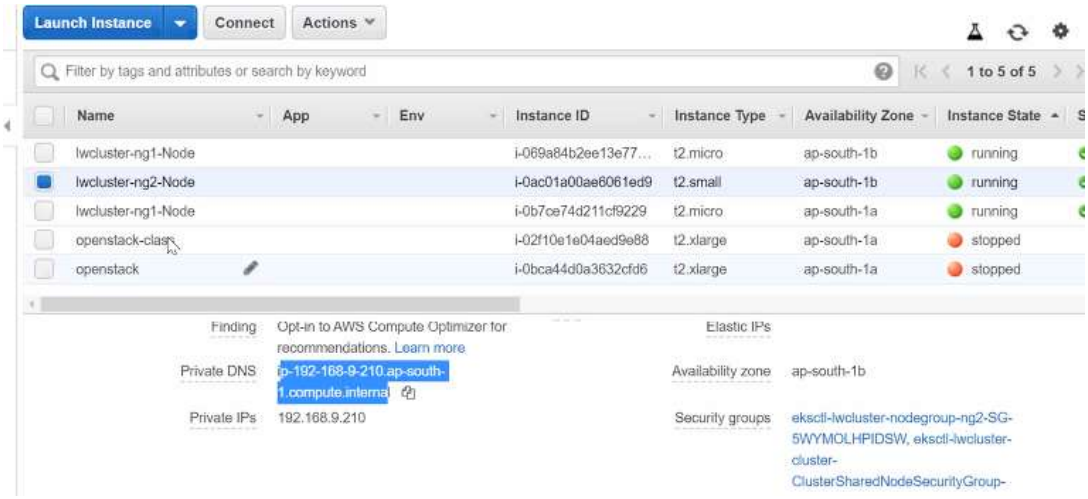
And here we can launch the deployment

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl create deployment myweb --image=vimal1
3/apache-webserver-php
deployment.apps/myweb created
```

And this way we can use this

And internally they are connected to the AWS and then EKS and then they are we have launched our POD in one of the node in the EC2 of the Amazon

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pods -o wide
NAME                    READY   STATUS    RESTARTS   AGE   IP              NODE
            NOMINATED NODE   READINESS GATES
myweb-79b48fb9f5-c7wbn   1/1     Running   0          49s   192.168.2.156   ip-192-168-9-210.ap-south-1.compute
.internal   <none>           <none>
```



And we can see that it is only one container in the POD as we have asked for one
And now if we want to increase it we can use the scaling

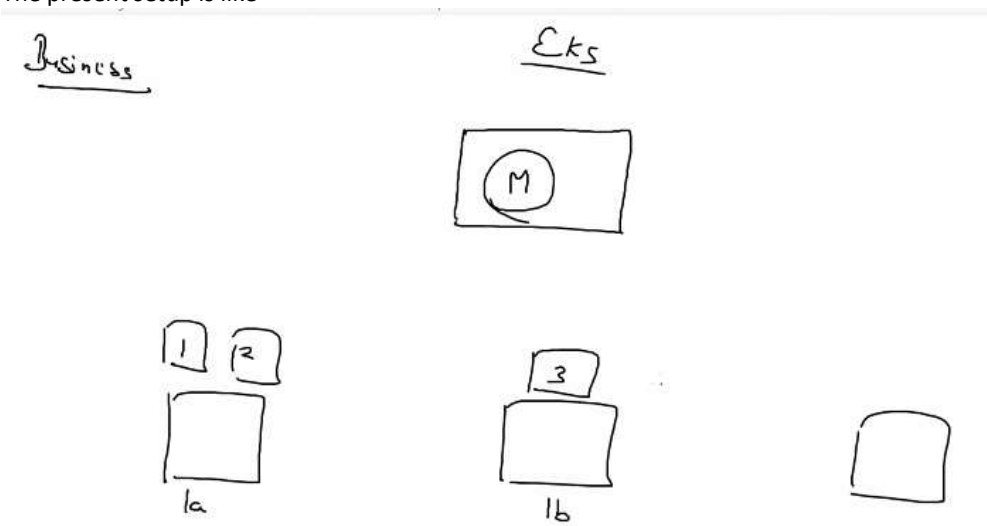
```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl scale deployment myweb --replicas=3
deployment.apps/myweb scaled

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pods
NAME                      READY   STATUS              RESTARTS   AGE
myweb-79b48fb9f5-c7wbn    1/1     Running             0          2m29s
myweb-79b48fb9f5-p99hx    1/1     Running             0          7s
myweb-79b48fb9f5-xm5tc    0/1     ContainerCreating   0          7s
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pods  -o  wide
NAME                      READY   STATUS    RESTARTS   AGE     IP               NODE
                NOMINATED NODE   READINESS GATES
myweb-79b48fb9f5-c7wbn    1/1     Running   0          2m38s   192.168.2.156    ip-192-168-9-210.ap-south-1.comp
ute.internal    <none>           <none>
myweb-79b48fb9f5-p99hx    1/1     Running   0          16s     192.168.11.0     ip-192-168-9-210.ap-south-1.comp
ute.internal    <none>           <none>
myweb-79b48fb9f5-xm5tc    1/1     Running   0          16s     192.168.34.113   ip-192-168-57-189.ap-south-1.com
pute.internal   <none>           <none>
```

And the beauty of the cluster is that these pods are launched in different Nodes
And the one who plans this is the Scheduler
And the beauty here is that one node is in the different data centre and others are in other data centre

The present setup is like

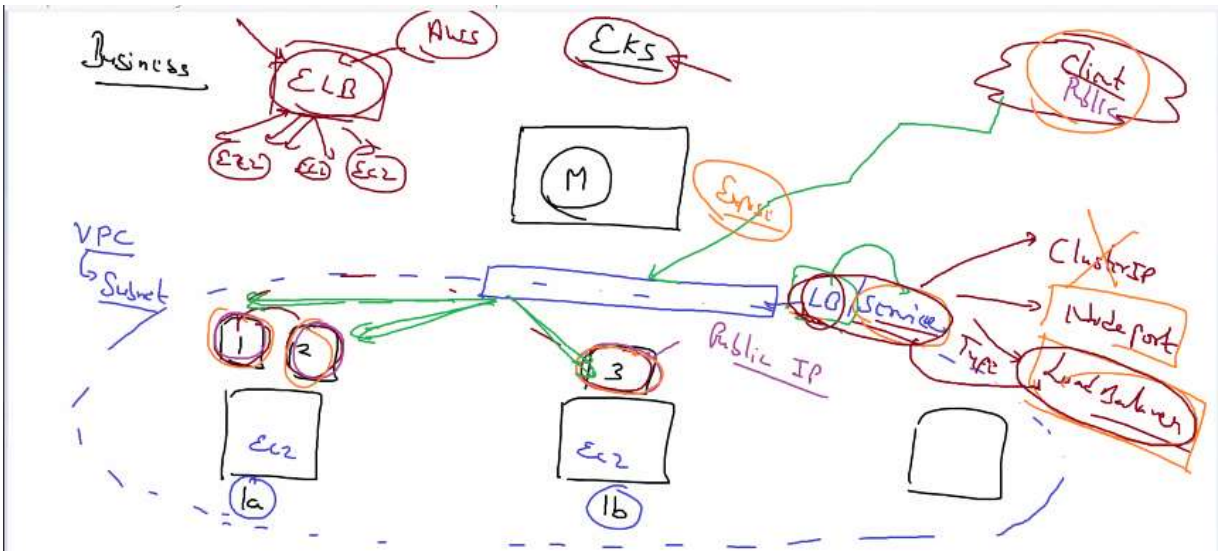


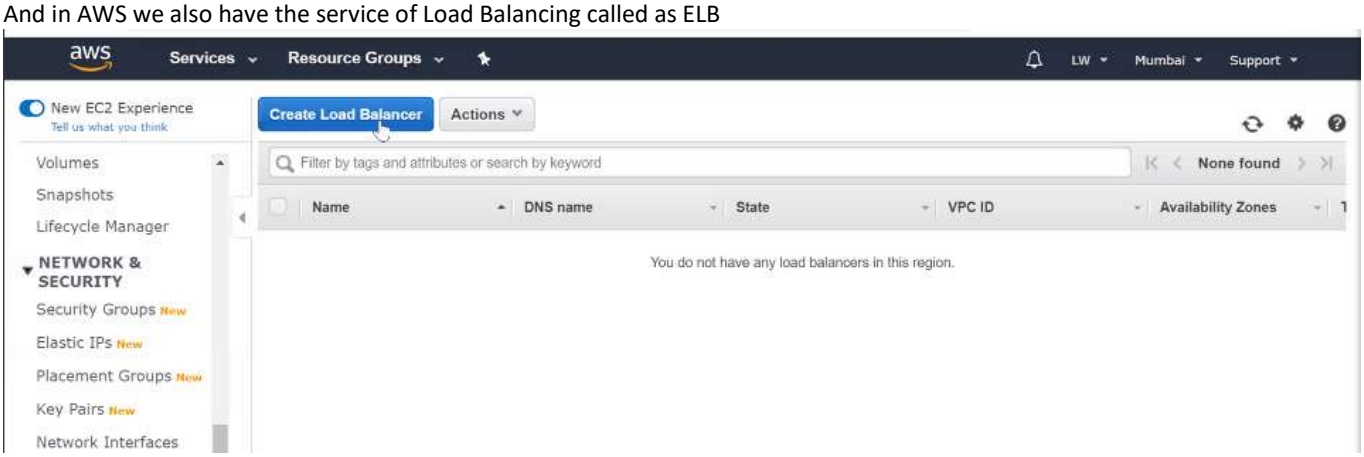
And here they are running all the nodes and instances in their own network or VPC and inside a Subnet

And for this we can't connect them from outside world as they are using the Public IP or in the Public World
And for this we have to create a program called Service or Load balancer

- And in K8s we have 3 types of these like
  - Node Port (where we expose it using the Node IP)
  - Cluster IP
  - Load Balancer
- And here we will be using the Load Balancer service
- And this one give us the access to the outside world and also perform the load balancing
- And for this we have 2 types
  - Either we can use the Node Port or
  - Can use Load Balancer

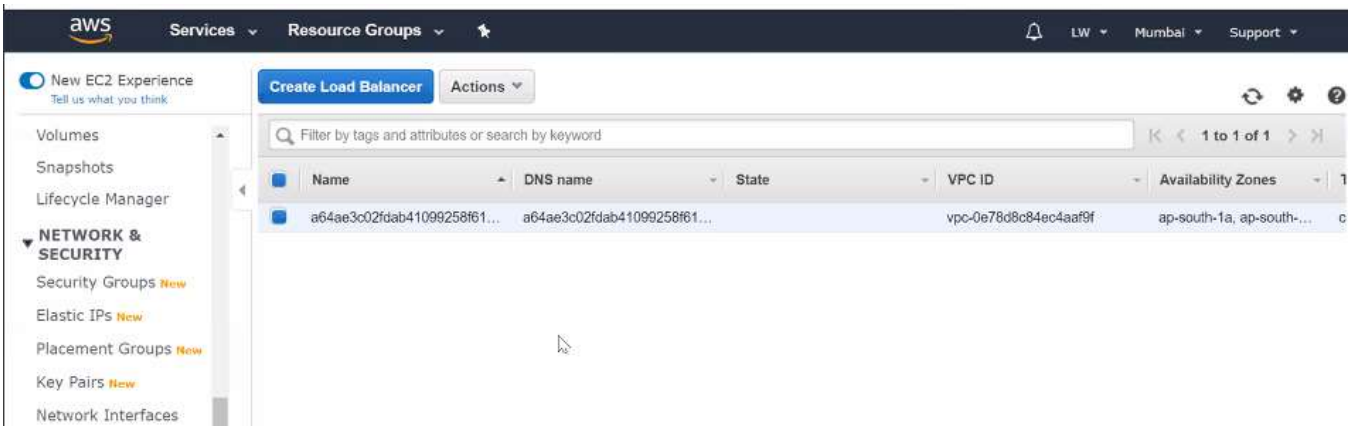
- And today we will be using the Load Balancer

- 

- And in AWS we also have the service of Load Balancing called as ELB
- 

- And for this we have to click here and then we have to configure it
- But if we use the Load Balancer as a service in K8s (this is the internal one in K8s)
- Then it will perform the load balancing
- But also where it also have the knowledge or check which type of Cloud we are using (here it is AWS)
- And then they check for the LB service they have which we can use (here it is ELB)
- And then they will create one automatically and then connect it to the K8s internal Load balancer Service
- Means the IP of the ELB is used as the Load Balancing IP in the K8s by the service

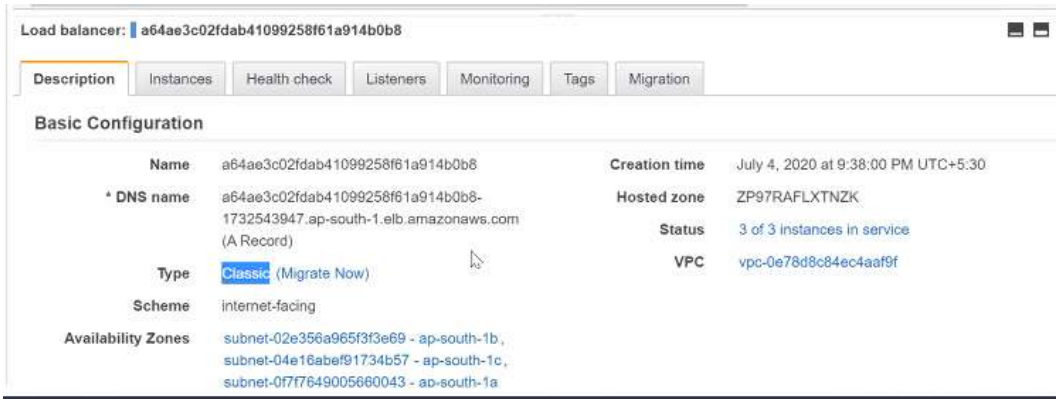
```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl expose deployment myweb  --type=LoadBalancer  --port=80
```



And this has created as ELB for the Load Balancer (External Load Service)

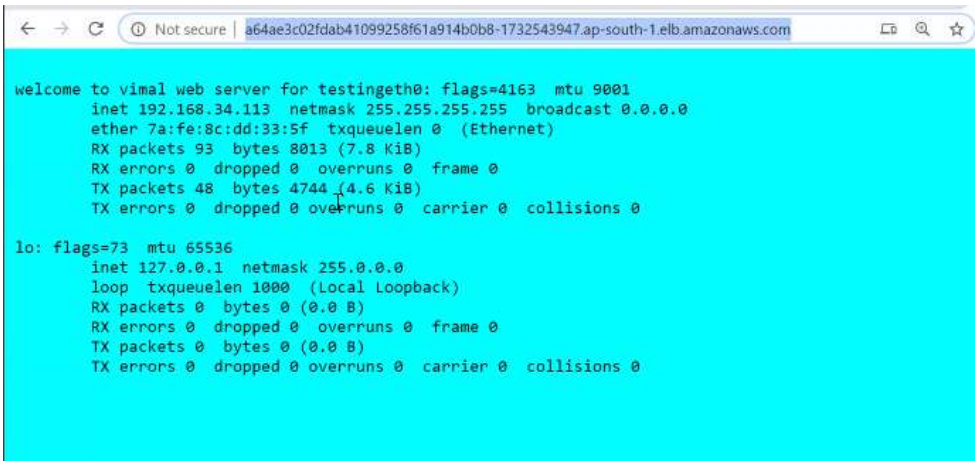
Technically K8s is the independent service and as well as the ELB
But these are integrated in the EKS

And the benefit of the ELB is that it will provide us the Public IP through which we can access the POD or instance form anywhere from the world



And in the ELB they are using the Classic Load Balancer and actually we have 3 types of the Load Balancer

And this the use case of the Load balancer with that of the Node Port in K8s
As when we want to connect some load Balancer then we can use the Load Balancer instead of Node Port



And now we can connect to the page or apache server

And here we have 2 load balancer
- One is the EKS load balancer
- And the other is the K8s Load Balancer

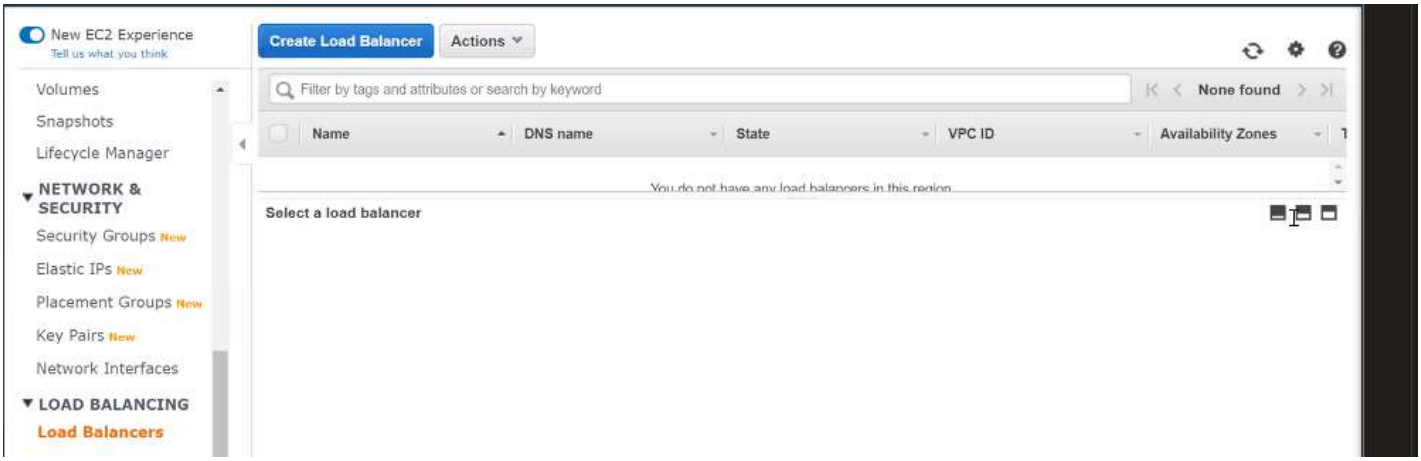
And here they are balancing it in between the PODS and also in between the Nodes and the nodes are in between different Data centres so we have the load balanced between the Data Centres too

It is like the Base Infrastructure is maintained by the AWS and the Management of the POD or application is done by the Instance

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  describe service/myweb
Name:                     myweb
Namespace:                lwns
Labels:                   app=myweb
Annotations:              <none>
Selector:                 app=myweb
Type:                     LoadBalancer
IP:                       10.100.147.38
LoadBalancer Ingress:     a64ae3c02fdab41099258f61a914b0b8-1732543947.ap-south-1.elb.amazonaws.com
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  32753/TCP
Endpoints:                192.168.11.0:80,192.168.2.156:80,192.168.34.113:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
```

And if we describe our Load Balancer or service then we observe that the Ingress (the input or the Hits) is received by this IP or URL

And now if we delete all in K8s then it will delete the service means the Load Balancer is deleted which means the ELB is also deleted and this we can observe this



And here the complete focus we can look into the App and the management of the resources are also done by the EKS

And if we want to go inside the container then we can use

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl exec -it myweb-79b48fb9f5-6hf46   bash
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl exec -it myweb-79b48fb9f5-6hf46   bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl kubectl exec [P
OD] -- [COMMAND] instead.
[root@myweb-79b48fb9f5-6hf46 /]#
```

And now we are inside the POD

- This is like
  - IN AWS cloud in Mumbai DC
  - We have a Cluster
  - In which we have Node (EC2)
  - And then to a POD in it
  - And we have logged into it from our home

And say if we want to add or change something in the POD then we have to login into it and then we can change it

And we can do this without logging in to the POD
And for this we are just using the K8s command and we don't need to know anything about the AWS
And this can be done using the Copy Command

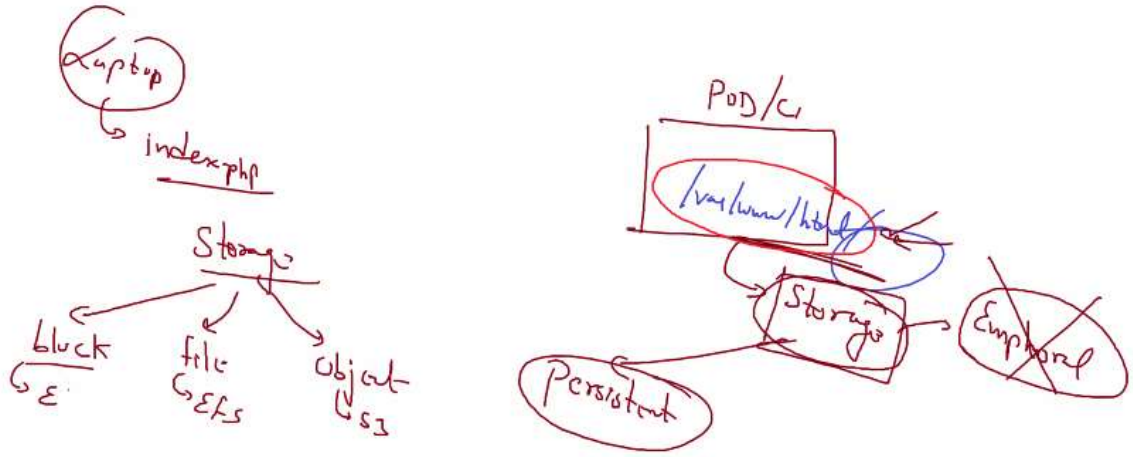
```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl cp  index.php myweb-79b48fb9f5-6hf46:/var/www/html/index.php
```

And this has changed the page

And whatever we have copied inside the POD then it is copied it into the POD storage and by default it is Ephemeral in nature (temporary)
And for this we need to store then or make it persistent

Say for this if we delete all the POD then it gets deleted and if we want to relaunch it automatically then we use the RS which is a controller and this will monitor the POD

And now it will relaunch it



But the new POD it made from the image so it will not have the data that we have copied in the Previous Node

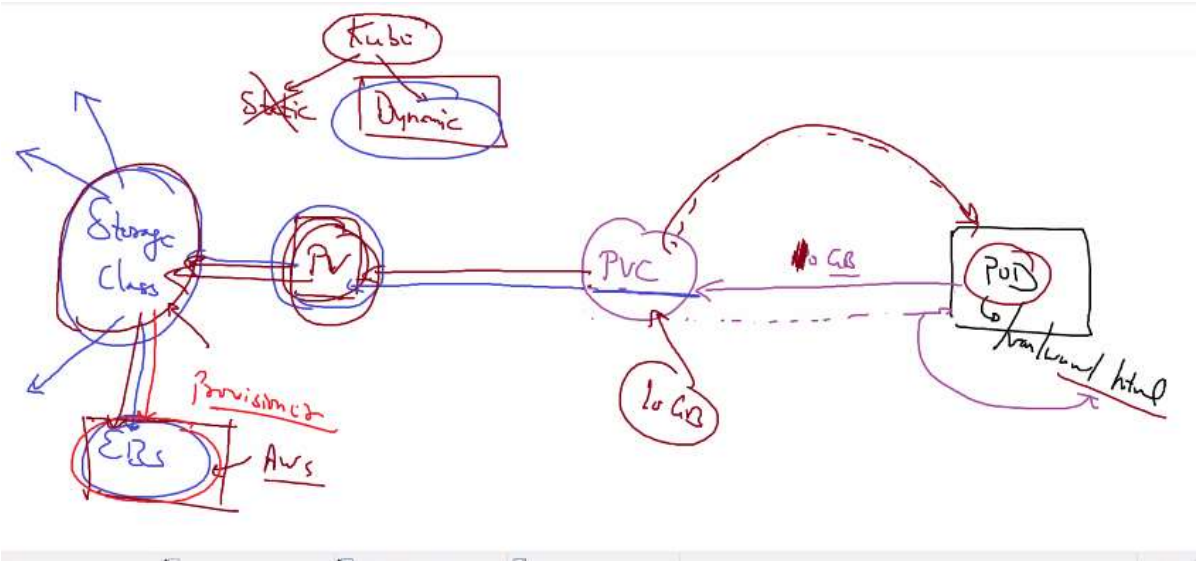
And to make it persistent we have to attach a new storage to it
And in AWS it is called as Volume
And for this e can use 3 types of storage like

- Block (EBS)
- File (EFS)
- Object (S3)

And since we are connect a hard disk we have to use the Block Storage and also the then format and mount it
And this volume here in K8s (as we just want to use the EKS K8s to manage the resources) it is called as Persistent Volume
And the since we are asking for the PV it called as PVC
So the POD will ask for the PVC and the PVC will ask the PV for some space

and creating this PV we have 2 ways:

- Static
- Dynamic



And generally we are using the dynamic
And if we create a PVC then this will automatically that will create the PV

And the PV is asking from some program
As which program is the one that will manage form where we are getting the Physical Storage
And this program is called as Storage Class

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pvc
No resources found in lwns namespace.

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pv
No resources found in lwns namespace.

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get sc
NAME            PROVISIONER             AGE
gp2 (default)   kubernetes.io/aws-ebs   111m
```

And we have one storage class already created for us

```
NAME            PROVISIONER             AGE
gp2 (default)   kubernetes.io/aws-ebs   111m
```

And this storage class will use the EBS or contact to it if anybody asks it for the storage it will automatically give the storage and the volume in the AWS
And this is how they are connected internally the EKS and the AWS services
And for this we don't have any command but we have to use the code part using the YAML file code

And the file looks like somewhat of

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: lwpvc1
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl create -f pvc.yml
persistentvolumeclaim/lwpvc1 created
```

And since we have created the Storage Class we don’t need to create the PV and will be created automatically when we create a PVC

But now if we see that the PV and hard disk is not created
Because the storage class is the one who will do that

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl describe sc gp2
Name:            gp2
IsDefaultClass:  Yes
Annotations:     kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"storage.k8s.io/v1","kind":"Sto
rageClass","metadata":{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"},"name":"gp2"},"para
meters":{"fsType":"ext4","type":"gp2"},"provisioner":"kubernetes.io/aws-ebs","volumeBindingMode":"WaitForFirstC
onsumer"}
,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>
```

And if we describe that in the Annotation we have the Volume Bind Mode set to the Wait for the first Consume
Means it will wait till the consumer is asked or uses the storage until then it will not create (for the 1st time)
And that's why we have the PVC pending

```
Events:                <none>

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get  pvc
NAME     STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
lwpvc1   Pending                                      gp2            2m16s
```

But as soon as the consumer the POD will ask for the storage it will create the PVC and (it will create the PV automatically as the storage class is Specified)
And this can be set in the deployment file (as it is better to set in the deployment instead of POD)

```
kubectl-edit-px7s8 - Notepad
File  Edit  Format  View  Help
      labels:
        app: myweb
    spec:
      volumes:
         - name: web-vol1
           persistentVolumeClaim:
              claimName: lwpvc1
      containers:
      - image: vimal13/apache-webserver-php
        imagePullPolicy: Always
        name: apache-webserver-php
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
```

```
      claimName: lwpvc1
containers:
- image: vimal13/apache-webserver-php
  volumeMounts:
      - mountPath: /var/www/html
        name: web-vol1
```

And now we have to tell the POD that it has to use the PVC
But it is better to say it to the deployment instead of the POD or container
And also tell where to mount in the POD

And this can be changed in the YAML file of the Deployment

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl edit     deploy  myweb
deployment.apps/myweb edited

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get  pods
NAME                     READY   STATUS              RESTARTS   AGE
myweb-79b48fb9f5-xvqdc   1/1     Running             0          21m
myweb-7b7c54778f-rfwwd   0/1     ContainerCreating   0          7s

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get  pvc
NAME     STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
lwpvc1   Bound    pvc-a7529549-6cbd-44a5-80e3-db758969e287   10Gi       RWO            gp2            7m48s
```

And now if we deploy it
Then the PODS are restarted as they are to be make the changes effective
And then we see that the PVC is also created and also if we describe the POD we see that the mount is also done

```
9274ca44118d38f4601c0080a91
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sat, 04 Jul 2020 22:18:23 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-882tg (ro)
      /var/www/html from web-vol1 (rw)
```

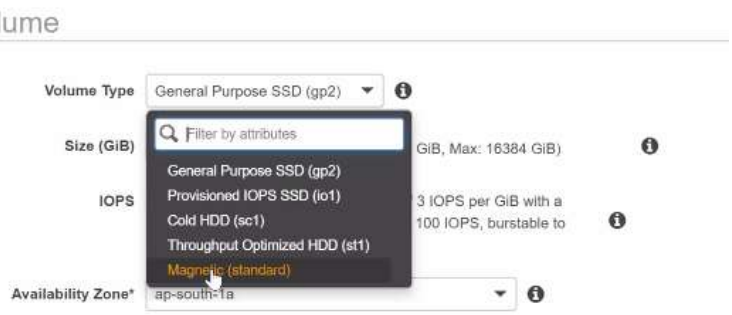
And as we have mounted nothing is there in the mount point at the start by
But as soon as we copy some files then those are added to the storage

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl delete pods --all
pod "myweb-7b7c54778f-rfwwd" deleted

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
myweb-7b7c54778f-vrwt8    1/1     Running   0          15s
```

And now if we have deleted the pod still the data is not lost made persistent

And also when we create the storage in the AWS we have different types



And when we create the PVC we want that the K8s must use the IOPS instead of GP2 (this is by default)

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl describe  sc
Name:            gp2
IsDefaultClass:  Yes
Annotations:     kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"storage.k8s.io/v1","kind":"Sto
rageClass","metadata":{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"},"name":"gp2"},"para
meters":{"fsType":"ext4","type":"gp2"},"provisioner":"kubernetes.io/aws-ebs","volumeBindingMode":"WaitForFirstC
onsumer"}
,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>
```

And for this we have to create an another storage type and for this we have to change the type

And for this we have to create a storage class using the YAML file

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: lwsc1
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
reclaimPolicy: Retain
```

And also if we observe in the PV or Storage Class then we see that the ReClaim Policy is Delete

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM         ST
ORAGECLASS   REASON   AGE
pvc-a7529549-6cbd-44a5-80e3-db758969e287   10Gi       RWO            Delete           Bound    lwns/lwpvc1   gp
2                    10m
```

Means if we delete the PVC then the HDD or Volume is also deleted
And if we want to retain the Volume instead of delete then we have to change the Re Claim Policy to Retain

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  delete pvc --all
persistentvolumeclaim "lwpvc1" deleted

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get pvc
No resources found in lwns namespace.

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM         ST
ORAGECLASS   REASON   AGE
pvc-a7529549-6cbd-44a5-80e3-db758969e287   10Gi       RWO            Delete           Failed   lwns/lwpvc1   gp
2                    12m

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM         ST
ORAGECLASS   REASON   AGE
pvc-a7529549-6cbd-44a5-80e3-db758969e287   10Gi       RWO            Delete           Failed   lwns/lwpvc1   gp
2                    12m

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get pv
No resources found in lwns namespace.
```

And now we have the 2 Storage Classes

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl create  -f sc.yml
storageclass.storage.k8s.io/lwsc1 created

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get sc
NAME            PROVISIONER             AGE
gp2 (default)   kubernetes.io/aws-ebs   138m
lwsc1           kubernetes.io/aws-ebs   6s
```

And in the PVC we never written the Storage class to be used as it will pick the by default Storage Class
But if we want to use the one we have created then we have to add the Storage class Section to tell which on to be used

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: lwpvc1
spec:
  storageClassName: lwsc1
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get pvc
NAME     STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
lwpvc1   Pending                                      lwsc1          4s

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM         ST
ORAGECLASS   REASON   AGE
pvc-a7b85e01-7783-401c-8547-e51dcf018fc3   10Gi       RWO            Retain           Bound    lwns/lwpvc1   lw
sc1                   5s

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get pvc
NAME     STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
lwpvc1   Bound    pvc-a7b85e01-7783-401c-8547-e51dcf018fc3   10Gi       RWO            lwsc1          16s
```

And now here we have created the volume without using or asked by the consumer
And now if we describe this Storage Class then

- We have not written any annotations but was there in the previous one

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get sc
NAME              PROVISIONER             AGE
gp2 (default)     kubernetes.io/aws-ebs   143m
lwsc1 (default)   kubernetes.io/aws-ebs   4m52s

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  edit sc gp2
storageclass.storage.k8s.io/gp2 edited

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  edit sc gp2
Edit cancelled, no changes made.

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get sc
NAME              PROVISIONER             AGE
gp2               kubernetes.io/aws-ebs   144m
lwsc1 (default)   kubernetes.io/aws-ebs   5m29s
```

And if we see that both are default now but this raises to the dilemma so
Then we have to change it in the other one by setting the is-default-class to false

```
storageclass.kubernetes.io/is-default-class: "false"
```

And now if we delete the PVC then we observe that the

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  delete pvc --all
persistentvolumeclaim "lwpvc1" deleted

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get pvc
No resources found in lwns namespace.

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl  get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS     CLAIM
STORAGECLASS   REASON   AGE
pvc-a7b85e01-7783-401c-8547-e51dcf018fc3   10Gi       RWO            Retain           Released   lwns/lwpvc1
lwsc1                   4m23s
```

The PV is retained

```
C:\Users\Vimal Daga\Desktop\eks_class_code>eksctl delete cluster -f cluster.yml
```

And if we use this command then the entire cluster is also will be deleted in one go
As internally it will delete the Cloud Formation template and as this is deleted the entire cluster is also deleted

And if we want to use these commands then we have to have the admin power so for this we have to create a user