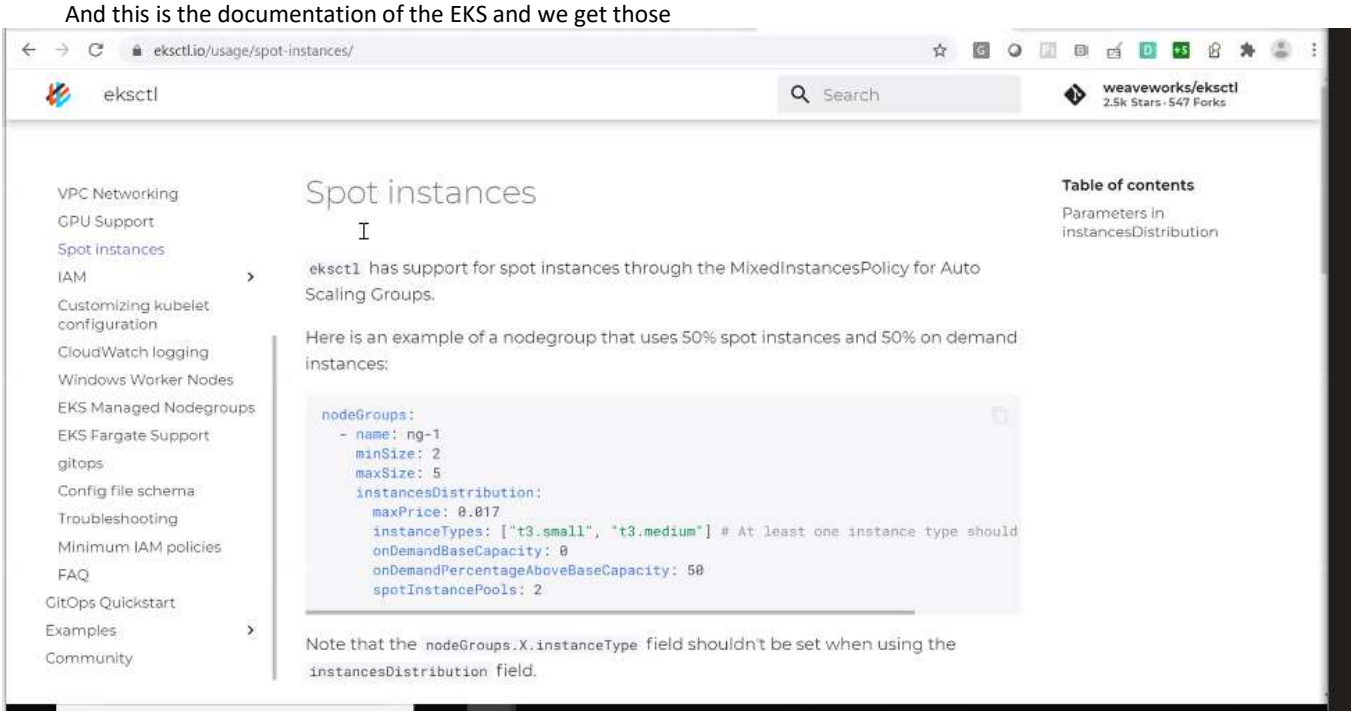
Day 2

05 July 2020 18:25

And also in the cluster we have created we can customize and also about the cost management we can set some limit of the price and also add some notifications needed

And if we use the Spot Instances

And this is the documentation of the EKS and we get those



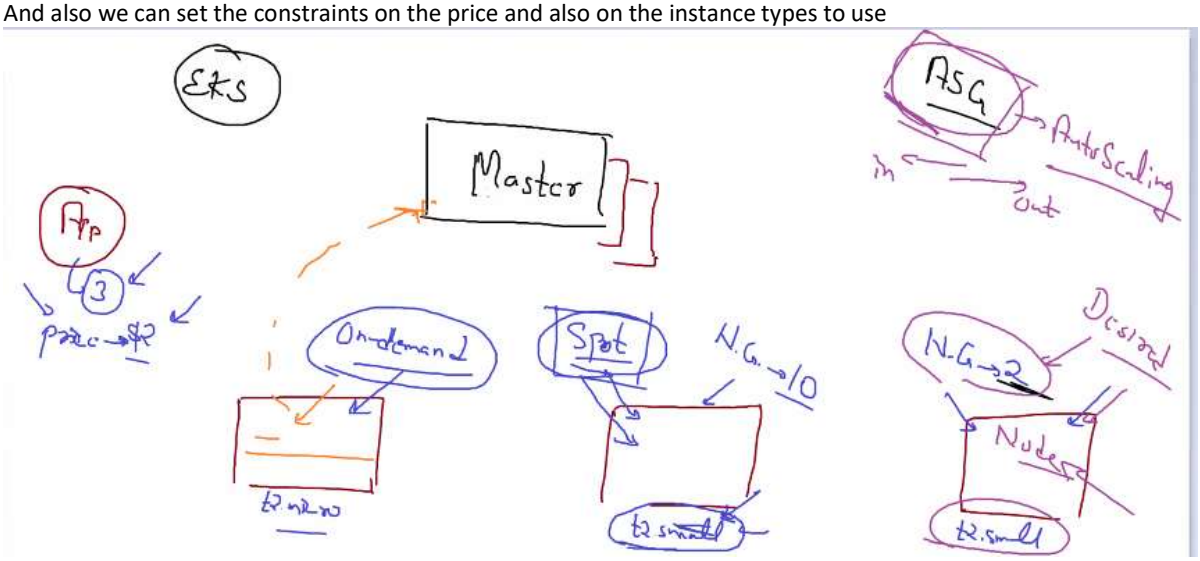
And here we can also have the scaling in the Nodes

Manually or Automatically

And for automatically we use the ASG Automatic Scaling Group

We are using the min and max size or range of the replicas

And also we can set the constraints on the price and also on the instance types to use



```
  - name: ng2
    desiredCapacity: 1
    instanceType: t2.small
    ssh:
       publicKeyName: mykey111222
  - name: ng-mixed
    minSize: 2
    maxSize: 5
    instancesDistribution:
      maxPrice: 0.017
      instanceTypes: ["t3.small", "t3.medium"] # At least one instance type
      onDemandBaseCapacity: 0
      onDemandPercentageAboveBaseCapacity: 50
      spotInstancePools: 2
      ssh:
        publicKeyName: mykey111222
```

Also check about the indentation too

And this kind of planning is called as Capacitative Planning

And technically EKSCTL was an independent tool and now it has been made as the official tool

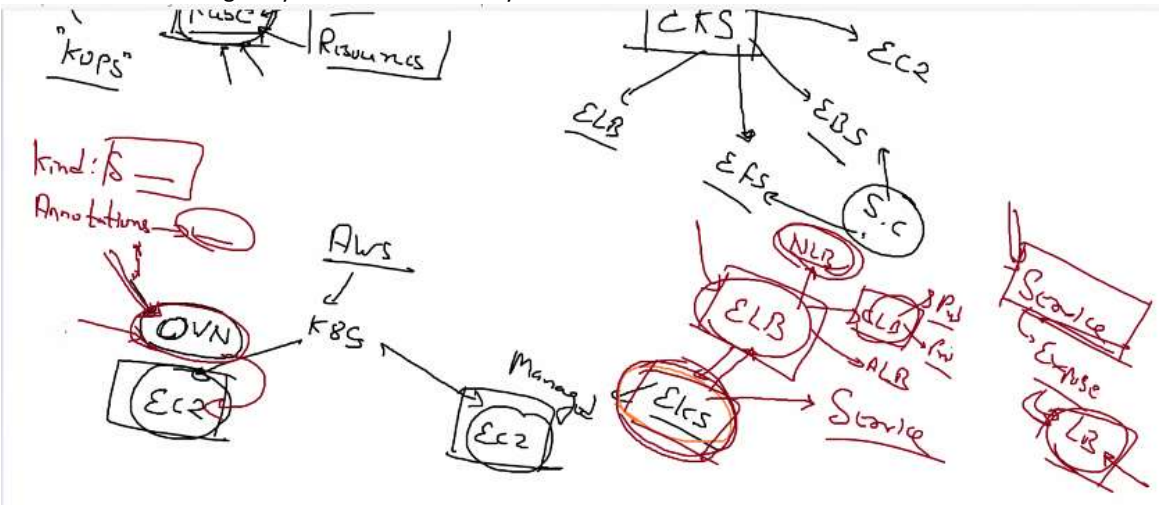
==========

Also we should remember why we need the EKS

- As we need to manage the K8s(which has to be done by someone and companies not so interested part)
- And also get the services and resources of the AWS

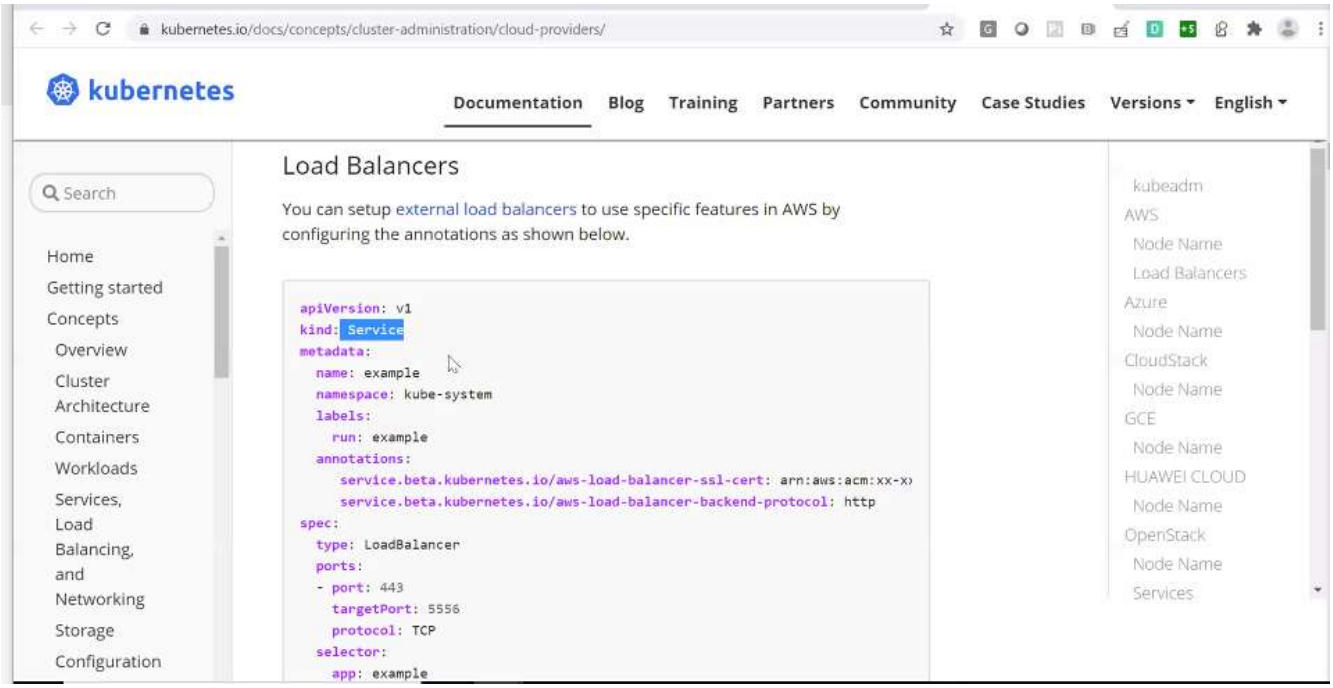
And this K8s cluster we can create our own and manage it on the AWS
Or if we want the management to be done by the AWS we have to use the EKS

And since the AWS is managed it is internally connected or coupled with the AWS services
But since EKS is managed by the AWS we can only connect to the AWS services



But if we want to use or connect with the external or other services then we have to own our own K8s cluster as we are managing so we can change it using some annotations for that

And this is the K8s concept



But if we want to use then we can set the annotation inside the K8s for that

==========

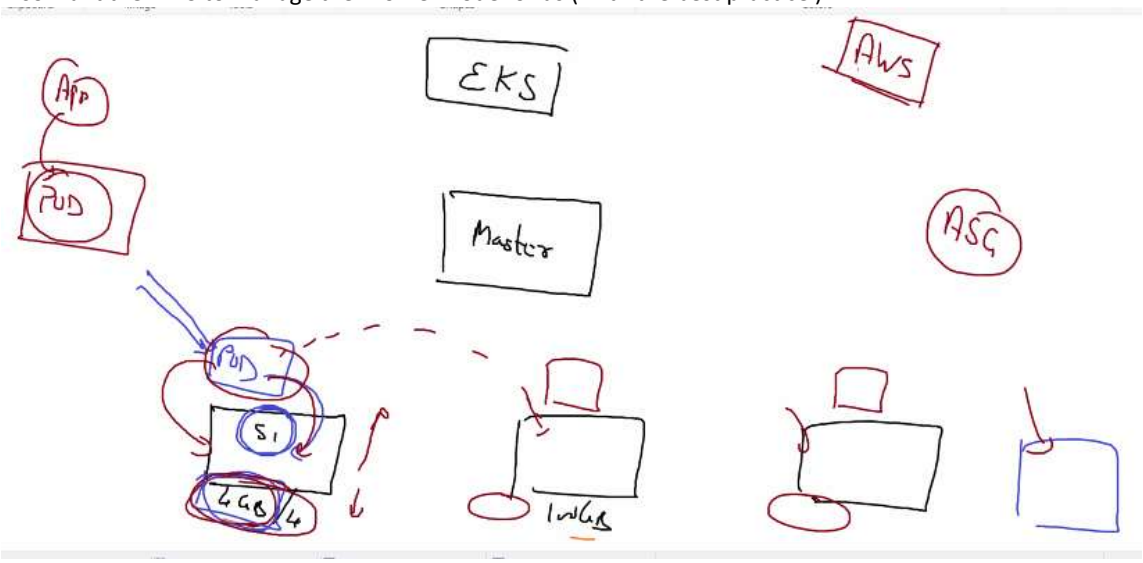
And Also the statement that AWS is managing the EKS or K8s cluster for us is half right and half wrong

As in EKS we have the Master Node that is launched by the AWS is not known to use
But in this cluster it is asking us about the how many instances or nodes we want and also for the resources we have to provide

And also like if we want to perform the scaling we might do it using manually or automatic (but it is horizontally)
But if we want to increase the Resources capacity of the Nodes called as Vertical Scaling we can't perform or might not be supported this

And this is the part where we are concerned

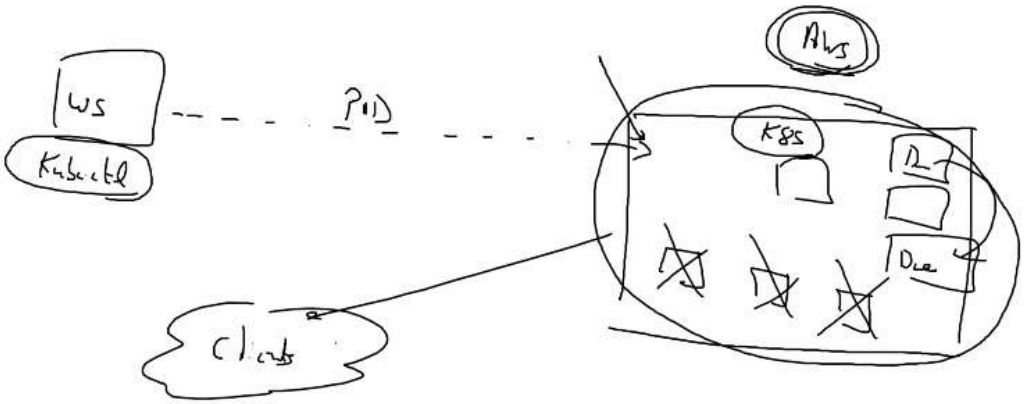
And for this we
Also want the AWS to manage the Worker Node for us (with the best practice)



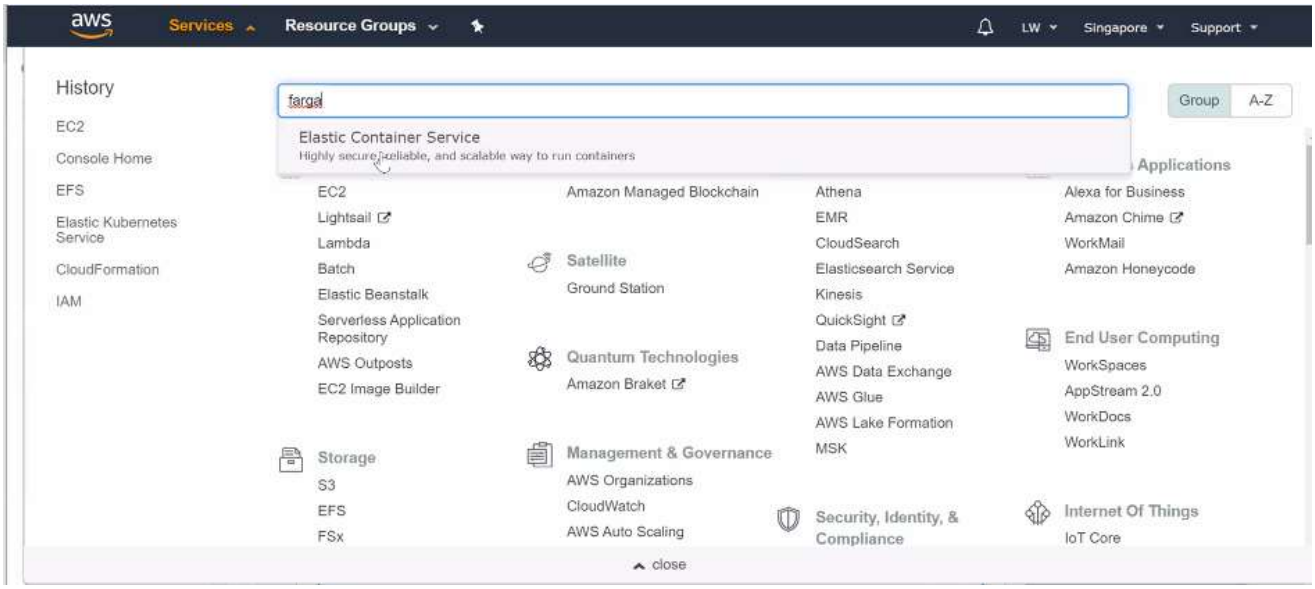
So it is like we need as Black box

- In which we need as K8s
- And we might don't need to know about that we have a Nodes or not and also the resources

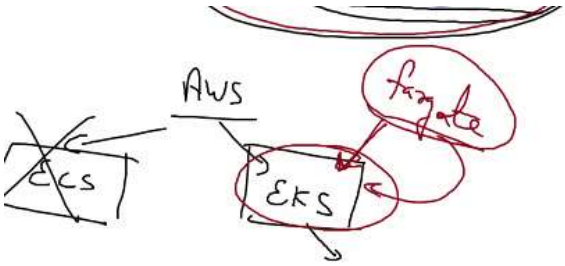
- Or in which availability zones which you are using



- And as per the client perspective we just need the service of the K8s
- And this is the present type of setup is encouraged such that
  - We don't have to manage the server or nodes (any)
  - We can do want we want
  - And as per the requirement they provide us the resources and also management



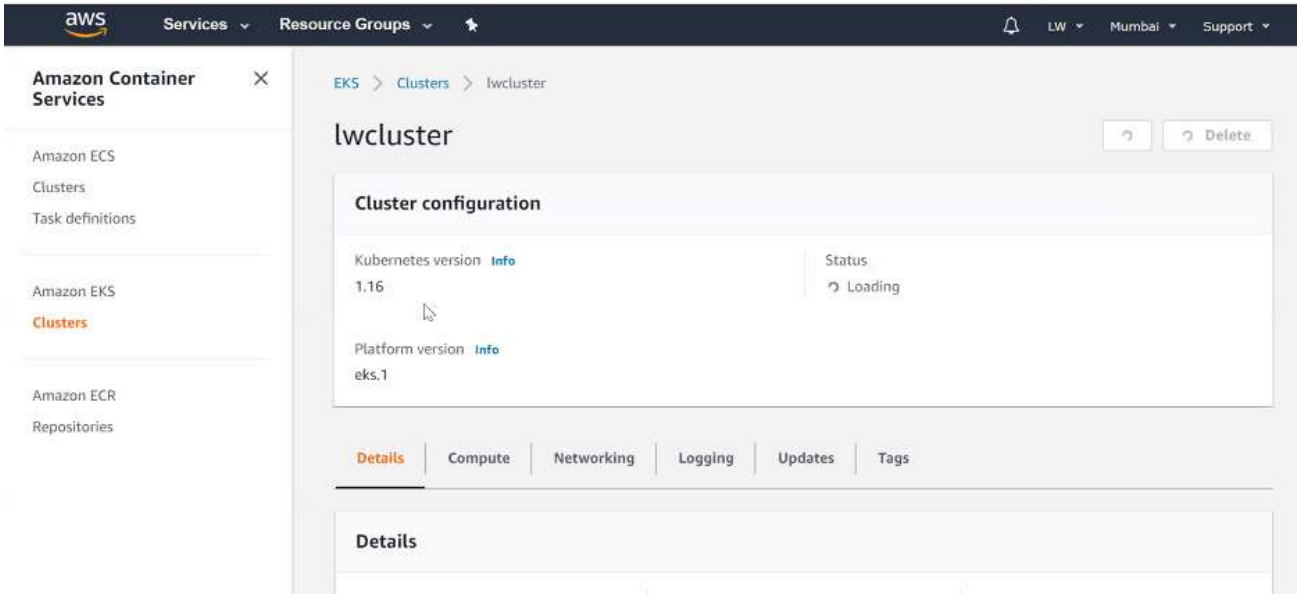
And this service or setup is called as Serverless architecture
And this service is provided by the AWS as Fargate
And this Fargate is a service (internal) of the ECS Elastic Container Service



And here if we want the K8s cluster on AWS

- Half by us and half by AWS managed we use the EKS and we have to write the code we have discussed above
- And if we want entirely managed by the AWS we use the ECS (Fargate)

And that's why if we use the EKS they show us that there is no Fargate profile needed



And we have launched the cluster we can see the there is no fargate used
But if we enable the ECS then they are using the Fargate profile

```
Commands:
  eksctl get cluster                Get cluster(s)
  eksctl get nodegroup              Get nodegroup(s)
  eksctl get iamserviceaccount      Get iamserviceaccount(s)
  eksctl get iamidentitymapping     Get IAM identity mapping(s)
  eksctl get labels                 Get nodegroup labels
  eksctl get fargateprofile         Get Fargate profile(s)

Common flags:
  -C, --color string   toggle colorized logs (valid options: true, false, fabulous) (default
"true")
  -h, --help           help for this command
  -v, --verbose int    set log level, use 0 to silence, 4 for debugging and 5 for debugging w
ith AWS debug logging (default 3)

Use 'eksctl get [command] --help' for more information about a command.


C:\Users\Vimal Daga\Desktop\eks_class_code> eksctl get fargateprofile
Error: --cluster must be set

C:\Users\Vimal Daga\Desktop\eks_class_code>
```

And if we see the help of the eksctl get -h (the help of the get command)

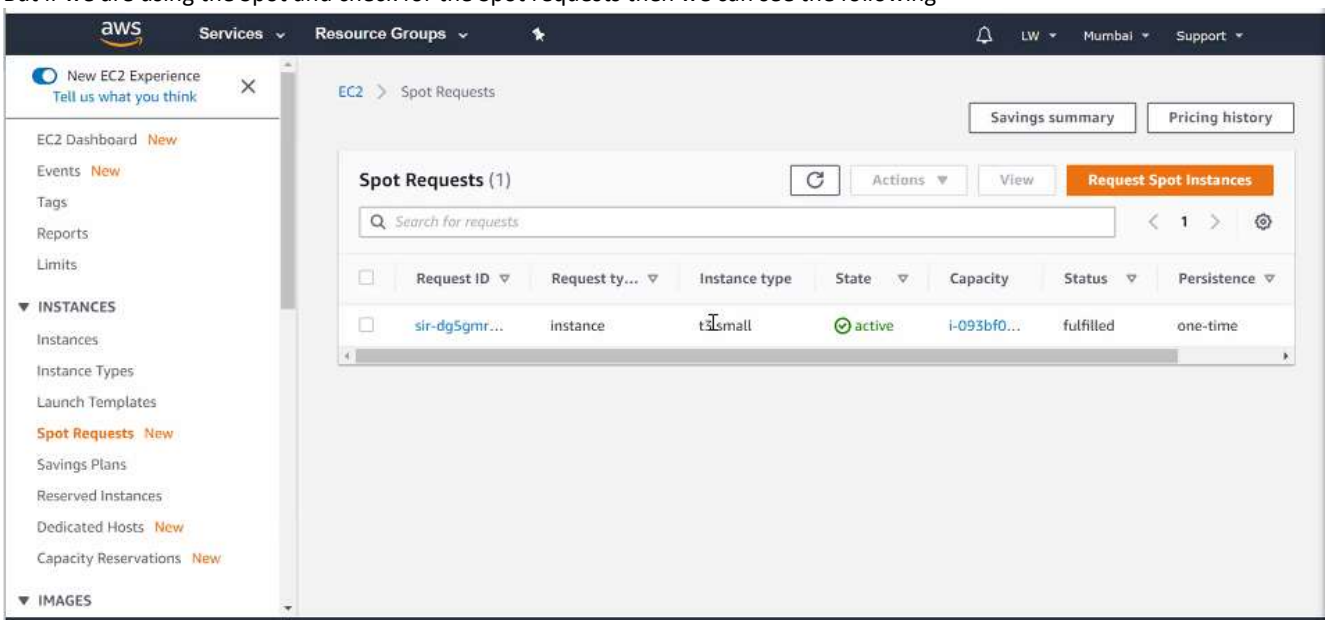
```
C:\Users\Vimal Daga\Desktop\eks_class_code>eksctl get fargateprofile   --cluster lwcluster
Error: failed to get Fargate profile(s) for cluster "lwcluster": AccessDeniedException:
        status code: 403, request id: d05caaff-46cf-42be-be6b-04d5fd8617b1
```

We get that there is no such and is also access is denied

And if we see the information about the

```
C:\Users\Vimal Daga\Desktop\eks_class_code>eksctl get nodegroup   --cluster lwcluster
CLUSTER         NODEGROUP       CREATED                 MIN SIZE        MAX SIZE        DESIRED CAPACITY     IN
STANCE TYPE   IMAGE ID
lwcluster       ng-mixed        2020-07-05T13:27:27Z    2               5               0
              t3.small      ami-073969767527f7306
lwcluster       ng1             2020-07-05T13:27:28Z    2               2               2
              t2.micro        ami-073969767527f7306
lwcluster       ng2             2020-07-05T13:27:28Z    1               1               1
              t2.small        ami-073969767527f7306
```

But if we are using the Spot and check for the Spot requests then we can see the following



And one is launched as we have set 50 - 50

And if we want to scale the Node Group then we can using the following command

```
C:\Users\Vimal Daga\Desktop\eks_class_code>eksctl scale nodegroup   --cluster lwcluster --name ng2 --nodes=3
[ℹ]  scaling nodegroup stack "eksctl-lwcluster-nodegroup-ng2" in cluster eksctl-lwcluster-cluster
[!]  the desired nodes 3 is greater than current nodes-max/maxSize 1
Error: failed to scale nodegroup for cluster "lwcluster", error the desired nodes 3 is greater than current nod
es-max/maxSize 1
```

But it has failed as we have set the maximum limit as 1 in that node Group

```
C:\Users\Vimal Daga\Desktop\eks_class_code>eksctl scale nodegroup   --cluster lwcluster --name ng2 --nodes=3
   --nodes-max=5
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>eksctl scale nodegroup   --cluster lwcluster --name ng2 --nodes=3
   --nodes-max=5
[ℹ]  scaling nodegroup stack "eksctl-lwcluster-nodegroup-ng2" in cluster eksctl-lwcluster-cluster
[ℹ]  scaling nodegroup, desired capacity from 1 to 3, max size from 1 to 5
```

And now we have to change in the config as if we want to connect to the cluster

```
C:\Users\Vimal Daga>aws eks update-kubeconfig --name lwcluster
Updated context arn:aws:eks:ap-south-1:417149810339:cluster/lwcluster in C:\Users\Vimal Daga\
.kube\config
```

And if we want to login inside the Node (we can do it only for the slave as master they are managing it)

```
C:\Users\Vimal Daga\Downloads>ssh -i mykey111222.pem  -l ec2-user 13.127.51.15
The authenticity of host '13.127.51.15 (13.127.51.15)' can't be established.
ECDSA key fingerprint is SHA256:4JtrB4zqFTurYppHd3m5vYgMcE9EvSEUzPCvv8BvqnQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '13.127.51.15' (ECDSA) to the list of known hosts.
Last login: Thu Jun 18 01:20:30 2020 from 205.251.233.50

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
4 package(s) needed for security, out of 10 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-192-168-92-70 ~]$ sudo su - root
[root@ip-192-168-92-70 ~]#
[root@ip-192-168-92-70 ~]#
[root@ip-192-168-92-70 ~]# free -m
              total        used        free      shared  buff/cache   available
Mem:            983         289         108           0         585         568
Swap:             0           0           0
```

```
   Active: active (running) since Sun 2020-07-05 13:31:09 UTC; 23min ago
     Docs: https://docs.docker.com
 Main PID: 3974 (dockerd)
    Tasks: 11
   Memory: 379.2M
   CGroup: /system.slice/docker.service
           └─3974 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jul 05 13:32:21 ip-192-168-92-70.ap-south-1.compute.internal dockerd[3974]: time="2020-07-...
Jul 05 13:32:22 ip-192-168-92-70.ap-south-1.compute.internal dockerd[3974]: time="2020-07-...
Jul 05 13:32:23 ip-192-168-92-70.ap-south-1.compute.internal dockerd[3974]: time="2020-07-...
Jul 05 13:32:24 ip-192-168-92-70.ap-south-1.compute.internal dockerd[3974]: time="2020-07-...
Jul 05 13:32:25 ip-192-168-92-70.ap-south-1.compute.internal dockerd[3974]: time="2020-07-...
Jul 05 13:32:26 ip-192-168-92-70.ap-south-1.compute.internal dockerd[3974]: time="2020-07-...
Jul 05 13:32:27 ip-192-168-92-70.ap-south-1.compute.internal dockerd[3974]: time="2020-07-...
Jul 05 13:32:28 ip-192-168-92-70.ap-south-1.compute.internal dockerd[3974]: time="2020-07-...
Jul 05 13:32:29 ip-192-168-92-70.ap-south-1.compute.internal dockerd[3974]: time="2020-07-...
Jul 05 13:32:30 ip-192-168-92-70.ap-south-1.compute.internal dockerd[3974]: time="2020-07-...
```

And we can see that the docker is running

And also running some kubelet program

```
[root@ip-192-168-92-70 ~]# ps aux | grep kubelet
root      4573  1.1  9.0 864464 91240 ?        Ssl  13:31   0:15 /usr/bin/kubelet --node-ip=1
92.168.92.70 --node-labels=alpha.eksctl.io/cluster-name=lwcluster,alpha.eksctl.io/nodegroup-n
ame=ng1,alpha.eksctl.io/instance-id=i-01805899c6238a55e --max-pods=4 --register-node=true --r
egister-with-taints= --cloud-provider=aws --container-runtime=docker --network-plugin=cni --c
ni-bin-dir=/opt/cni/bin --cni-conf-dir=/etc/cni/net.d --pod-infra-container-image=60240114345
2.dkr.ecr.ap-south-1.amazonaws.com/eks/pause-amd64:3.1 --kubeconfig=/etc/eksctl/kubeconfig.ya
ml --config=/etc/eksctl/kubelet.yaml
root     24817  0.0  0.0 119420   932 pts/0    S+   13:55   0:00 grep --color=auto kubelet
```

And it is not pure serverless in EKS (as if so we can't go inside and see what is happening) but as we have descided the slaves we can go inside it and then

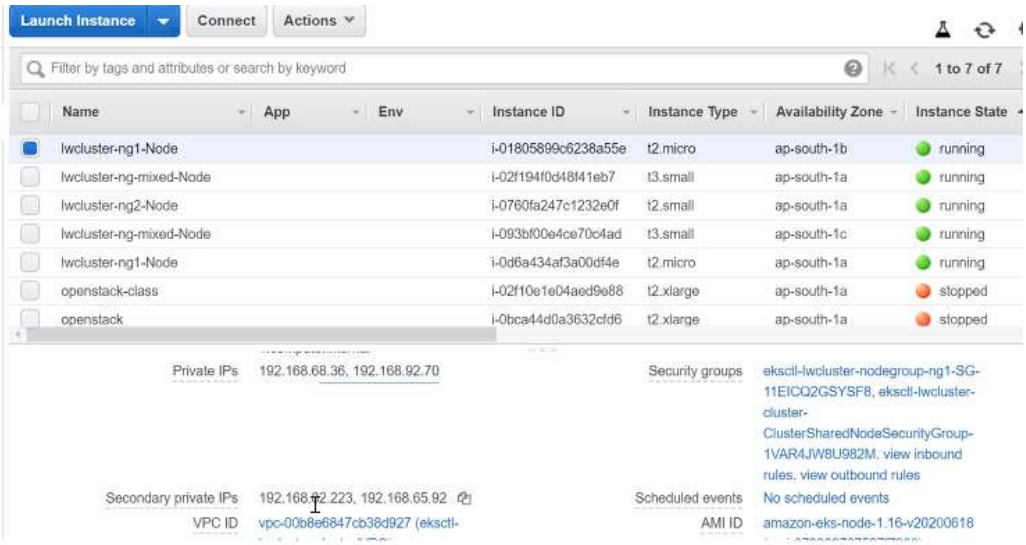
```
 Ssl  13:31   0:15 /usr/bin/kubelet -
ster-name=lwcluster,alpha.eksctl.io/r
c6238a55e --max-pods=4 --register-noc
```

And here we have use the CNI and due to this we can launch only 4 PODS in this NODE as this is due to the limitation of the CNI

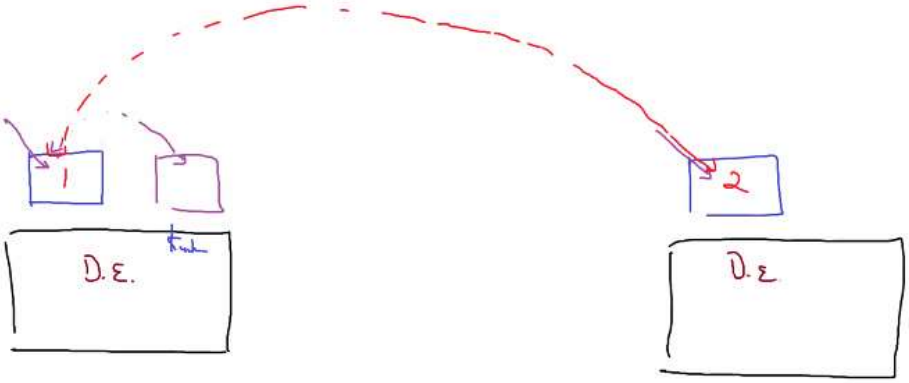
```
3:31   0:14 /usr/bi
ame=lwcluster,alpha
e0f --max-pods=11 -
```

And in another NODE we have more PODS

It is looking like it is based on the instance type



And also in this instance type we have more than 1 Network Cards and also has many Ips (other and main)

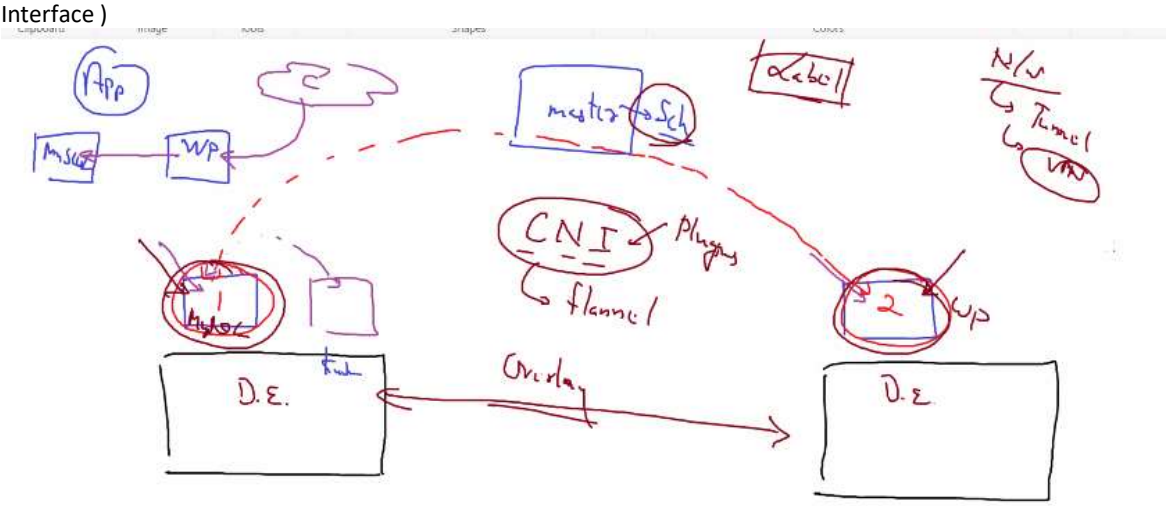


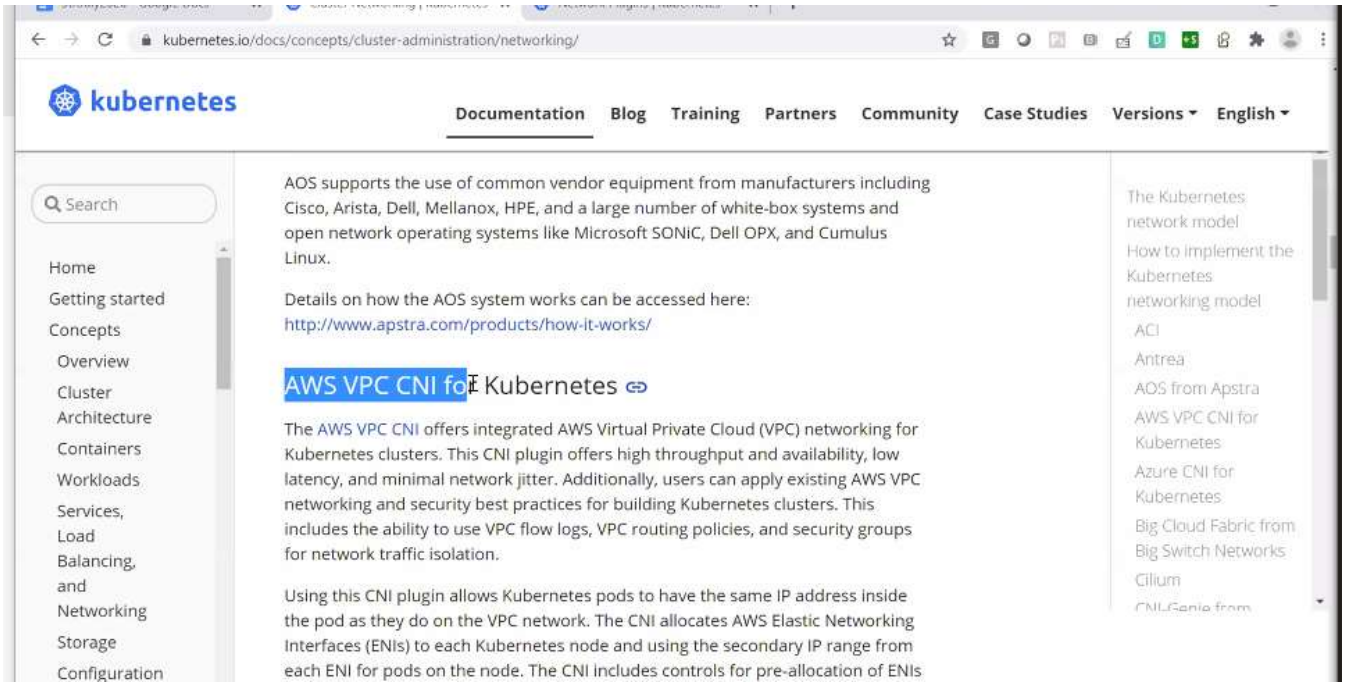
And actually we have 2 slaves nodes and there is no connectivity in between then
And also we have the Docker in both of them and also we have the PODS launched in a node
And if we want to have the connection between the PODS then we can't and this is the by default behaviour of the Container Engines

And these pods are launched but it was decided by the Master node's Scheduler

But if we have the requirement to have such connectivity between then (this is like in case of the Word Press Example)

And if we want the connectivity then we have the concept in Container WORLD called as CNI (an interface Container Network Interface)

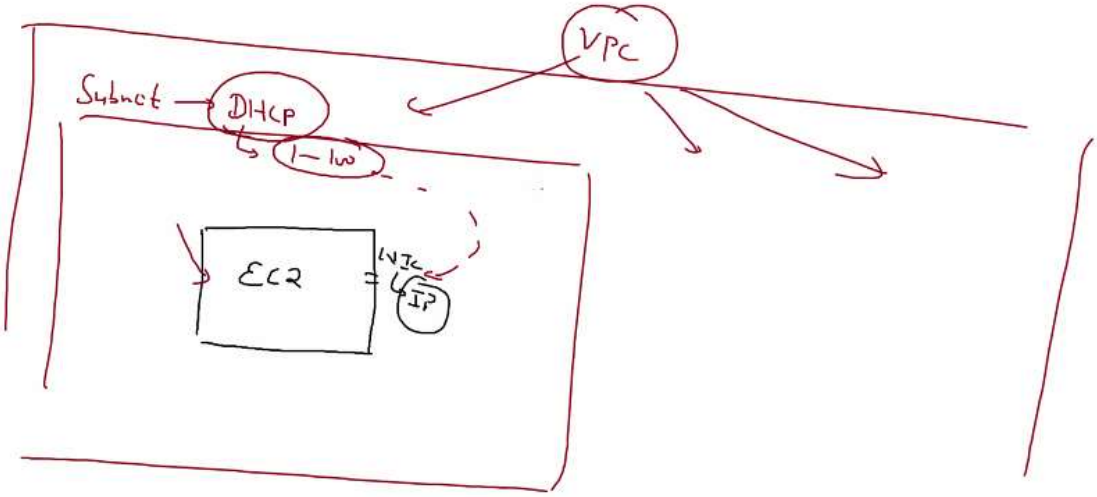


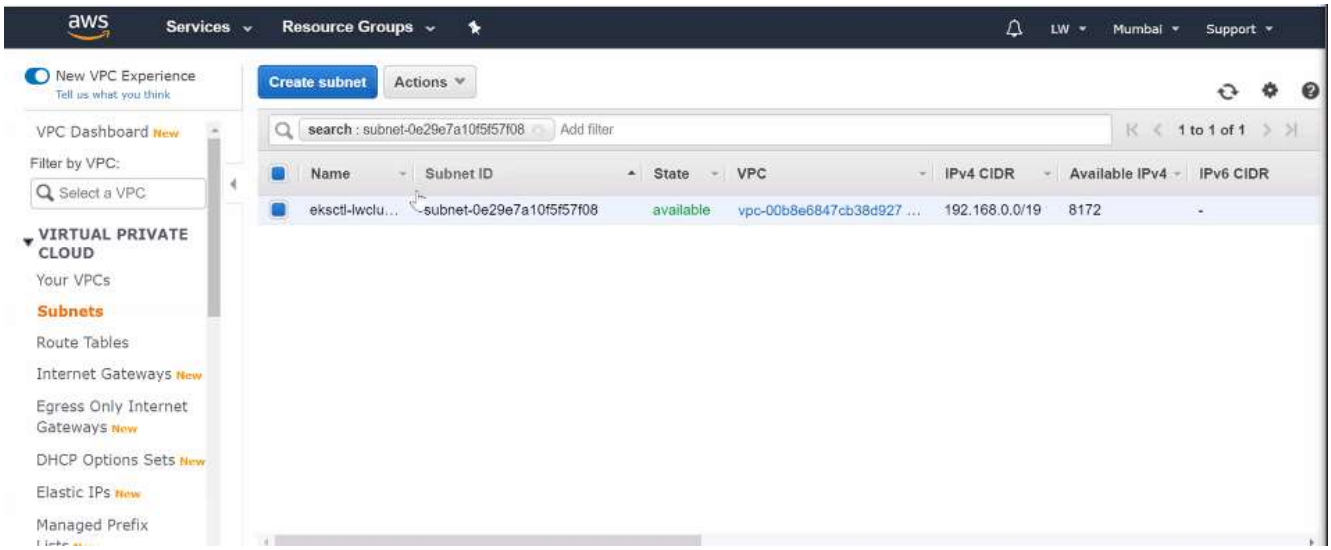


And in K8s we have many such programs that will do this for us and one of the flavour is Flannel
And for this we need the Plugin that will provide the CNI (concept)

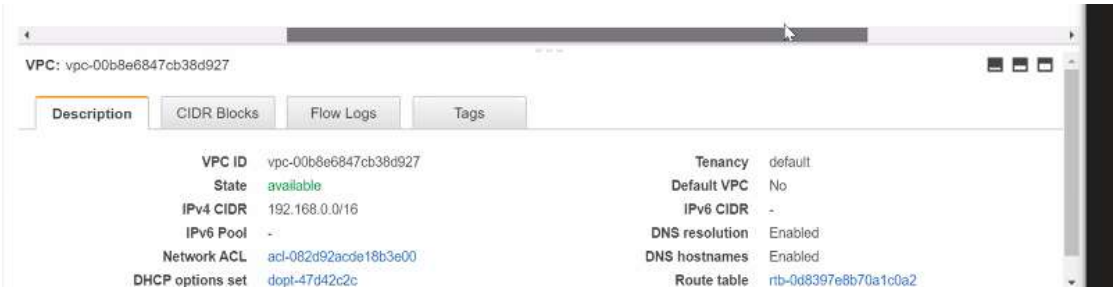
And here if we are managing the K8s cluster then we have to install it
But here the AWS is managing the Master node for us so we have the AWS done this setup for us and for this they are using their own AWS CNI plugin

And we are launching the Node instance in EC2
And it has the IP's but from where it is getting it from
This is got by the Subnet (which is provide us)
And there are many such subnets and these are a part of the VPC





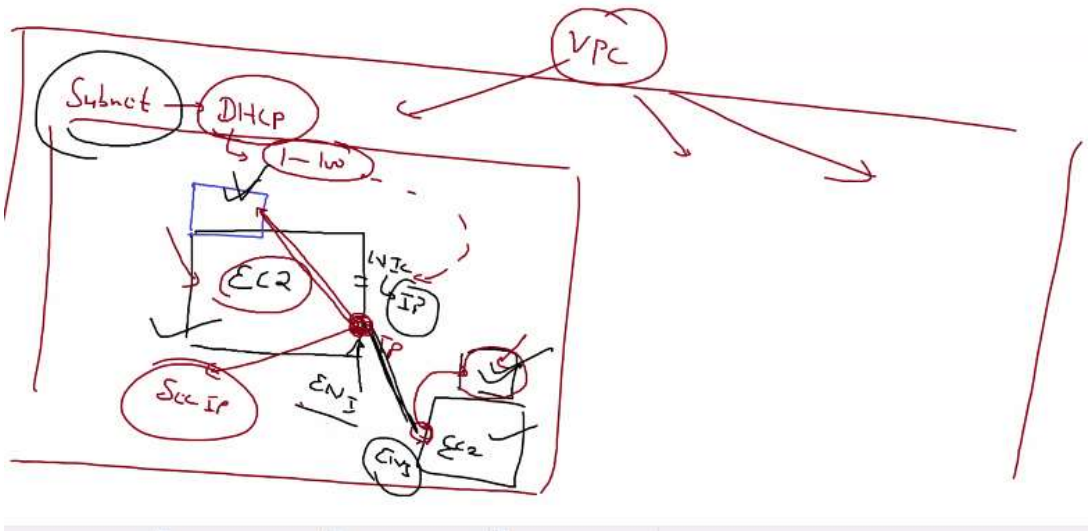
And if we see that the subnet is created for us automatically
And this is created by the EKS when we have used the EKSCTL command



And they have set up some DHCP setup for us and all other and this is the one that will give us the Ip's for us

And they give us many IP addresses for each subnet
But why they do give so many
As if we launch a pod in a node the best way to give it the networking is to attach a Network Card to the node and then attach it to the node

And this extra IP we are giving to the node is called as Secondary IP

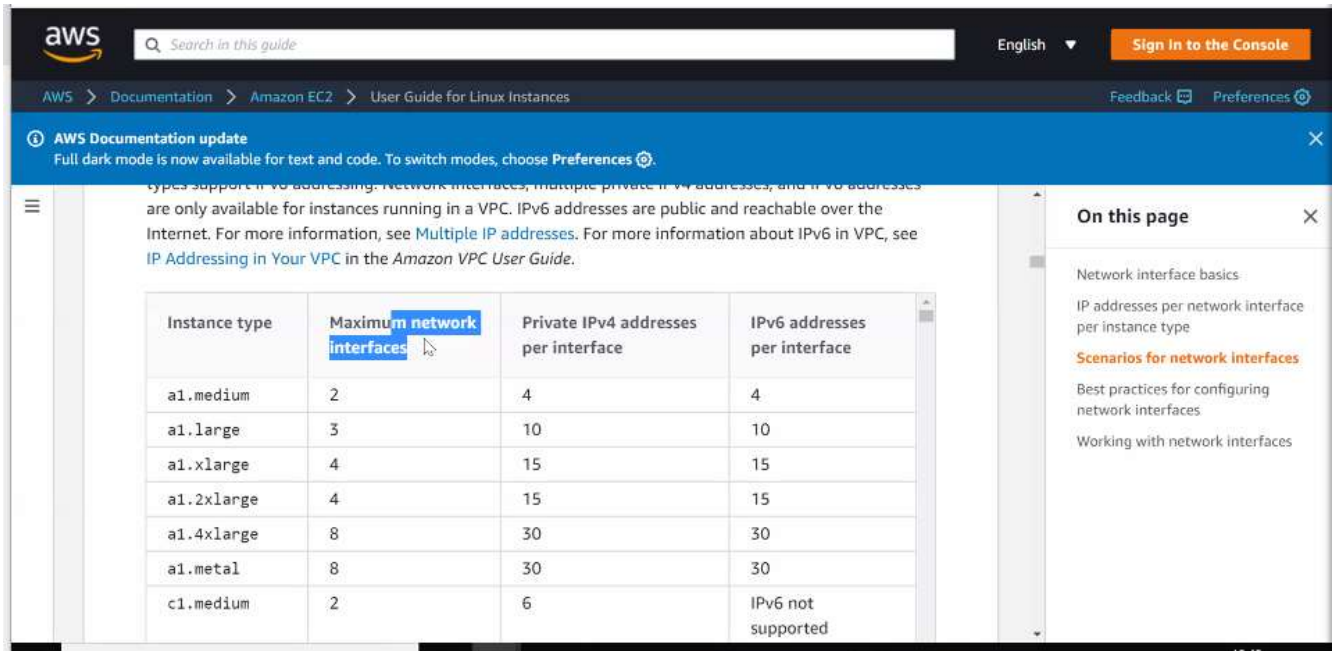


And that's why we have the 3 secondary IP's (for the node with max 4 pods) as we don't have the PODS running but there is internal service that has taken them to use them in future when we launch the nodes

```
[root@ip-192-168-92-70 ~]# ifconfig
eni36ae91284b9: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet6 fe80::64d0:55ff:fe81:bdc9  prefixlen 64  scopeid 0x20<link>
        ether 66:d0:55:81:bd:c9  txqueuelen 0  (Ethernet)
        RX packets 5639  bytes 464676 (453.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5641  bytes 1804034 (1.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eni39ec897ac29: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet6 fe80::463:f2ff:fecc:74e8  prefixlen 64  scopeid 0x20<link>
        ether 06:63:f2:cc:74:e8  txqueuelen 0  (Ethernet)
        RX packets 5585  bytes 460261 (449.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5589  bytes 1787621 (1.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

And we can give the Secondary IP's using the CNI only

And here if we see that we have 4 max PODS can be launched in some Instance types
And there is no issue about the networking or
And this is the limitation of the Instance Types



Instance type	Maximum network interfaces	Private IPv4 addresses per interface	IPv6 addresses per interface
a1.medium	2	4	4
a1.large	3	10	10
a1.xlarge	4	15	15
a1.2xlarge	4	15	15
a1.4xlarge	8	30	30
a1.metal	8	30	30
c1.medium	2	6	IPv6 not supported

and here we can check for the limitations of the instance types

t1.micro	2	2	IPv6 not supported
t2.nano	2	2	2
t2.micro	2	2	2
t2.small	3	4	4
t2.medium	3	6	6
t2.large	3	12	12
t2.xlarge	3	15	15

And for the T2.micro we have 2 max no. of network interfaces we can set
And 2 Ip's per interface so we can just have 4 Ips can be assigned
And since we have 4 Ip's can we assigned we can launch only 4 PODS
And this is because are set in the CNI plugin the AWS is using
And also as they are attaching the real Private Ip's for the Nodes
So we can just launch as many
And this is why planning is very important

And if we want the private profiles then we have to use

fcluster - Notepad

File Edit Format View Help

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

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

fargateProfiles:
  - name: fargate-default
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pods -n kube-system
NAME                       READY   STATUS    RESTARTS   AGE
aws-node-88rdc             1/1     Running   0          33m
aws-node-96jck             1/1     Running   0          54m
aws-node-fkknb             1/1     Running   0          52m
aws-node-gkbz4             1/1     Running   0          53m
aws-node-jwpkj             1/1     Running   0          54m
aws-node-xvwrp             1/1     Running   0          52m
aws-node-zqwpt             1/1     Running   0          32m
coredns-6856799b8d-bwmgh   1/1     Running   0          60m
coredns-6856799b8d-kfb8s   1/1     Running   0          60m
kube-proxy-4zt7s           1/1     Running   0          53m
kube-proxy-8wv2p           1/1     Running   0          54m
kube-proxy-bdgdg           1/1     Running   0          54m
kube-proxy-j8pwt           1/1     Running   0          32m
kube-proxy-l6vxv           1/1     Running   0          52m
kube-proxy-p9mjc           1/1     Running   0          33m
kube-proxy-r6552           1/1     Running   0          52m
```

And these are the PODS running in the namespace (main one)

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pods -n kube-system
NAME                       READY   STATUS    RESTARTS   AGE
aws-node-88rdc             1/1     Running   0          33m
aws-node-96jck             1/1     Running   0          54m
aws-node-fkknb             1/1     Running   0          52m
aws-node-gkbz4             1/1     Running   0          53m
aws-node-jwpkj             1/1     Running   0          54m
aws-node-xvwrp             1/1     Running   0          52m
aws-node-zqwpt             1/1     Running   0          32m
coredns-6856799b8d-bwmgh   1/1     Running   0          60m
coredns-6856799b8d-kfb8s   1/1     Running   0          60m
kube-proxy-4zt7s           1/1     Running   0          53m
kube-proxy-8wv2p           1/1     Running   0          54m
kube-proxy-bdgdg           1/1     Running   0          54m
kube-proxy-j8pwt           1/1     Running   0          32m
kube-proxy-l6vxv           1/1     Running   0          52m
kube-proxy-p9mjc           1/1     Running   0          33m
kube-proxy-r6552           1/1     Running   0          52m
```

And this core DNS in the main pod that is responsible and helping us for the CNI
And all these are the internal services

And when we are planning to launch the Server less Architecture then we have to just mention the which namespace to use for the default purpose

fcluster - Notepad

File Edit Format View Help

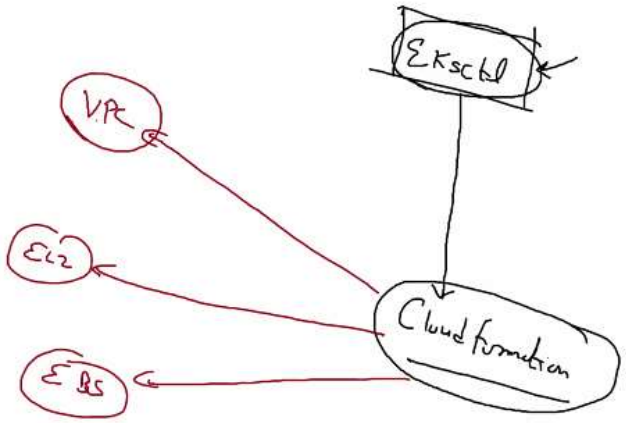
```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

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

fargateProfiles:
  - name: fargate-default
    selectors:
      - namespace: kube-system
      - namespace: default
```

And this file is called as fargate profile

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

And using this we can run the file

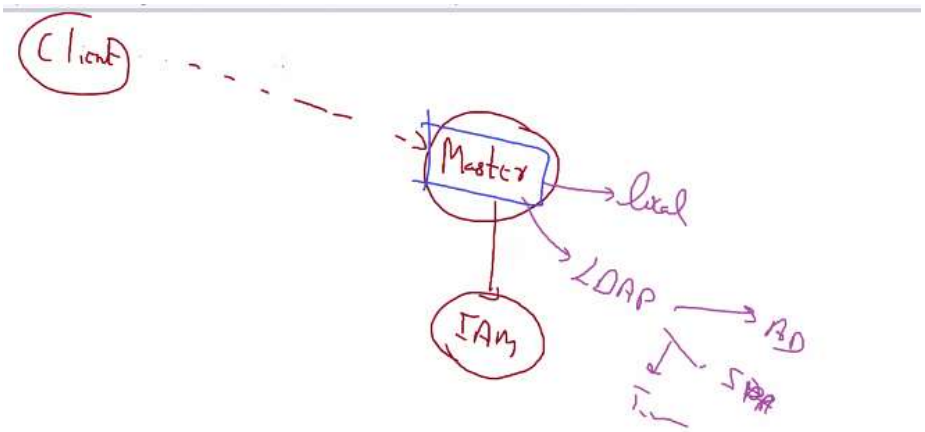
When we run the EKSCTL command (this is just an automation program)
But internally it is using the Cloud Formation that will do everything for us



And in AWS world if one service want to contact to another then they need some power or Permission
And this kind of permission is called as Rule and that is the actual use of Rule

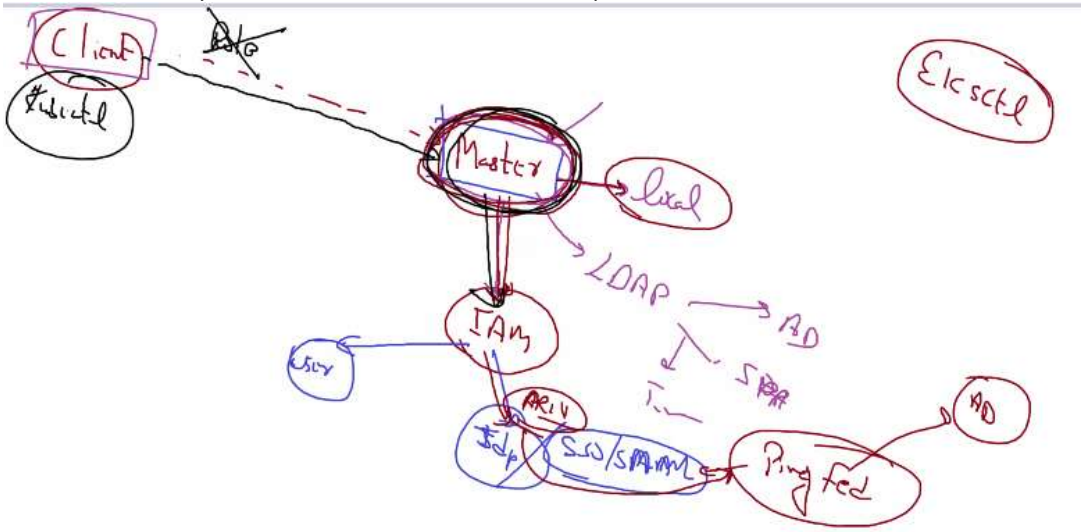
And if we see we need 2 kinds of powers

- Role to use and tell the Cloud Formation
- And the Rule for the power between the services



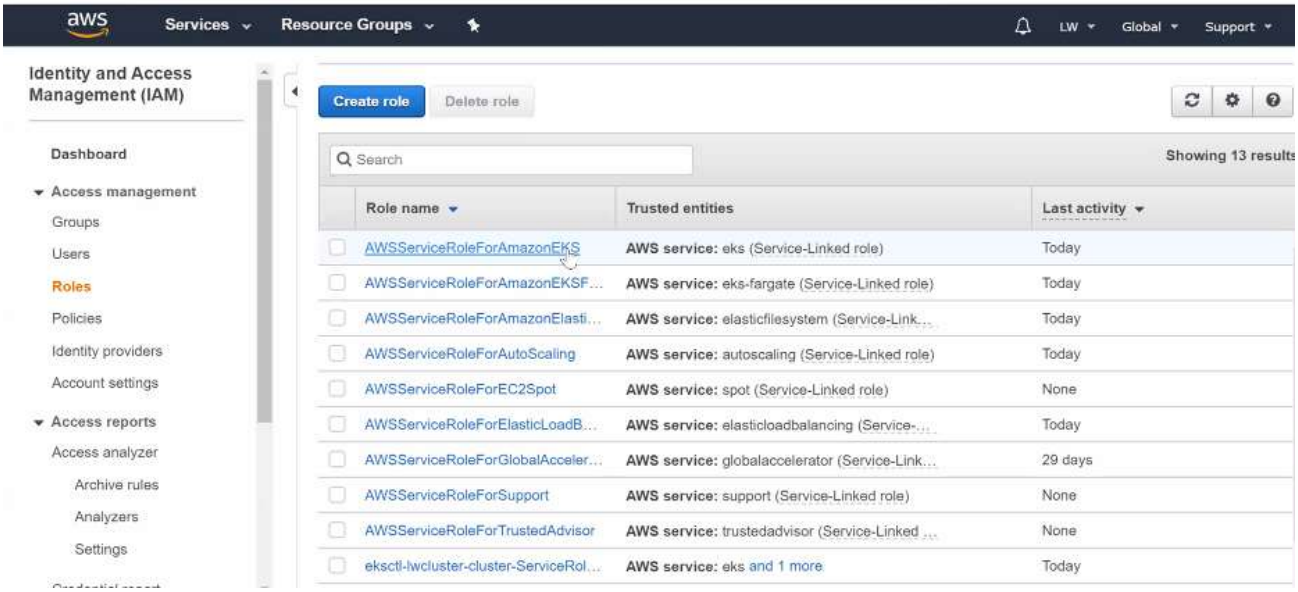
And that is why when we use the eksctl they configure the Master node with the user credentials that we have in the IAM (ROLE) concept

And there are many other features in the IAM that it will provide us

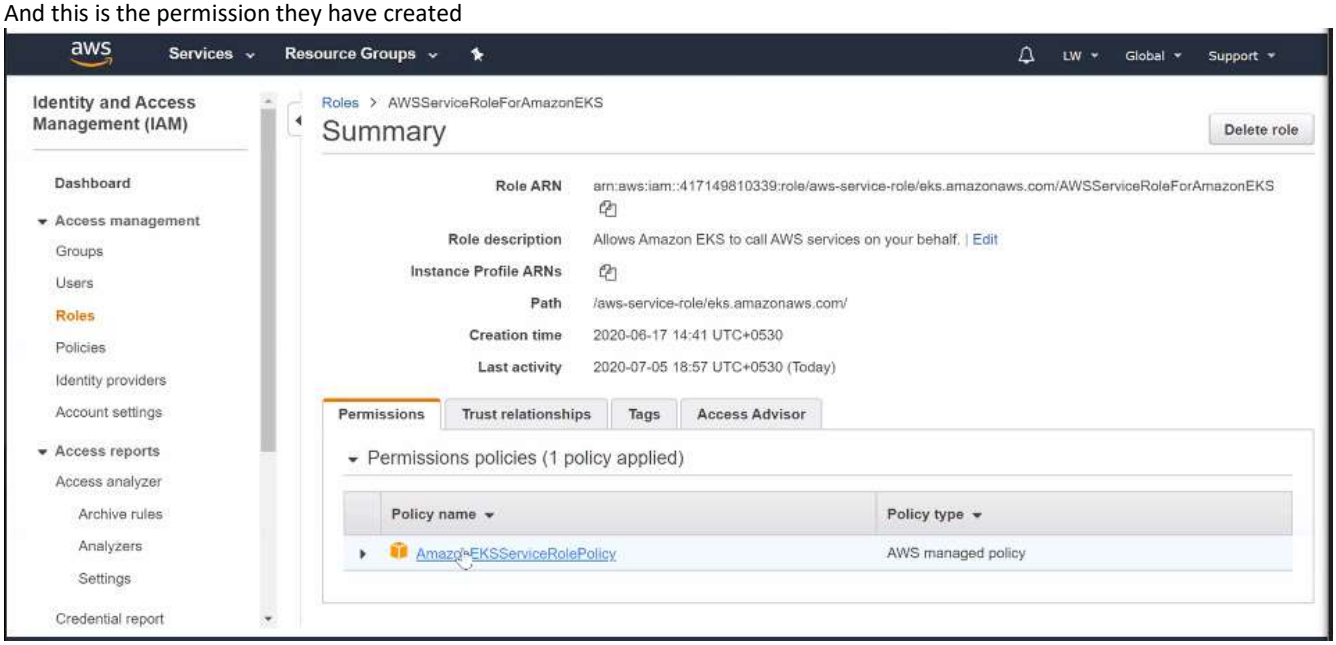


And in this we have some ARN which will help us to integrate with the SSO or SAML

So here the client is the kubectl and the master is using the IAM profile for that
But there is no use or need of ROLE in the K8s to connect the client with that



And there is some role already created for the EKS for us
And this is the permission they have created



And in this way they have the permission to the services to be able to communicate between each other

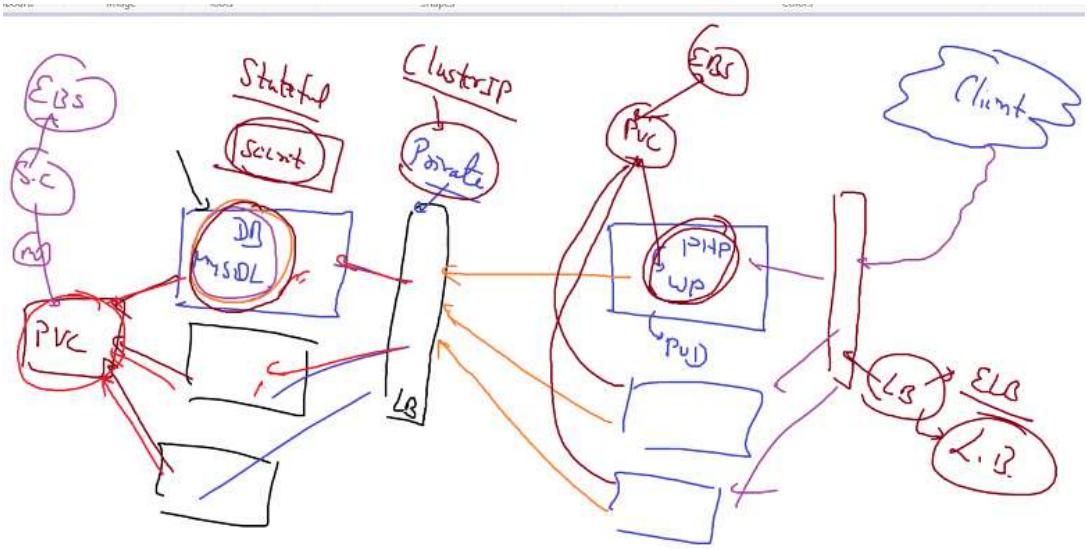
And this is the role they are using for the login
And they don't have the pass word as they get the token when needed

And in the fargate profile we see that they are doing some creation of the Fargate profile

And all these permissions are added to the IAM Role we are using which is created by the EKSCTL

==========

So we would try to create the following setup in the server less architecture on the Example of Word Press we use



And here we have the Word Press and MySQL setup (that we have created in the K8s class of DevOpsAL)

- Where we have a POD for the word press and MySQL and also we have these launch on EC2
- And will have 2 Load Balancers one is internal and one is external
- And we have PVC for both MySQL and Word Press
- And these PVC are launched by the EBS using the Storage class that is created by default

```
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 20Gi
```

And we can change the access modes to ReadWriteMany if we want to have the PVC able to access by the replicas of the same instance

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  type: LoadBalancer
```

And here we have are using the type of the service is called Load Balancer

And in the AWS we have 3 types of the Load balancers
And if we want to use some external load balancer and to tell more about that for this we have to tell the K8s about that using the annotations

```
04-07-2020  01:45    <DIR>          .
04-07-2020  01:45    <DIR>          ..
24-06-2020  18:04               402 cm-pod.yml
04-07-2020  01:23               478 deployment.yml
21-05-2020  02:10               206 kustomization.yaml
05-07-2020  20:21             1,238 mysql-deployment.yaml
22-06-2020  20:43             1,029 mysql.yml
24-06-2020  17:58               363 pod.yml
04-07-2020  16:01               203 pvc.yml
16-06-2020  20:49               379 rc.yml
24-06-2020  22:09               635 rs.yml
22-06-2020  19:50               588 rs_storage.yml
04-07-2020  15:59               168 sc.yml
22-06-2020  20:37               134 secret.yml
23-06-2020  20:10               185 service.yml
05-07-2020  20:29             1,323 wordpress-deployment.yaml
18-06-2020  20:12               346 wp.yml
              15 File(s)          7,677 bytes
               2 Dir(s)   1,788,260,352 bytes free
```

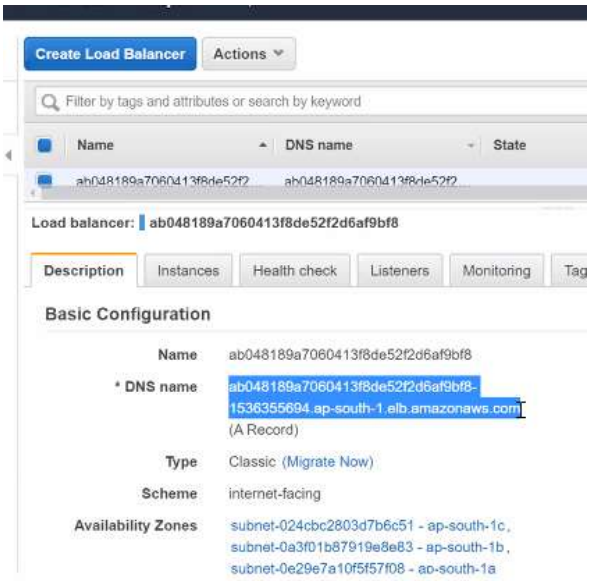
And to tell flow of the file to use we use the concept of the Kustomization in K8s

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
secretGenerator:
- name: mysql-pass
  literals:
  - password=redhat
resources:
  - mysql-deployment.yaml
  - wordpress-deployment.yaml
```

And to run the file we use the following command

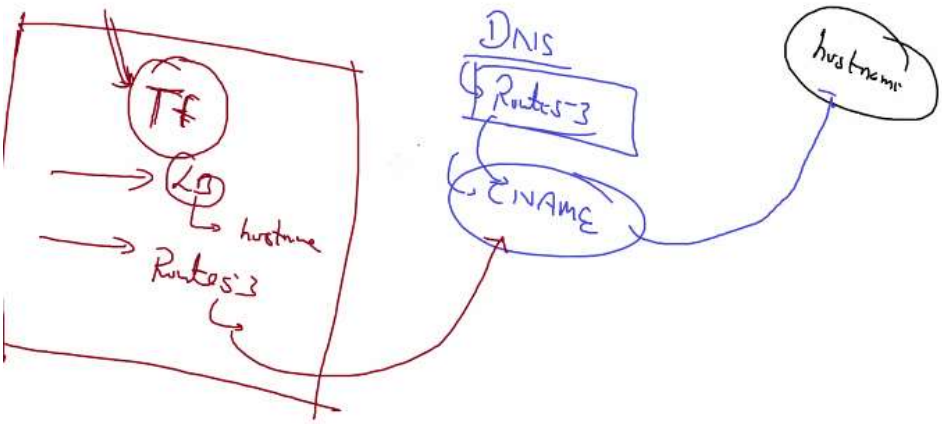
```
C:\Users\Vimal Daga\Desktop\kube_cloud>kubectl create -k .
```

```
C:\Users\Vimal Daga\Desktop\kube_cloud>kubectl create -k .
secret/mysql-pass-ctm2f4889c created
service/wordpress-mysql created
service/wordpress created
deployment.apps/wordpress-mysql created
deployment.apps/wordpress created
persistentvolumeclaim/mysql-pv-claim created
persistentvolumeclaim/wp-pv-claim created
```



And here we have the load balancer given a name and this might change next time and that's why we don't give this name to the client

And for this we have to create or give an host name for the Load balancer



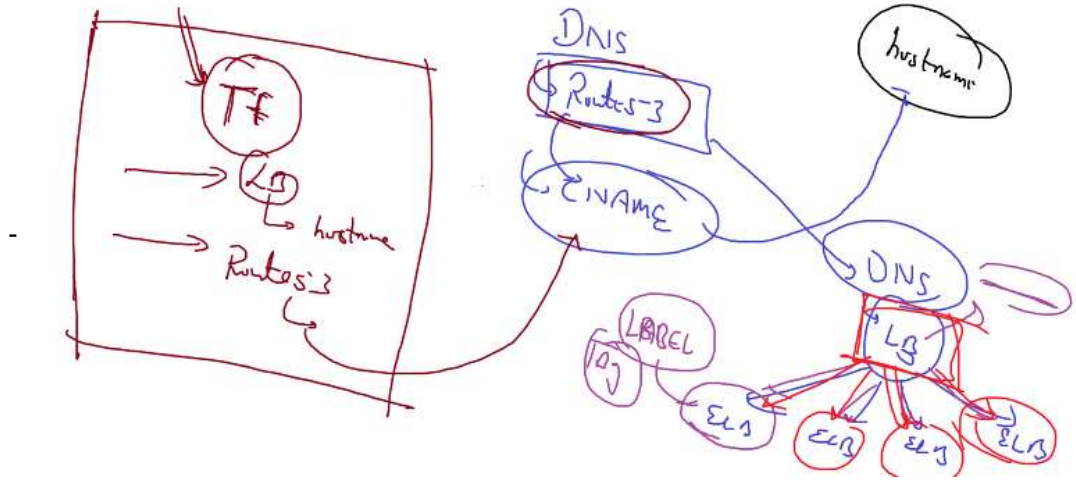
And this we have to update in the DNS record
And for this we have a service in AWS called Route53 and in which we create the CNAME (this is like the Host name)

And if we update the DNS record and if we somehow terminate the Load Balancer then the next time the name changed and also the DNS record we have written fails as it does not find the name we have given

And to solve this we have 2 ways:

- Dynamic
  - In this we can use the Terraform as we can store the value of the current name in some variable and then use it to update the record and this makes it dynamic
  - So in this dynamic world it is better to set the tags as these don't change but the name and hostname will be changed
  - And in terraform as we can manage many things we have the provision to manage these things

- And the other way is



  - Where we have to use the DNS and use a Load balancer and use it to launch another Load balancer

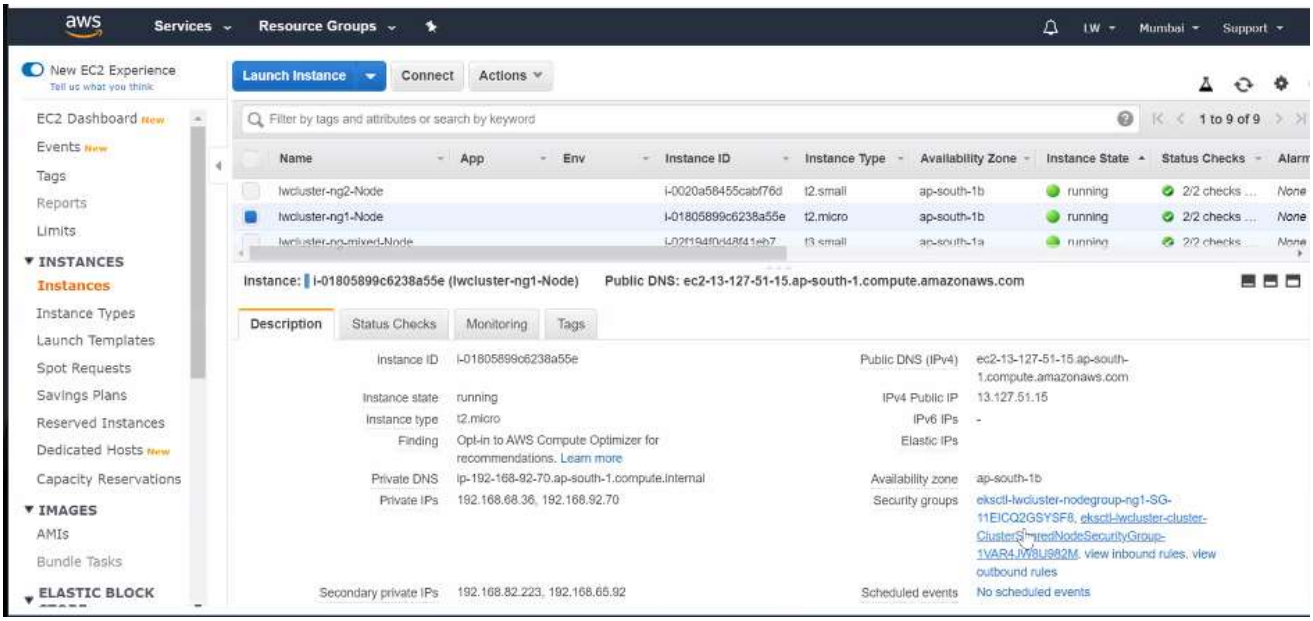
-
- But these are not official but these are the solutions we have

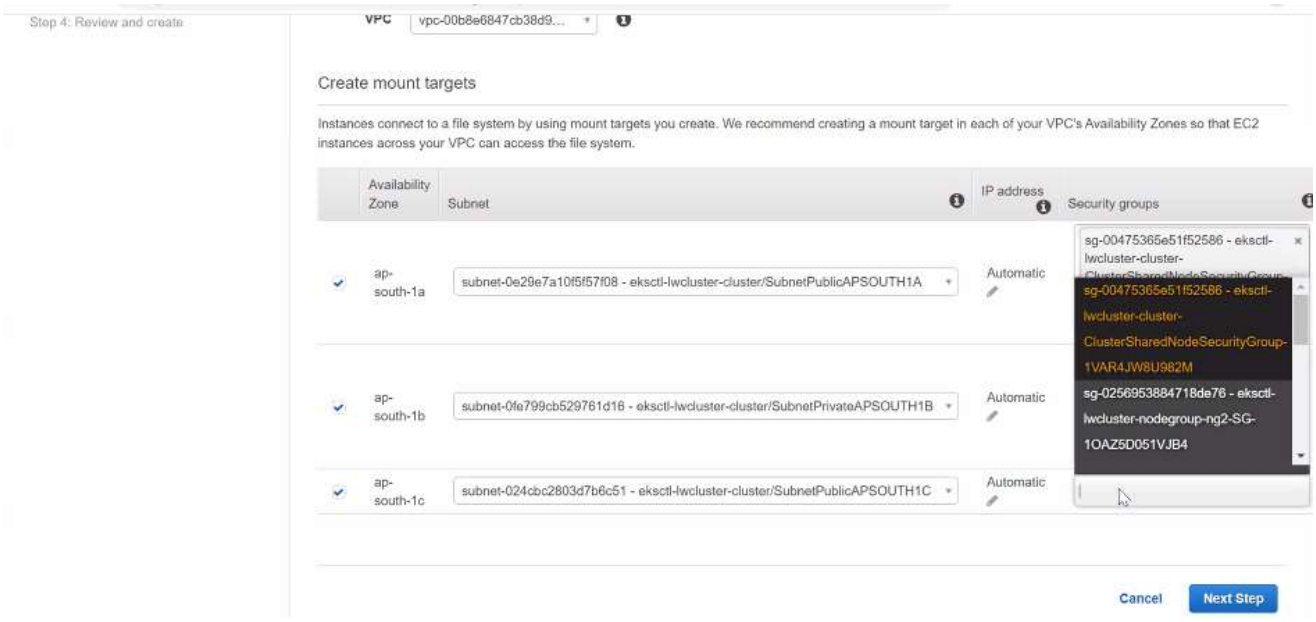
==========

And now there is a problem in the Scaling
As it is just a one command and will be created as desired
But the actually the problem is with the EBS as it is not allowed to access from the other data centre
Like there is a chance as the POD can be launched in anywhere

```
C:\Users\Vimal Daga\Desktop\kube_code>kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
wordpress-88cb86b9b-z59jc         1/1     Running   5          19m
wordpress-mysql-66b4cc9ccb-ztkld  1/1     Running   0          7s

C:\Users\Vimal Daga\Desktop\kube_code>kubectl get pods -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP              NODE
                    NOMINATED NODE   READINESS GATES
wordpress-88cb86b9b-z59jc         1/1     Running   5          19m   192.168.36.156   ip-192-168-60-83.ap-sout
h-1.compute.internal   <none>           <none>
wordpress-mysql-66b4cc9ccb-ztkld  1/1     Running   0          36s   192.168.3.42     ip-192-168-17-34.ap-sout
h-1.compute.internal   <none>           <none>

C:\Users\Vimal Daga\Desktop\kube_code>
```

So to avoid this we have to use the service in AWS called as EFS elastic File System
And while creating this EFS we have to select the same VPC we have and also the same security Group to be used
As then only they can ping or connect to each other





```
[ℹ]  you can enable it with 'eksctl utils update-cluster-logging --region=ap-southeast-1 --cluster=far-lwcluste
r'
[ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "f
ar-lwcluster" in "ap-southeast-1"
[ℹ]  2 sequential tasks: { create cluster control plane "far-lwcluster", no tasks }
[ℹ]  building cluster stack "eksctl-far-lwcluster-cluster"
[ℹ]  deploying stack "eksctl-far-lwcluster-cluster"
[ℹ]  waiting for the control plane availability...
[!]  unable to write kubeconfig , please retry with 'eksctl utils write-kubeconfig -n far-lwcluster': unable to
 read existing kubeconfig file "C:\\Users\\Vimal Daga/.kube/config": error loading config file "C:\Users\Vimal
Daga/.kube/config": read C:\Users\Vimal Daga/.kube/config: The process cannot access the file because another p
rocess has locked a portion of the file.
[ℹ]  no tasks
[ℹ]  all EKS cluster resources for "far-lwcluster" have been created
[ℹ]  creating Fargate profile "fargate-default" on EKS cluster "far-lwcluster"
```

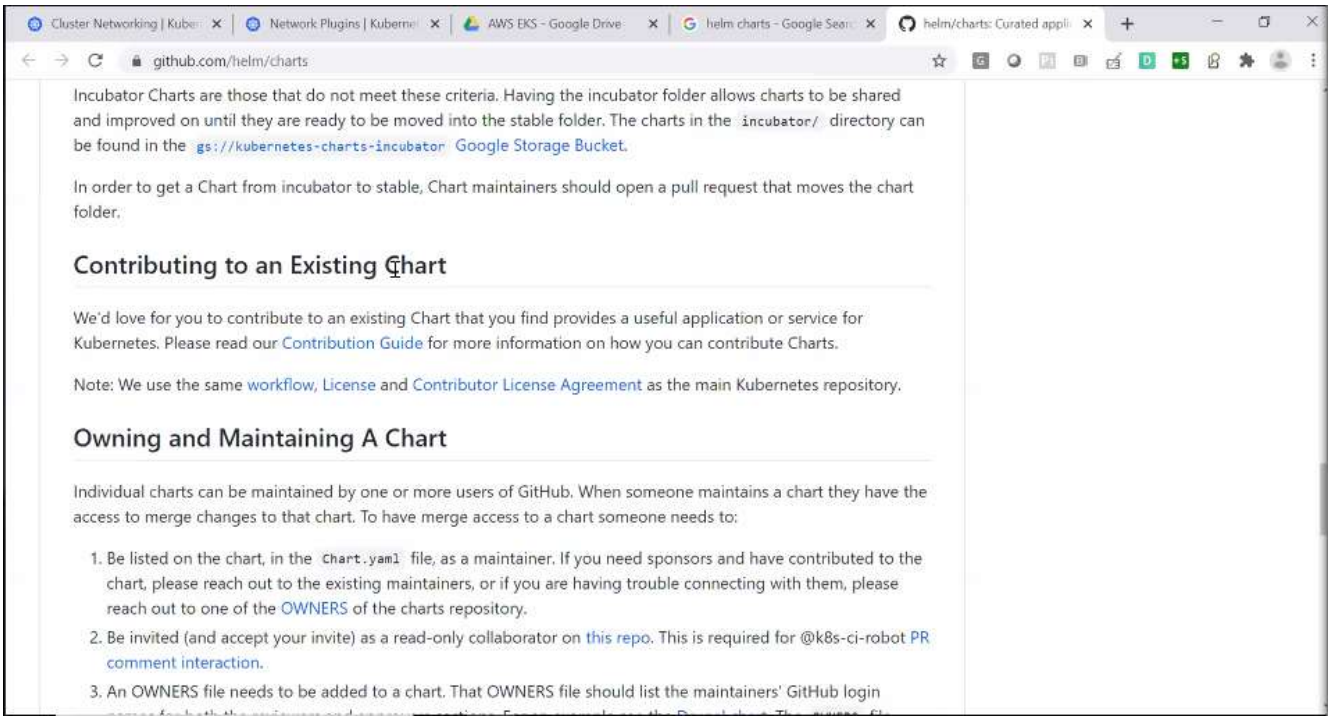
And also here we see the control plane which is the Master Node that is what is managing and creating the service
And they are creating for us

==========

And whatever we create in the K8s or any app we have create or any software we have In the world we have the community guys who have created they try to create the package of all the things we need for that APP

And in K8s we have a such a package manager called as HELM

And they have there own commands (helm the client one)and then we can use it to install anything and all the packages will be installed



And in helm we have many charts that will provide us the info of the many packages etc
And before using the HELM we have to initialize it

```
Command Prompt - helm  init

C:\Users\Vimal Daga\Desktop\kube_code>helm init
$HELM_HOME has been configured at C:\Users\Vimal Daga\.helm.
```

```
C:\Users\Vimal Daga\Desktop\kube_code>helm init
$HELM_HOME has been configured at C:\Users\Vimal Daga\.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated users' policy.
To prevent this, run `helm init` with the --tiller-tls-verify flag.
For more information on securing your installation see: https://v2.helm.sh/docs/securing_installation/

C:\Users\Vimal Daga\Desktop\kube_code>helm repo add stable https://kubernetes-charts.storage.googleapis.com
"stable" has been added to your repositories
```

```
C:\Users\Vimal Daga\Desktop\kube_code>helm repo add stable https://kubernetes-charts.storage.googleapis.com
"stable" has been added to your repositories

C:\Users\Vimal Daga\Desktop\kube_code>helm repo list
NAME    URL
stable  https://kubernetes-charts.storage.googleapis.com

C:\Users\Vimal Daga\Desktop\kube_code>helm repo search -l
```

And now we get all the packages we have
And then we can launch as many packages we want

And here the packages are also called as Charts

```
stable/zeppelin                         1.0.1          0.7.2                                  Web-based noteb
ook that enables data-driven, interactive ...
stable/zeppelin                         1.0.0          0.7.2                                  Web-based noteb
ook that enables data-driven, interactive ...
stable/zetcd                            0.1.9          0.0.3                                  CoreOS zetcd He
lm chart for Kubernetes
stable/zetcd                            0.1.8          0.0.3                                  CoreOS zetcd He
lm chart for Kubernetes
stable/zetcd                            0.1.7          0.0.3                                  CoreOS zetcd He
lm chart for Kubernetes
stable/zetcd                            0.1.6          0.0.3                                  CoreOS zetcd He
lm chart for Kubernetes
stable/zetcd                            0.1.5          0.0.3                                  CoreOS zetcd He
lm chart for Kubernetes
stable/zetcd                            0.1.4          0.0.3                                  CoreOS zetcd He
lm chart for Kubernetes
stable/zetcd                            0.1.3          0.0.3                                  CoreOS zetcd He
lm chart for Kubernetes
stable/zetcd                            0.1.2          0.0.3                                  CoreOS zetcd He
lm chart for Kubernetes
stable/zetcd                            0.1.0          0.0.3                                  CoreOS zetcd He
lm chart for Kubernetes

C:\Users\Vimal Daga\Desktop\kube_code>kubectl create ns lw1
namespace/lw1 created
```

And now if we use the HELM to install any package then

- It will create all the YAML code we need
- And also create everything that the packages need like the PVC, services etc.,
- And this will be done automatically and will be done in one click

```
C:\Users\Vimal Daga\Desktop\kube_code>kubectl get pods -n kube-system
NAME                              READY   STATUS    RESTARTS   AGE
aws-node-88rdc                    1/1     Running   0          154m
aws-node-96jck                    1/1     Running   0          175m
aws-node-fkknb                    1/1     Running   0          173m
aws-node-gkbz4                    1/1     Running   0          174m
aws-node-jwpkj                    1/1     Running   0          175m
aws-node-xvwrp                    1/1     Running   0          173m
aws-node-zqwpt                    1/1     Running   0          153m
coredns-6856799b8d-bwmgh          1/1     Running   0          3h1m
coredns-6856799b8d-kfb8s          1/1     Running   0          3h1m
kube-proxy-4zt7s                  1/1     Running   0          174m
kube-proxy-8wv2p                  1/1     Running   0          175m
kube-proxy-bdgdg                  1/1     Running   0          175m
kube-proxy-j8pwt                  1/1     Running   0          153m
kube-proxy-l6vxv                  1/1     Running   0          173m
kube-proxy-p9mjc                  1/1     Running   0          154m
kube-proxy-r6552                  1/1     Running   0          173m
tiller-deploy-6974685dbc-t8rlv    1/1     Running   0          4m29s
```

And here we might need to provide the tiller more security or power and permission so that it will allow or we can access with the helm

```
# kubectl -n kube-system create serviceaccount tiller

# kubectl create clusterrolebinding tiller --clusterrole cluster-admin
--serviceaccount=kube-system:tiller

# helm init --service-account tiller


# kubectl get pods --namespace kube-system
```

And for this we have to create an account in the tiller and for this we have to run these commands

```
C:\Users\Vimal Daga\Desktop\kube_code>kubectl -n kube-system create serviceaccount tiller
serviceaccount/tiller created

C:\Users\Vimal Daga\Desktop\kube_code>
C:\Users\Vimal Daga\Desktop\kube_code>kubectl create clusterrolebinding tiller --clusterrole cluster-admin --se
rviceaccount=kube-system:tiller
clusterrolebinding.rbac.authorization.k8s.io/tiller created

C:\Users\Vimal Daga\Desktop\kube_code>
C:\Users\Vimal Daga\Desktop\kube_code>
C:\Users\Vimal Daga\Desktop\kube_code>
C:\Users\Vimal Daga\Desktop\kube_code>helm init --service-account tiller
$HELM_HOME has been configured at C:\Users\Vimal Daga\.helm.
Warning: Tiller is already installed in the cluster.
(Use --client-only to suppress this message, or --upgrade to upgrade Tiller to the current version.)

C:\Users\Vimal Daga\Desktop\kube_code>
C:\Users\Vimal Daga\Desktop\kube_code>
C:\Users\Vimal Daga\Desktop\kube_code>helm init --service-account tiller --upgrade
$HELM_HOME has been configured at C:\Users\Vimal Daga\.helm.

Tiller (the Helm server-side component) has been updated to gcr.io/kubernetes-helm/tiller:v2.16.9 .
```

```
C:\Users\Vimal Daga\Desktop\kube_code>kubectl get pods -n kube-system
NAME                             READY   STATUS    RESTARTS   AGE
aws-node-88rdc                   1/1     Running   0          157m
aws-node-96jck                   1/1     Running   0          178m
aws-node-fkknb                   1/1     Running   0          176m
aws-node-gkbz4                   1/1     Running   0          177m
aws-node-jwpkj                   1/1     Running   0          178m
aws-node-xvwrp                   1/1     Running   0          176m
aws-node-zqwpt                   1/1     Running   0          156m
coredns-6856799b8d-bwmgh         1/1     Running   0          3h4m
coredns-6856799b8d-kfb8s         1/1     Running   0          3h4m
kube-proxy-4zt7s                 1/1     Running   0          177m
kube-proxy-8wv2p                 1/1     Running   0          178m
kube-proxy-bdgdg                 1/1     Running   0          178m
kube-proxy-j8pwt                 1/1     Running   0          156m
kube-proxy-l6vxv                 1/1     Running   0          176m
kube-proxy-p9mjc                 1/1     Running   0          157m
kube-proxy-r6552                 1/1     Running   0          176m
tiller-deploy-8488d98b4c-zmh2k   1/1     Running   0          20s
```

And now the tiller program has re-deployed and now we can run the helm to install the packages

```
C:\Users\Vimal Daga\Desktop\kube_code>helm install --name my-release stable/jenkins
```

```
==> v1/Pod(related)
NAME                                READY  STATUS   RESTARTS  AGE
my-release-jenkins-576885ff8b-v5chp  0/2    Pending  0         0s

==> v1/Role
NAME                              AGE
my-release-jenkins-schedule-agents  0s
my-release-jenkins-casc-reload      0s

==> v1/RoleBinding
NAME                               AGE
my-release-jenkins-schedule-agents   0s
my-release-jenkins-watch-configmaps  0s

==> v1/Secret
NAME            TYPE    DATA  AGE
my-release-jenkins  Opaque  2     0s

==> v1/Service
NAME                    TYPE       CLUSTER-IP      EXTERNAL-IP  PORT(S)    AGE
my-release-jenkins        ClusterIP  10.100.181.100  <none>       8080/TCP   0s
my-release-jenkins-agent  ClusterIP  10.100.186.34   <none>       50000/TCP  0s

==> v1/ServiceAccount
NAME                SECRETS  AGE
my-release-jenkins  1        0s
```

And in one click they create everything

`master.customJenkinsLabels`	Append Jenkins labels to the master	`{}`
`master.useSecurity`	Use basic security	`true`
`master.securityRealm`	Jenkins XML for Security Realm	XML for `Leg`
`master.authorizationStrategy`	Jenkins XML for Authorization Strategy	XML for `FullControl`
`master.deploymentLabels`	Custom Deployment labels	Not set
`master.serviceLabels`	Custom Service labels	Not set
`master.podLabels`	Custom Pod labels	Not set
`master.adminUser`	Admin username (and password) created as a secret if useSecurity is true	`admin`
`master.adminPassword`	Admin password (and user) created as a secret if useSecurity is true	Random val
`master.admin.existingSecret`	The name of an existing secret containing the admin credentials.	`""`
`master.admin.userKey`	The key in the existing admin secret containing the username.	`jenkins-adm`

And more many customizations we have to pass many other options and we can know them from the chart (info)

And if we want to setup the entire Prometheus in one click

- We can create our own Configure YAML file or
- We can use the package

```
$ helm install stable/prometheus
```

Introduction

This chart bootstraps a Prometheus deployment on a Kubernetes cluster using the Helm package manager.

Prerequisites

- Kubernetes 1.3+ with Beta APIs enabled

Installing the Chart

To install the chart with the release name `my-release`:

```
$ helm install --name my-release stable/prometheus
```

The command deploys Prometheus on the Kubernetes cluster in the default configuration. The configuration section lists the parameters can be configured during installation.

> Tip: List all releases using `helm list`

Uninstalling the Chart

- And with this command with one click eveything will be done

```
# kubectl create namespace prometheus

# helm install  stable/prometheus     --namespace prometheus     --set
alertmanager.persistentVolume.storageClass="gp2"     --set
server.persistentVolume.storageClass="gp2"
```

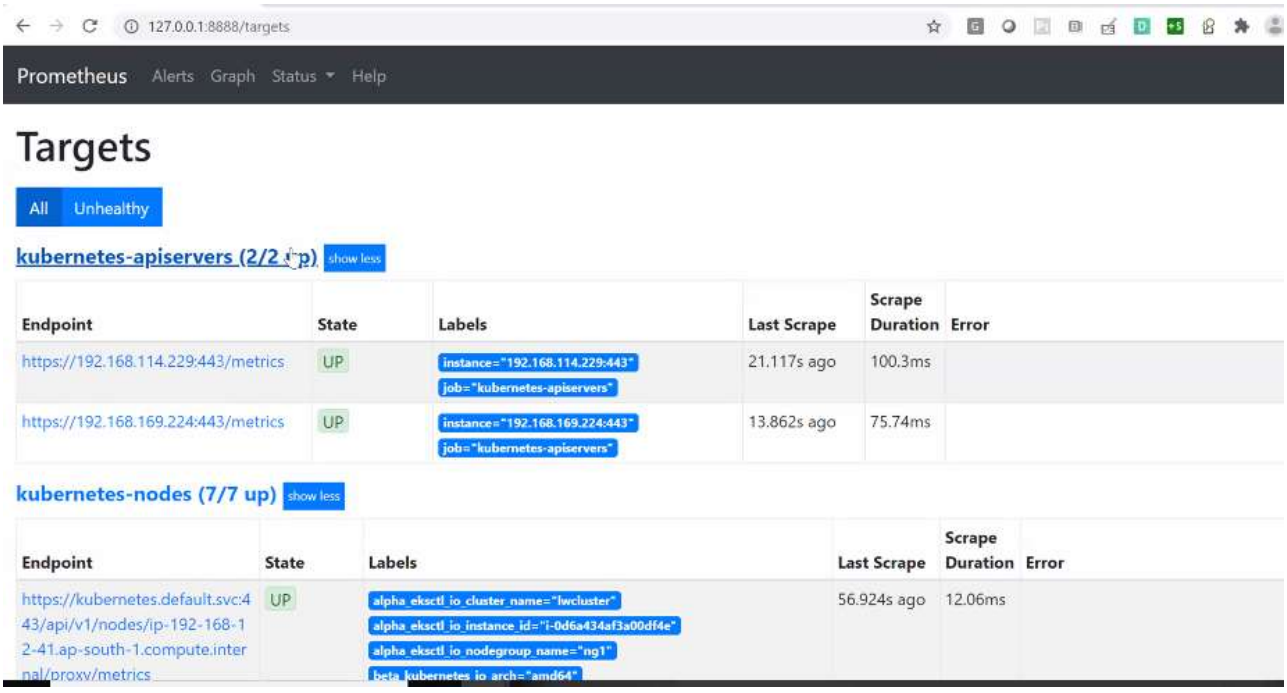
- And this is to say that the HELM has the option to connect with the AWS

```
C:\Users\Vimal Daga\Desktop\kube_code>kubectl get pods -n prometheus
NAME                                                READY   STATUS    RESTARTS   AGE
independent-beetle-kube-state-metrics-778f8b7854-lt8lf        1/1     Running   0          66s
independent-beetle-prometheus-alertmanager-5f64d6cfb6-fjfdc   2/2     Running   0          66s
independent-beetle-prometheus-node-exporter-7t9ss             0/1     Pending   0          66s
independent-beetle-prometheus-node-exporter-9vsfv             1/1     Running   0          66s
independent-beetle-prometheus-node-exporter-bvkjl             1/1     Running   0          66s
independent-beetle-prometheus-node-exporter-mtqlz             1/1     Running   0          66s
independent-beetle-prometheus-node-exporter-srwl9             1/1     Running   0          66s
independent-beetle-prometheus-node-exporter-vg4hx             1/1     Running   0          66s
independent-beetle-prometheus-node-exporter-x57b2             1/1     Running   0          66s
independent-beetle-prometheus-pushgateway-7675d9d9d8-fhfb2    1/1     Running   0          66s
independent-beetle-prometheus-server-7dfbb47dc4-64fxh         1/2     Running   0          66s

C:\Users\Vimal Daga\Desktop\kube_code>kubectl get svc  -n prometheus
NAME                                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
independent-beetle-kube-state-metrics      ClusterIP   10.100.95.8      <none>        8080/TCP   78s
independent-beetle-prometheus-alertmanager ClusterIP   10.100.231.232   <none>        80/TCP     78s
independent-beetle-prometheus-node-exporter ClusterIP  None             <none>        9100/TCP   78s
independent-beetle-prometheus-pushgateway  ClusterIP   10.100.138.169   <none>        9091/TCP   78s
independent-beetle-prometheus-server       ClusterIP   10.100.23.14     <none>        80/TCP     78s
```

And if we want to connect to the Prometheus we have launched we have to use the server IP

And it is in the public world
And if we want to connect it to the from the private world we have to perform the port forward such that we can connect it to the Prometheus in the Public World

```
C:\Users\Vimal Daga\Desktop\kube_code>kubectl -n prometheus  port-forward svc/independent-beetle-prometheus-ser
ver      8888:80
Forwarding from 127.0.0.1:8888 -> 9090
Forwarding from [::1]:8888 -> 9090
Handling connection for 8888
Handling connection for 8888
Handling connection for 8888
Handling connection for 8888
Handling connection for 8888
Handling connection for 8888
```



And see this is the Prometheus and we have many targets already created for us

And if we see some pending in the EKS PODS the reason might be due to the limitation of the no. of PODS in the nodes they have been launched

```
C:\Users\Vimal Daga\Desktop\kube_code>kubectl get pods -n prometheus
NAME                                                READY   STATUS    RESTARTS   AGE
independent-beetle-kube-state-metrics-778f8b7854-lt8lf        1/1     Running   0          5m48s
independent-beetle-prometheus-alertmanager-5f64d6cfb6-fjfdc   2/2     Running   0          5m48s
independent-beetle-prometheus-node-exporter-7t9ss             0/1     Pending   0          5m48s
independent-beetle-prometheus-node-exporter-9vsfv             1/1     Running   0          5m48s
independent-beetle-prometheus-node-exporter-bvkjl             1/1     Running   0          5m48s
independent-beetle-prometheus-node-exporter-mtqlz             1/1     Running   0          5m48s
independent-beetle-prometheus-node-exporter-srwl9             1/1     Running   0          5m48s
independent-beetle-prometheus-node-exporter-vg4hx             1/1     Running   0          5m48s
independent-beetle-prometheus-node-exporter-x57b2             1/1     Running   0          5m48s
independent-beetle-prometheus-pushgateway-7675d9d9d8-fhfb2    1/1     Running   0          5m48s
independent-beetle-prometheus-server-7dfbb47dc4-64fxh         2/2     Running   0          5m48s
```

HELM is purely the topic of the K8s and we can use the HELM anyways

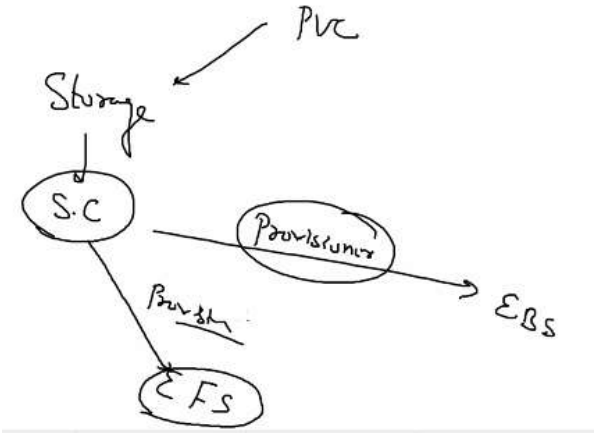
And if the Storage Class is the one that will create the PVC and the PVC is using some type of service form the AWS like EBS to create the storage
So for this we need to have provisioner for the SC to connect to such a service
And the same is needed for the EFS to be access by the SC
And also we need to provide the permission or role to have the access to use the EFS

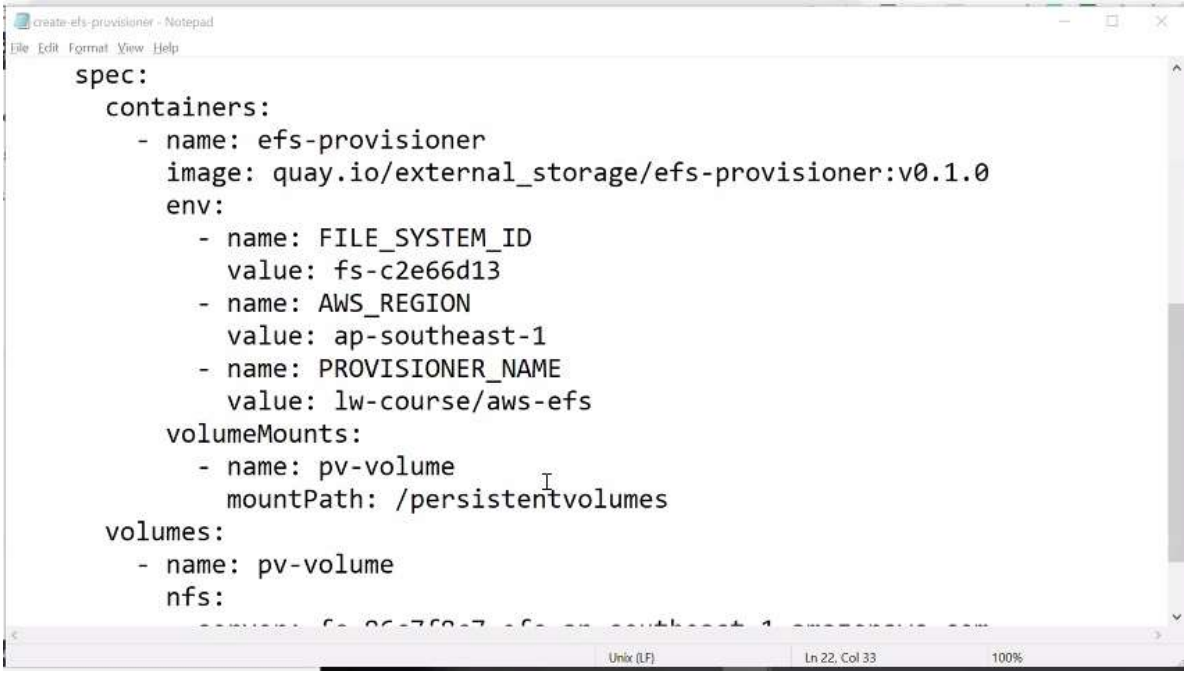
And this is for the Permission

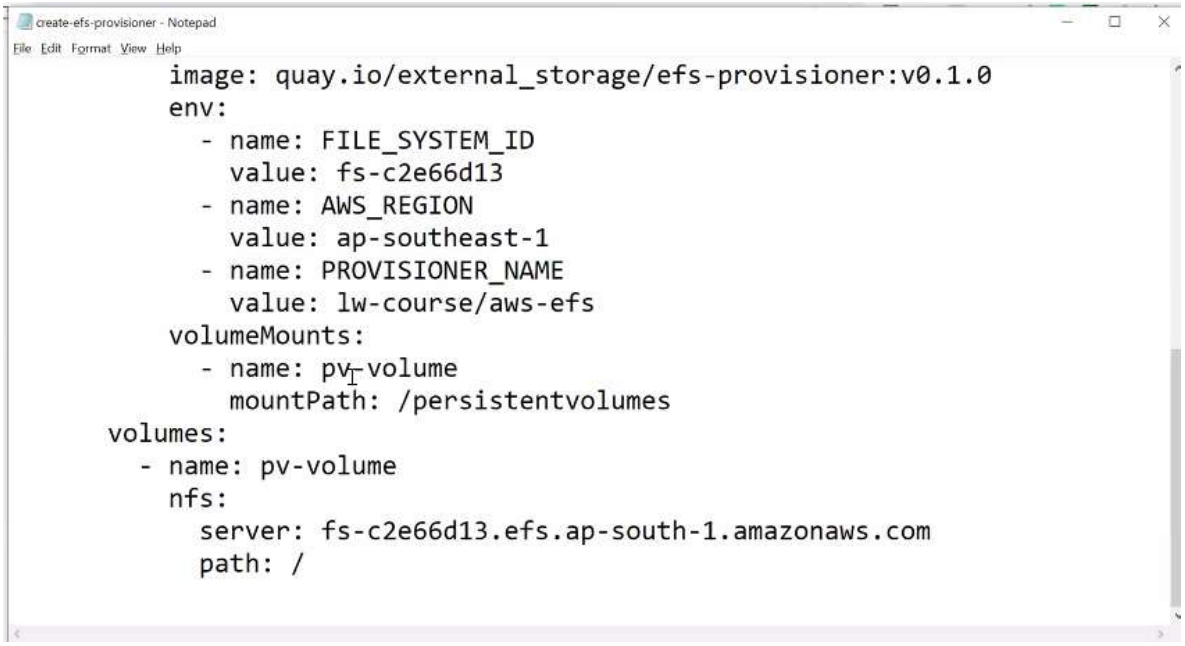
create-rbac - Notepad

File Edit Format View Help

```
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: nfs-provisioner-role-binding
subjects:
  - kind: ServiceAccount
    name: default
    namespace: lwns
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
```







And we have to make the server URL and the value of the File system ID

To add the EFS as the provisioner
And then we must also have the role created to have the access

```
C:\Users\Vimal Daga\Desktop\eks_class_code\Wordpress-on-EFS>notepad create-efs-provisioner.yaml

C:\Users\Vimal Daga\Desktop\eks_class_code\Wordpress-on-EFS>notepad create-rbac.yaml

C:\Users\Vimal Daga\Desktop\eks_class_code\Wordpress-on-EFS>notepad create-storage.yaml

C:\Users\Vimal Daga\Desktop\eks_class_code\Wordpress-on-EFS>notepad create-efs-provisioner.yaml

C:\Users\Vimal Daga\Desktop\eks_class_code\Wordpress-on-EFS>kubectl create ns lwns
namespace/lwns created

C:\Users\Vimal Daga\Desktop\eks_class_code\Wordpress-on-EFS>
C:\Users\Vimal Daga\Desktop\eks_class_code\Wordpress-on-EFS>kubectl create -f create-efs-provisioner.yaml  -n
 lwns
deployment.apps/efs-provisioner created

C:\Users\Vimal Daga\Desktop\eks_class_code\Wordpress-on-EFS>kubectl create -f create-rbac.yaml -n lwns
clusterrolebinding.rbac.authorization.k8s.io/nfs-provisioner-role-binding created

C:\Users\Vimal Daga\Desktop\eks_class_code\Wordpress-on-EFS>kubectl create -f  create-storage.yaml -n
```

And to make the EFS to be used for this the nodes must have the NFS support for them

And for this we have to just login in to the nodes for the first time and them install the nfs-utils to make the node allow EFS (this is working on the NFS protocol)

```
yum  install amazon-efs-utils
```

And here we the role of any automation script we can use the Ansible like tool to do this for us
And if we are using the MySQL then we must also have the secret which we have to pass the passwords that are needed

```
C:\Users\Vimal Daga\Desktop\eks_class_code\Wordpress-on-EFS>kubectl create secret generic mysql-pass  --from-li
teral=password=redhat
secret/mysql-pass created
```

And in the fargate if we see in the Singapore from the Web UI they show that none are there

```
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 ns
NAME              STATUS   AGE
default           Active   103m
kube-node-lease   Active   103m
kube-public       Active   103m
kube-system       Active   103m

C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get nodes
NAME                                                STATUS   ROLES    AGE   VERSION
fargate-ip-192-168-162-87.ap-southeast-1.compute.internal    Ready    <none>   56m   v1.16.8-eks-e16311
fargate-ip-192-168-187-236.ap-southeast-1.compute.internal   Ready    <none>   56m   v1.16.8-eks-e16311
```

```
C:\Users\Vimal Daga\Desktop\eks_class_code>kubectl get pods  --all-namespaces
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE
kube-system   coredns-cb56d5db9-r5fr7    1/1     Running   0          57m
kube-system   coredns-cb56d5db9-wcxdl    1/1     Running   0          57m
```

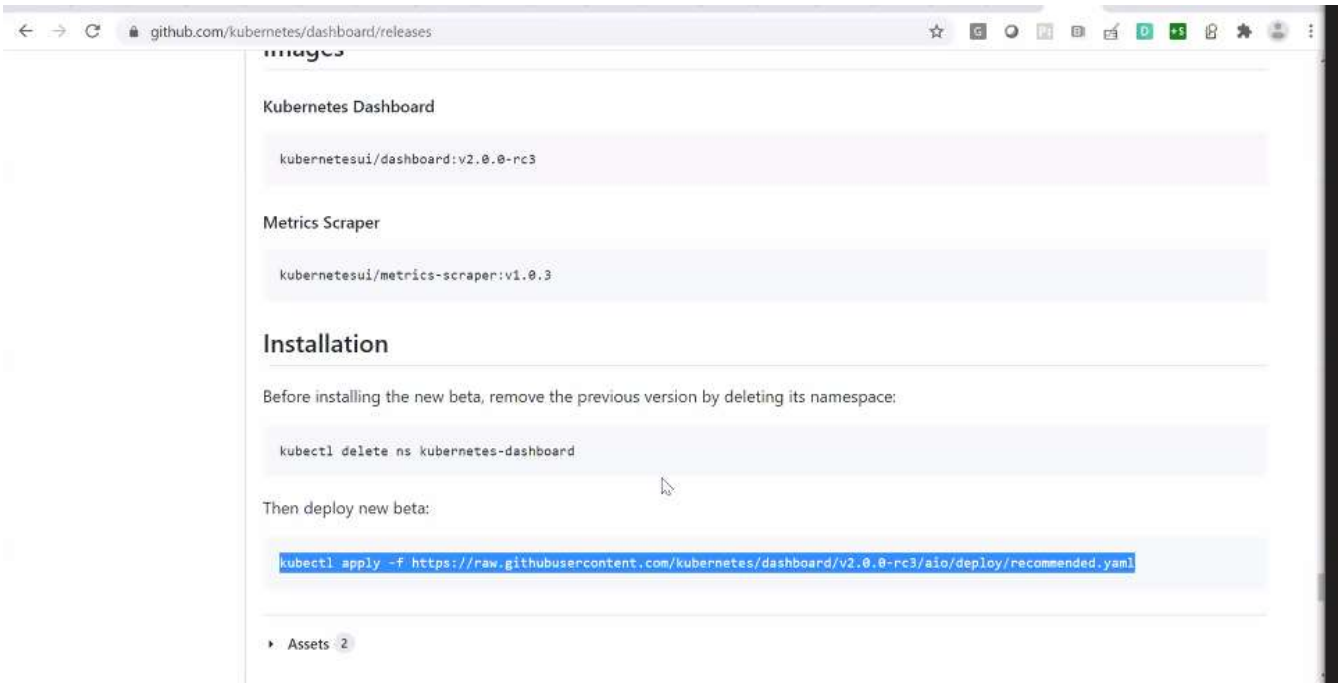
But internally we see they are some pods that are already working
And it is like On the fly and when we use they will create everything for us dynamically
And that is why it is called server less

And in the fargate we see that they are using some another container technology

```
Container Runtime Version:  containerd://1.3.2
```

And as the demand comes they will also scale for us
And this is called as server less as we don't to manage them but there is a server on which everything is running

But the server perspective we can't see the anything about the Master and Slave



And this command can be used to create the dashboard for the K8s Cluster we have
And this will run the file and will do all the things what will do us

And to connect we can use the Kube Proxy to connect to it
And for this the commands are provided in the Document

```
kubectl create serviceaccount dashboard-admin-sa
kubectl create clusterrolebinding dashboard-admin-sa
--clusterrole=cluster-admin --serviceaccount=default:dashboard-admin-sa

kubectl get secrets

kubectl describe secret dashboard-admin-sa-token-kw7vn
```

==========

The main motive of the program is that

- It is not about the AWS or K8s or Docker
- It is about the EKS
- And it is also about how the K8s is deployed in the real world
- And what all the integrations we have in the EKS and internally how it works but it will launch the required for us in just one click