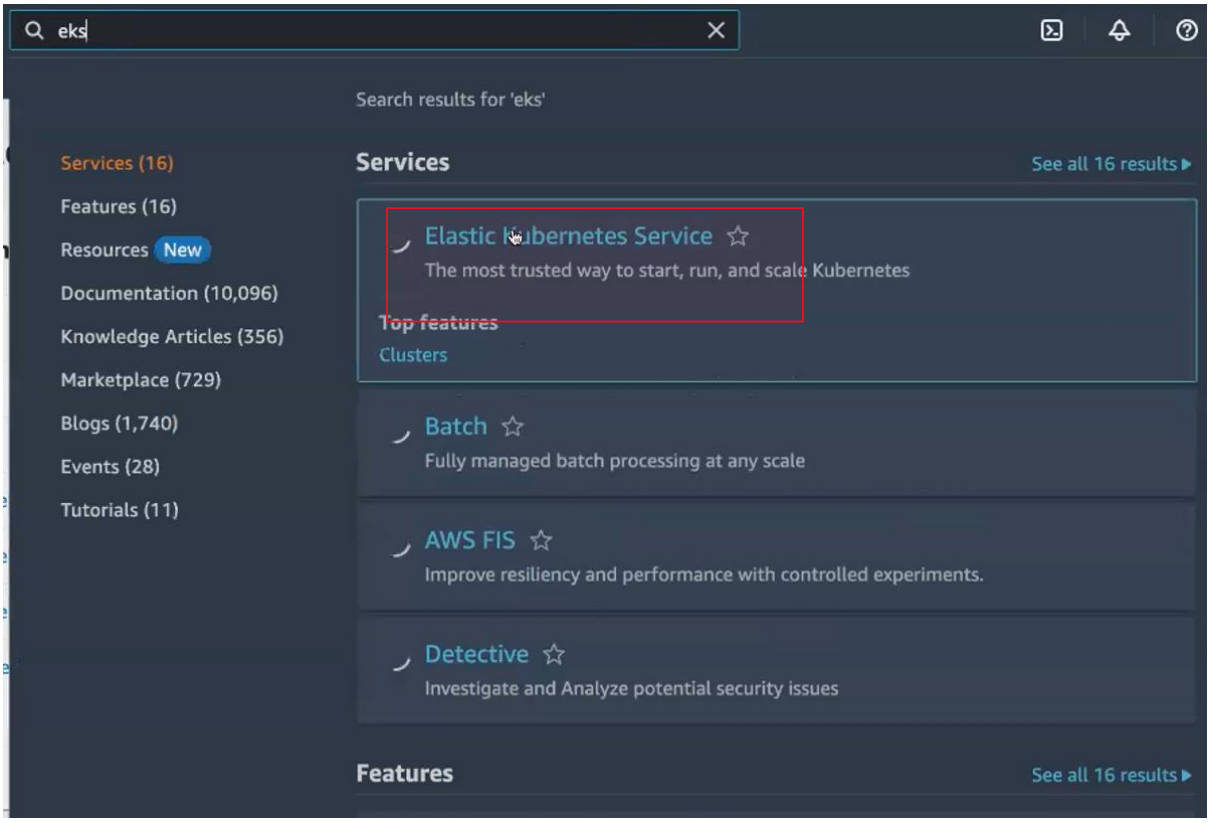
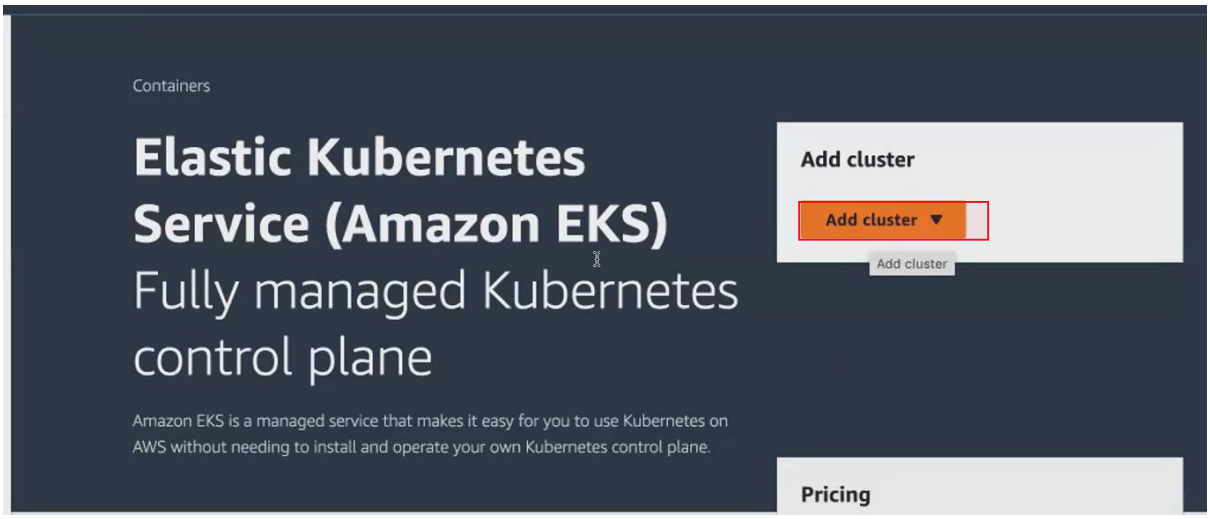
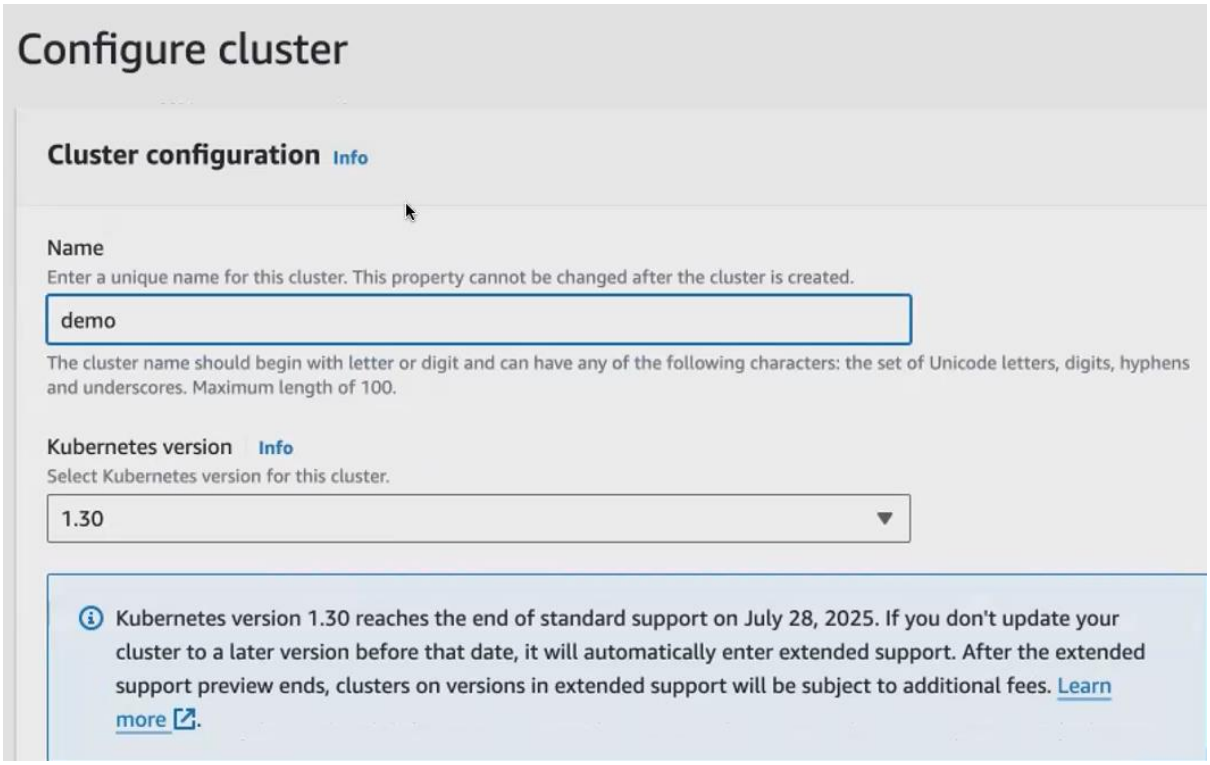
Day 1 - Kubernetes Introduction

- It is container orchestration tool. So, container orchestration mean, which is used to manage or to deal with your containers. In docker, we created containers.
- We use Docker to launch the container. Okay, you create a docker image out of an application. After creating the docker image in order to access your application. We containerized it. Okay? So we worked on one container. What in real time scenarios, you have multiple containers. You will be dealing with a lot of containers. So in order to manage those container.
- we go with a tool called Kubernetes
- This was developed by Google just to manage your containers. Okay without containers, You cannot work with Kubernetes. So Kubernetes require containerization technology to work on. So here we use Kubernetes to manage the container to monitor the container to take complete control of your containers Lifecycle, we will be using Kubernetes.
- The Kubernetes follows a master slave architecture. You have a master Node, and you have a slave node. We create cluster with Kubernetes. Cluster means you have multiple machine connected with one another.
- We will be creating cluster in eks. or if you're using azure, we have Aks, Kubernetes cluster a service provided by different cloud providers, or you can also create your cluster on premise.
- So Eks will help you to create the cluster, launch. The Kubernetes cluster in the Aws environment. You can go the manual way. So you create you click on, create cluster.

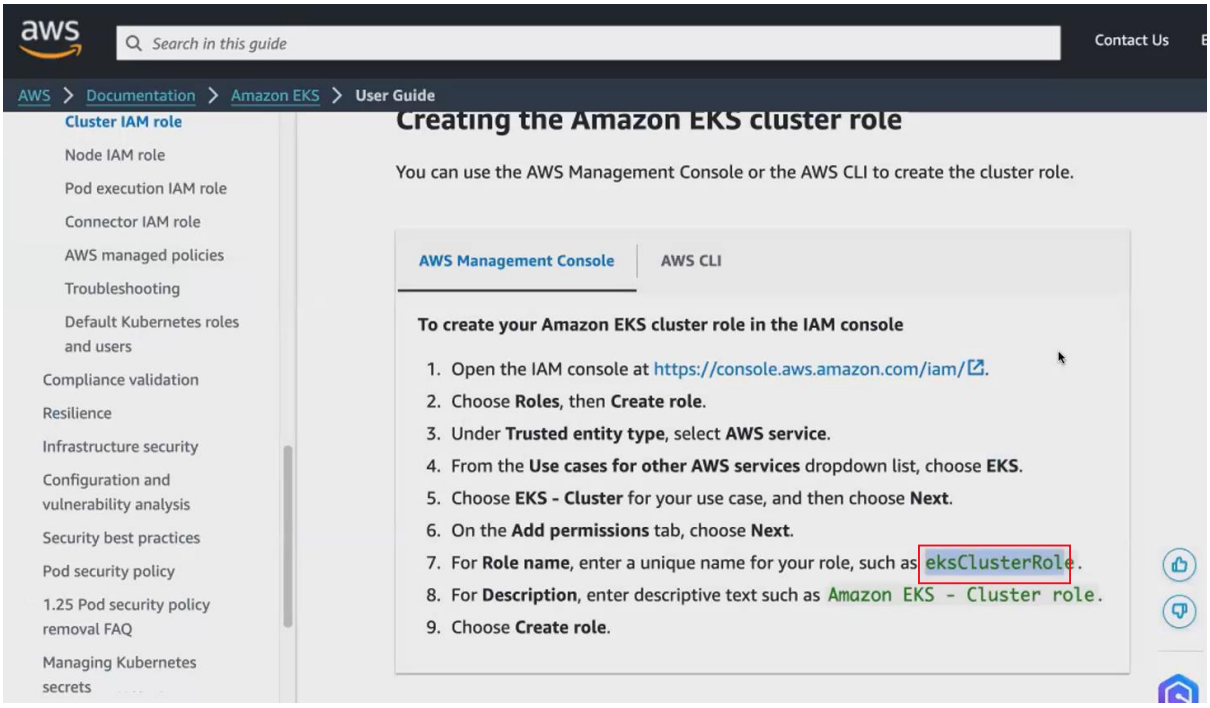


Its not free ..all are chargeble





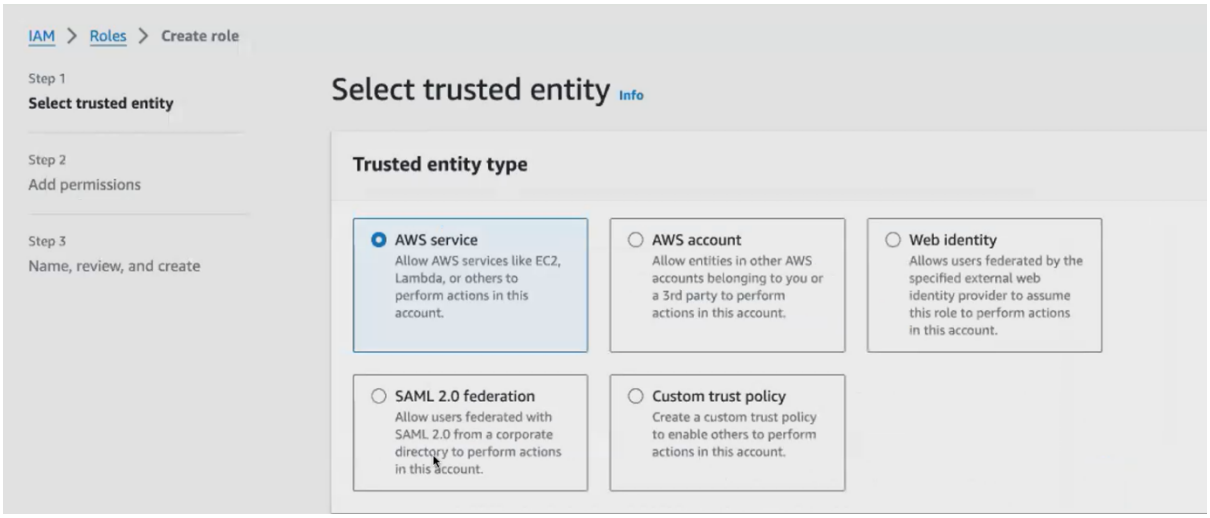
While add cluster , ask for **eks cluster role** .

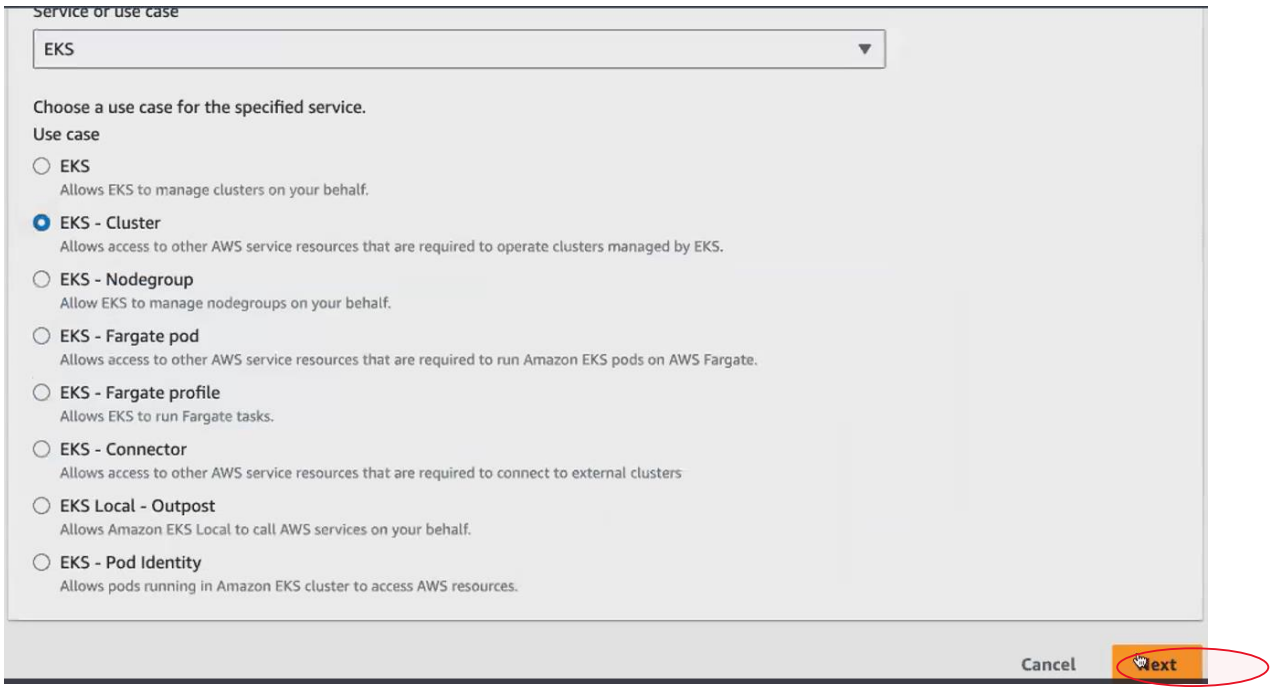


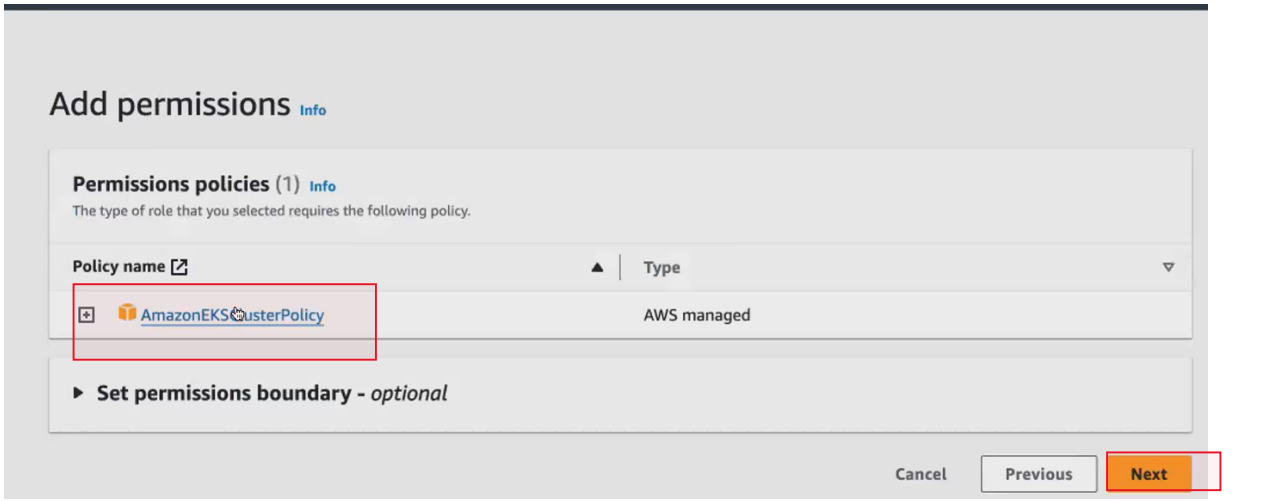
Scroll down

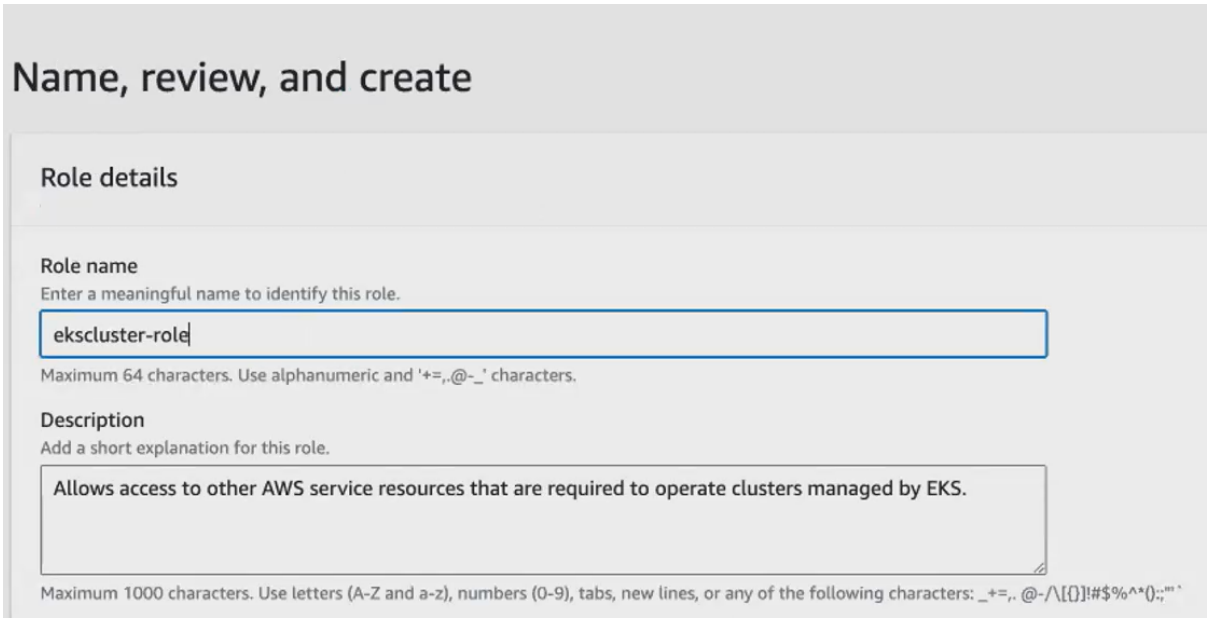
To check for the `eksClusterRole` in the IAM console

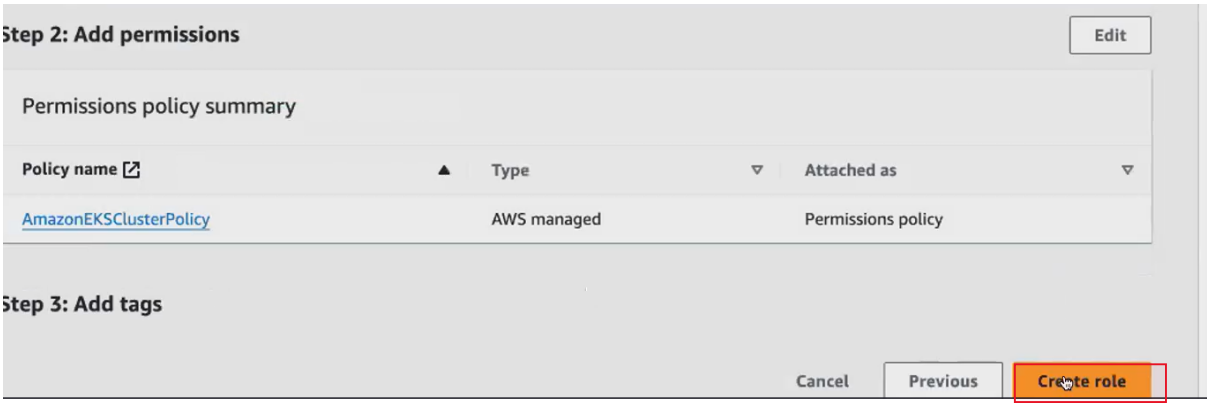
1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**.
3. Search the list of roles for `eksClusterRole`. If a role that includes `eksClusterRole` doesn't exist, then see Creating the Amazon EKS cluster role to create the role. If a role that includes `eksClusterRole` does exist, then select the role to view the attached policies.
4. Choose **Permissions**.
5. Ensure that the **AmazonEKSClusterPolicy** managed policy is attached to the role. If the policy is attached, your Amazon EKS cluster role is properly configured.
6. Choose **Trust relationships**, and then choose **Edit trust policy**.
7. Verify that the trust relationship contains the following policy. If the trust relationship matches the following policy, choose **Cancel**. If the trust relationship doesn't match, copy the policy into the **Edit trust policy** window and choose **Update policy**.

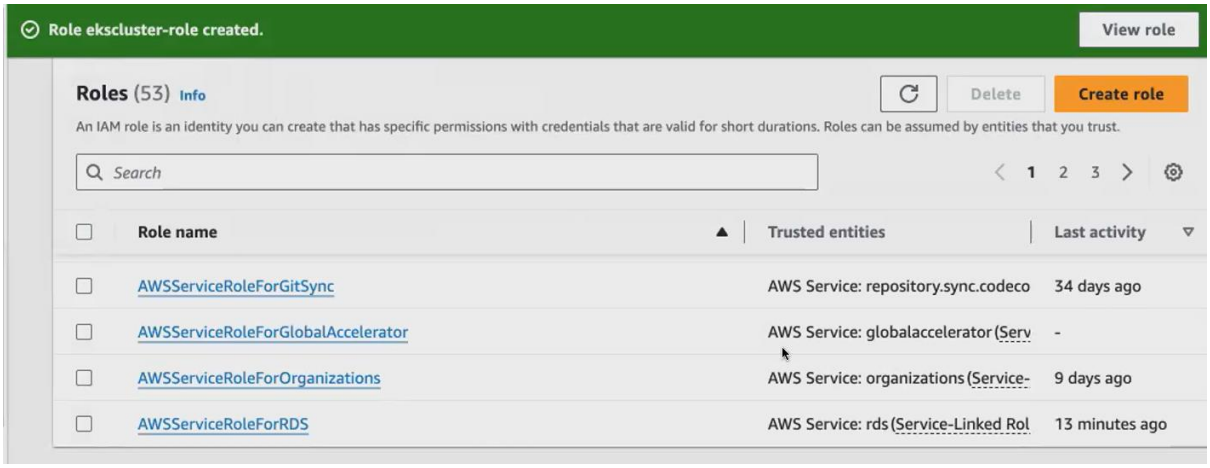


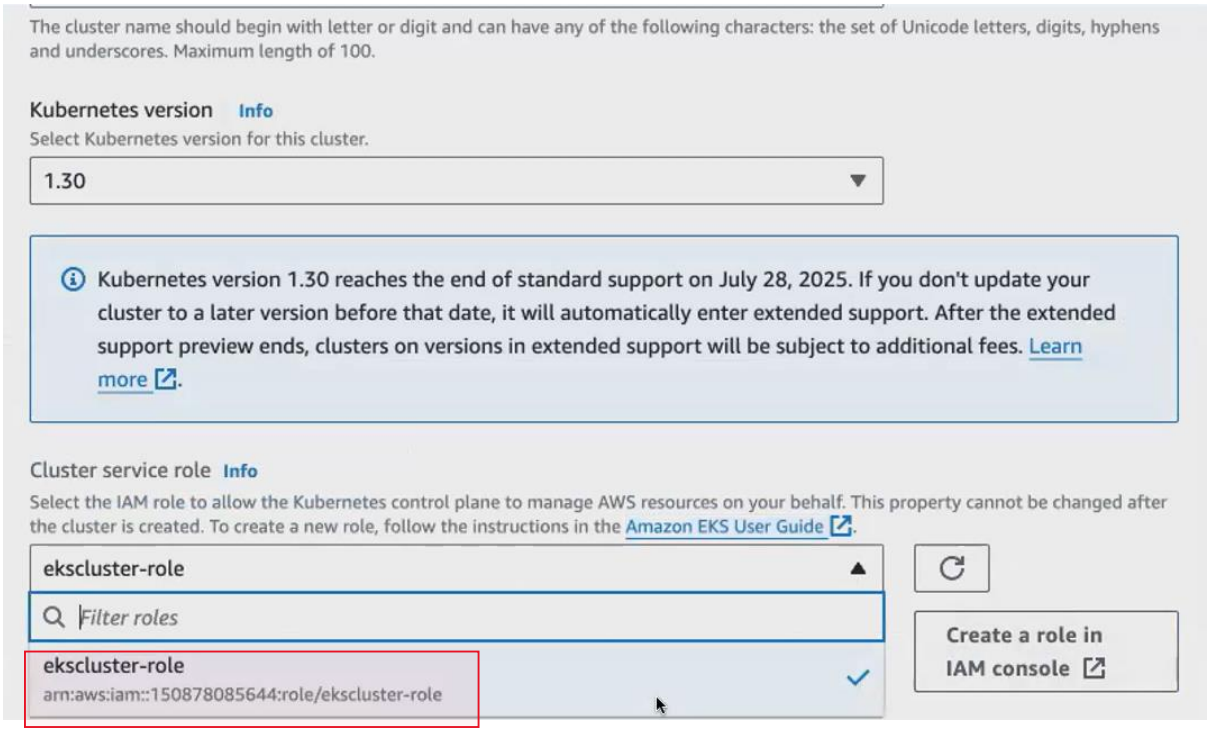

Service or use case
EKS
Choose a use case for the specified service.
Use case
EKS
Allows EKS to manage clusters on your behalf.
EKS - Cluster
Allows access to other AWS service resources that are required to operate clusters managed by EKS.
EKS - Nodegroup
Allow EKS to manage nodegroups on your behalf.
EKS - Fargate pod
Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate.
EKS - Fargate profile
Allows EKS to run Fargate tasks.
EKS - Connector
Allows access to other AWS service resources that are required to connect to external clusters
EKS Local - Outpost
Allows Amazon EKS Local to call AWS services on your behalf.
EKS - Pod Identity
Allows pods running in Amazon EKS cluster to access AWS resources.
Cancel
Next


Add permissions Info
Permissions policies (1) Info
The type of role that you selected requires the following policy.
Policy name
Type
AmazonEKSClusterPolicy
AWS managed
Set permissions boundary - optional
Cancel
Previous
Next

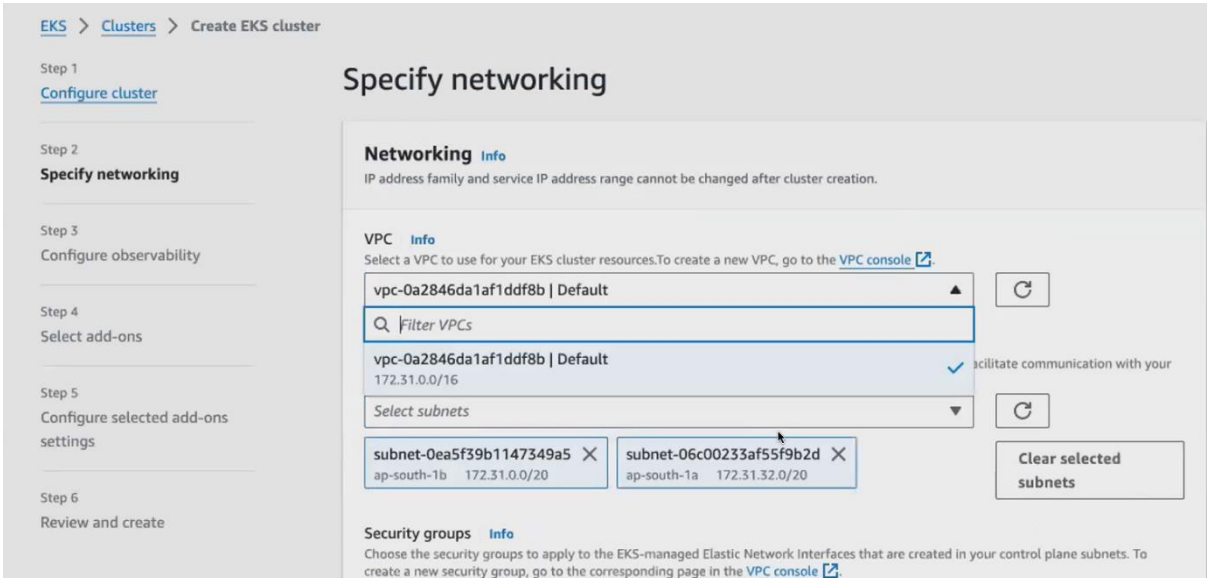

Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
ekscluster-role
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this role.
Allows access to other AWS service resources that are required to operate clusters managed by EKS.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/\[{}]!#$%^*();:"' `


Step 2: Add permissions
Edit
Permissions policy summary
Policy name
Type
Attached as
AmazonEKSClusterPolicy
AWS managed
Permissions policy
Step 3: Add tags
Cancel
Previous
Create role

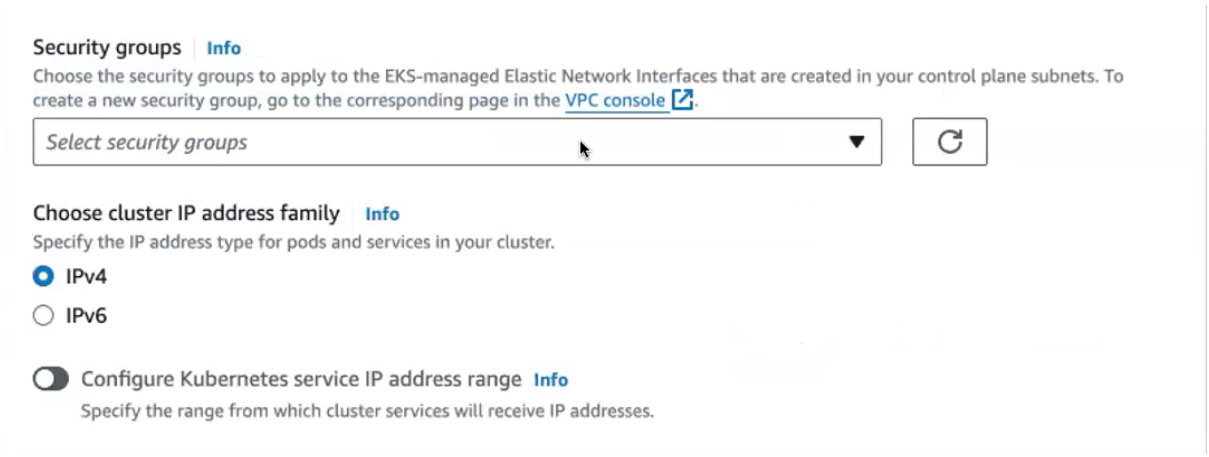

Role ekscluster-role created.
View role
Roles (53) Info
Delete
Create role
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.
Search
1 2 3
Role name
Trusted entities
Last activity
AWSServiceRoleForGitSync
AWS Service: repository.sync.codeco
34 days ago
AWSServiceRoleForGlobalAccelerator
AWS Service: globalaccelerator (Serv
-
AWSServiceRoleForOrganizations
AWS Service: organizations (Service-
9 days ago
AWSServiceRoleForRDS
AWS Service: rds (Service-Linked Rol
13 minutes ago

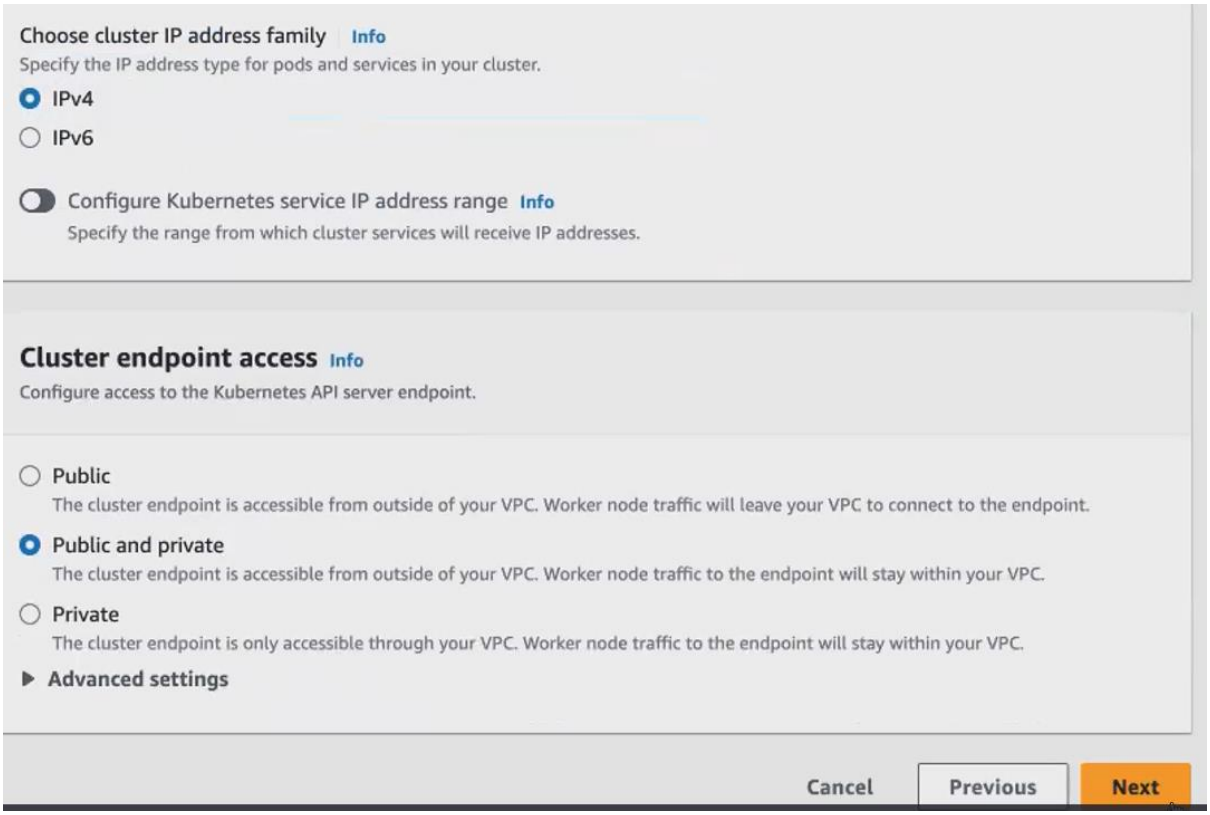


Remaining are all same in configure cluster. Click next

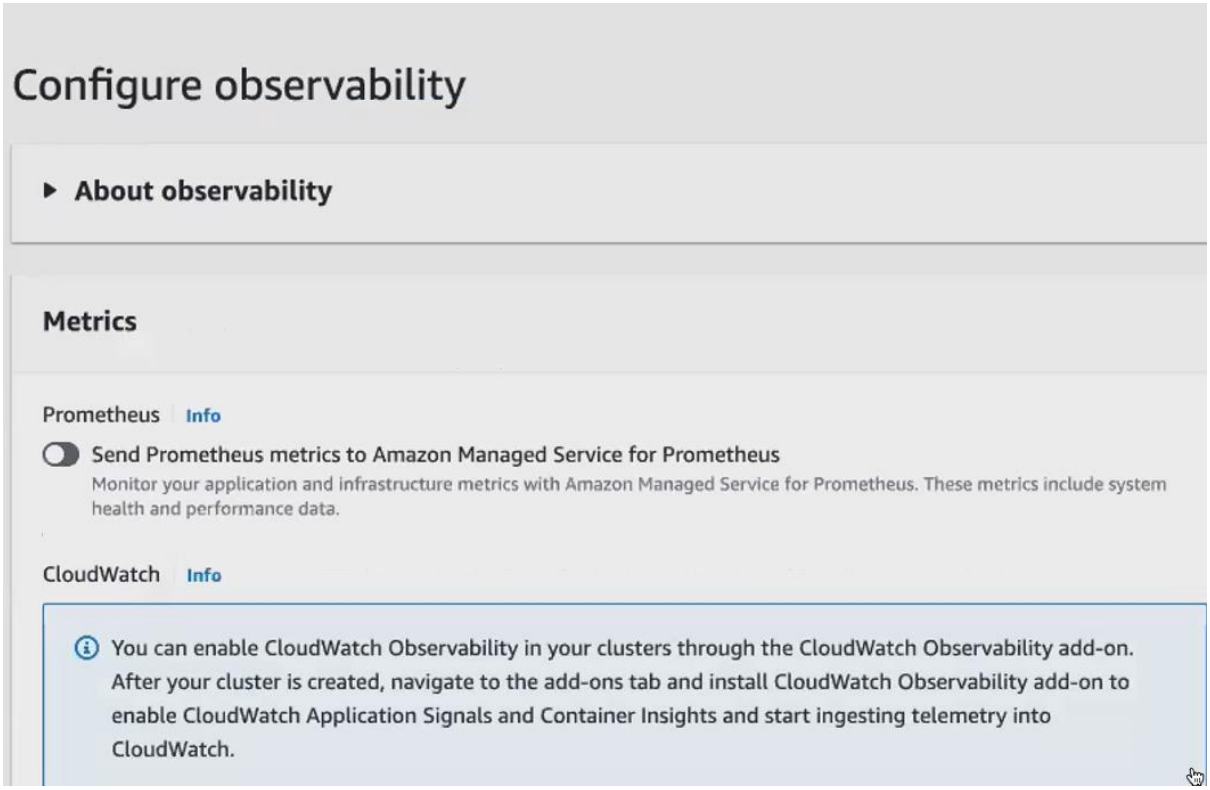


Security group is auto selected

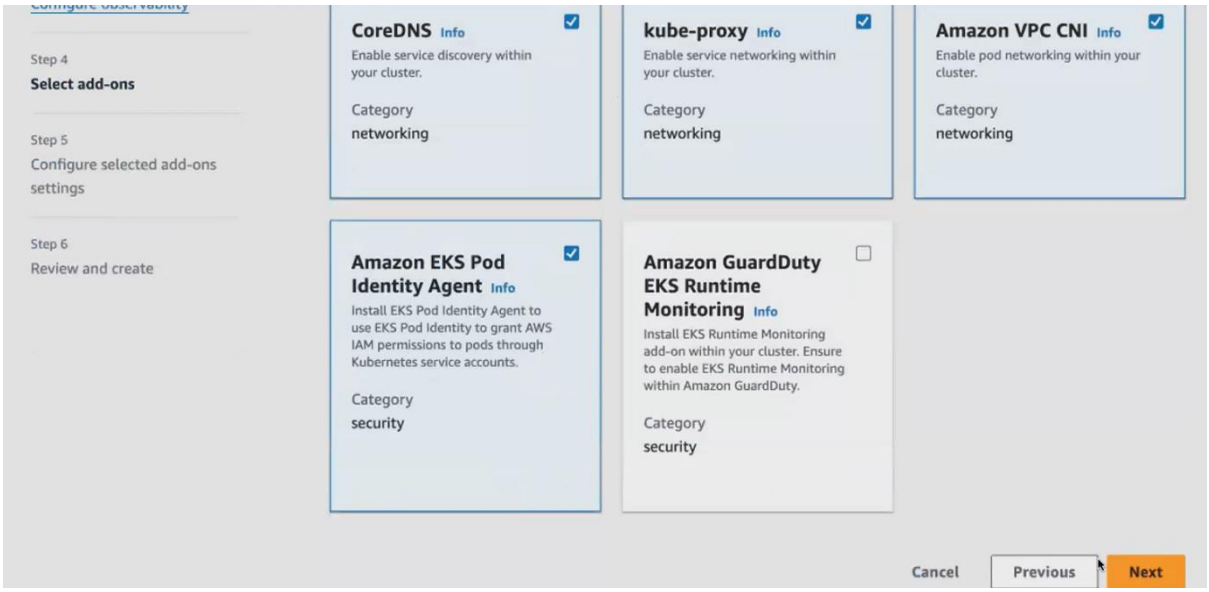




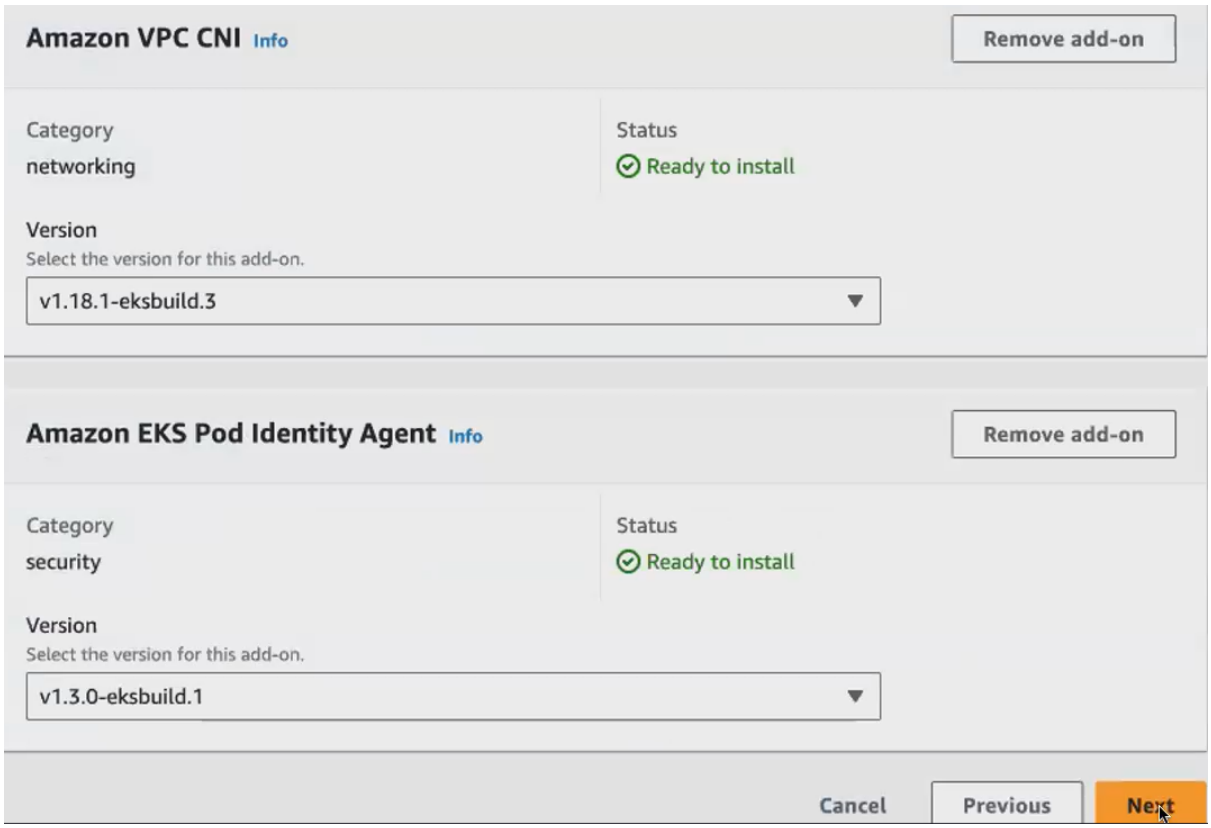
I don't want to enable any observability, your metrics, your cluster details, your application that is running inside your cluster. Everything will be monitored.

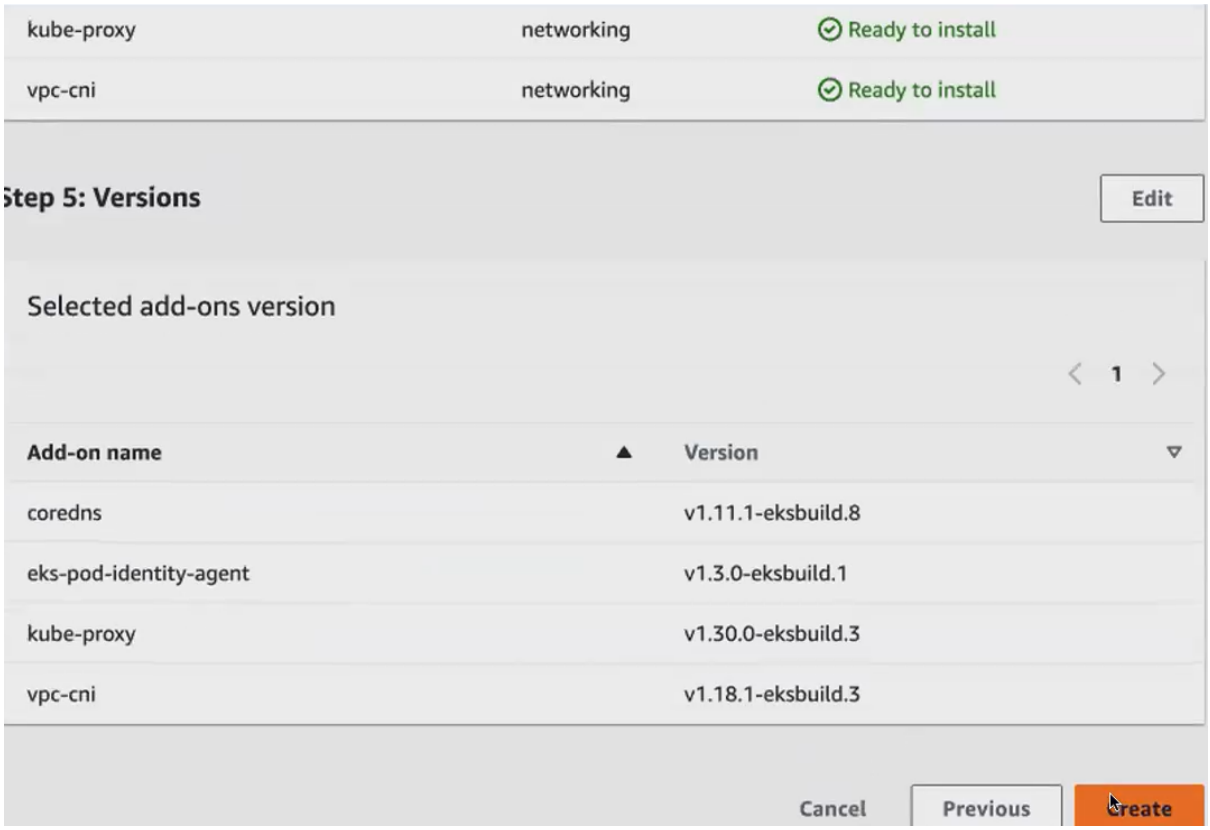


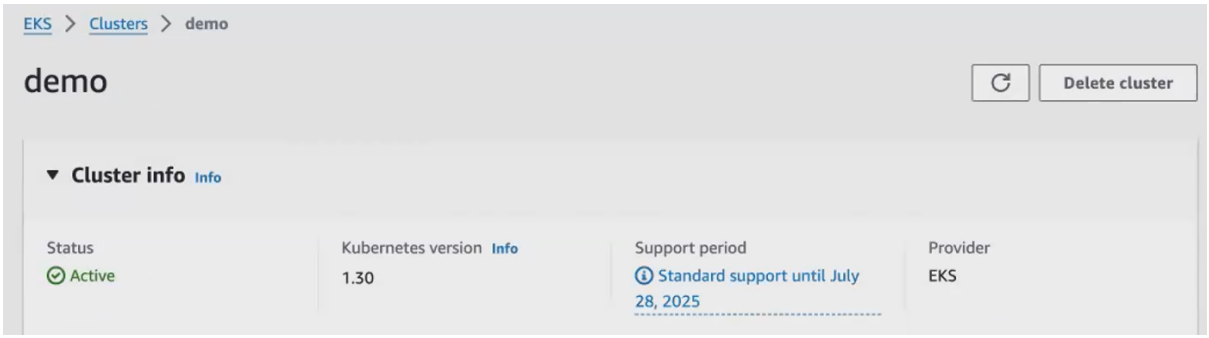
All are default add -ons .must be selected all



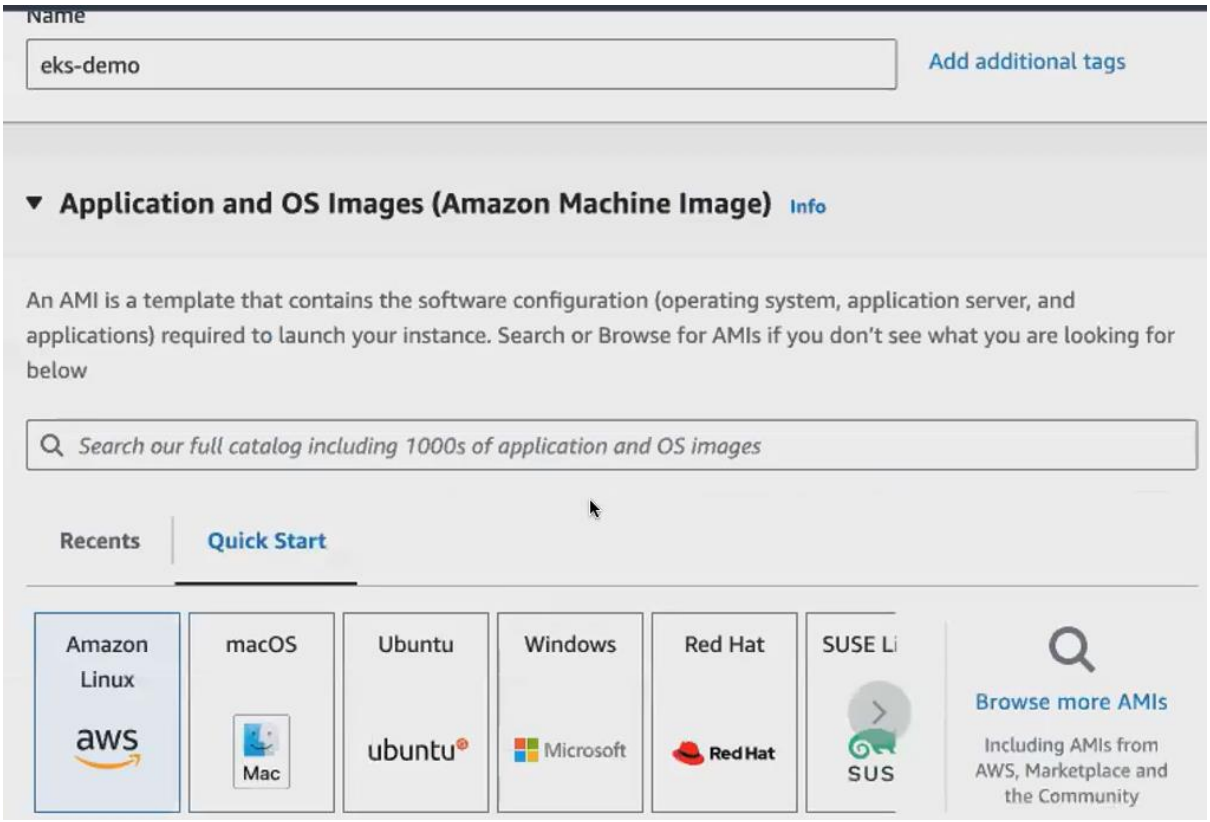
Already all selected

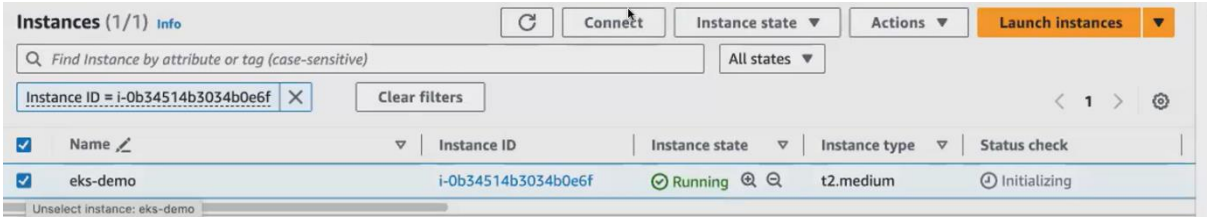






Launch an ec2 for cluster. Select ubuntu →t2.micro

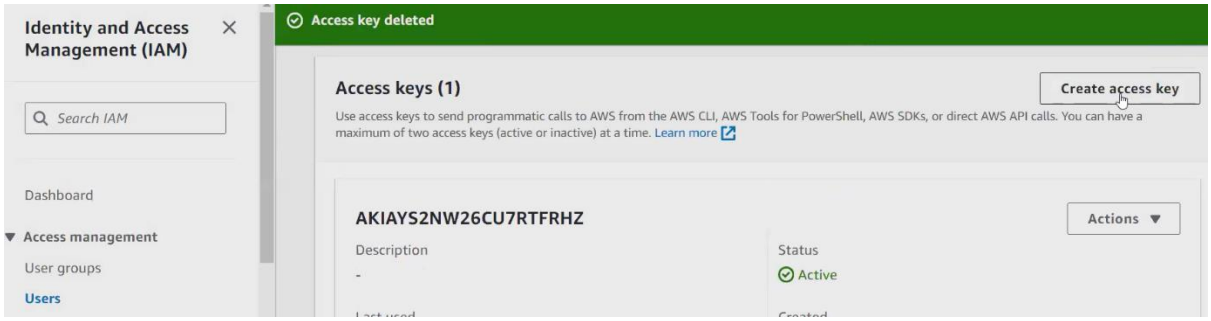


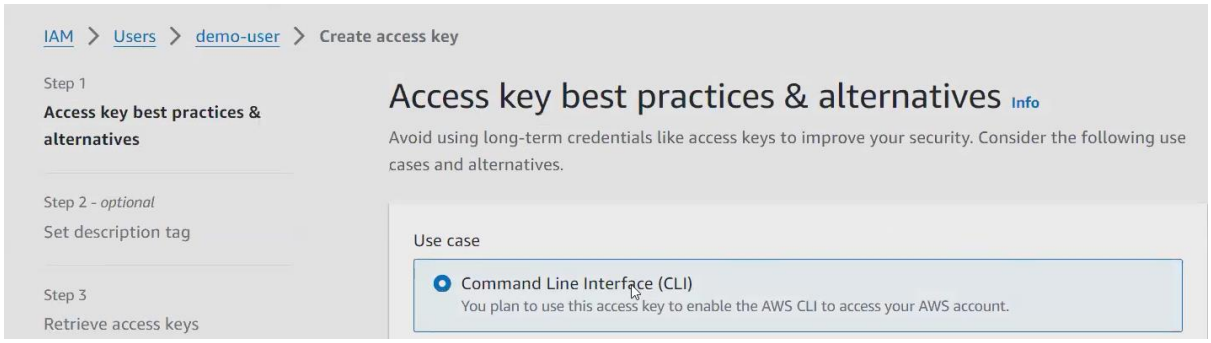


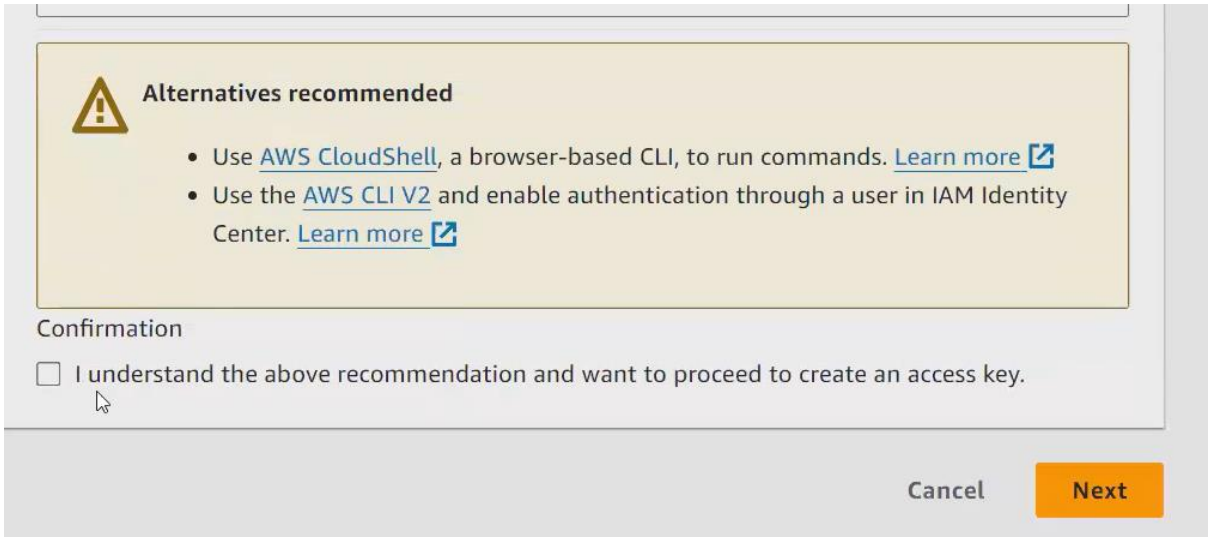
Connect an instance install awscli

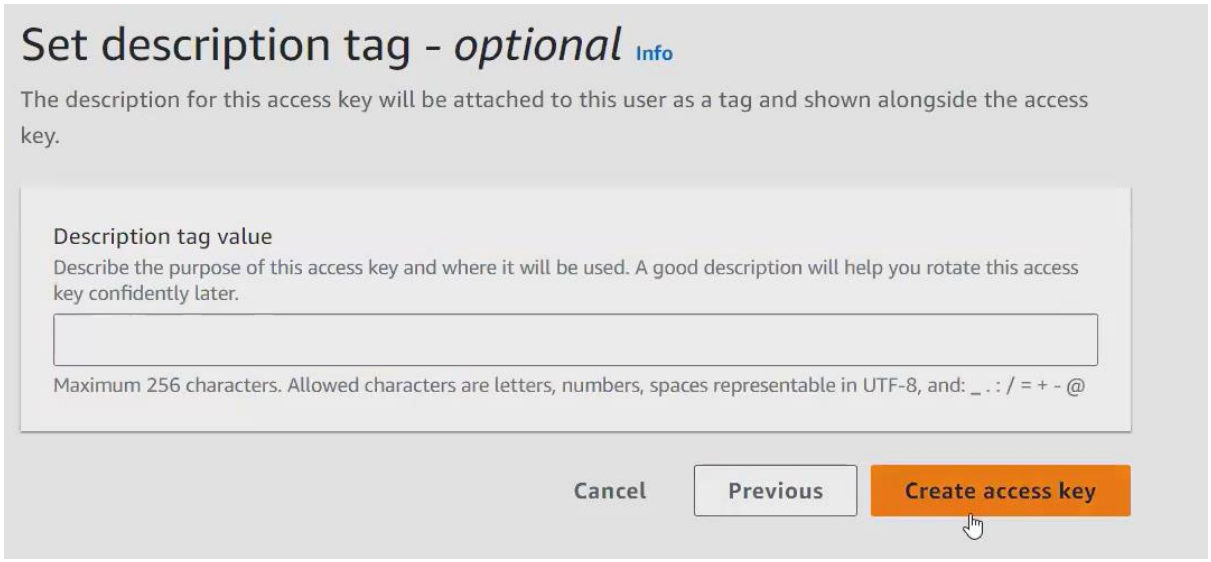
```
ubuntu@ip-172-31-38-243:~$ sudo apt update
ubuntu@ip-172-31-38-243:~$ sudo apt install awscli -y
ubuntu@ip-172-31-38-243:~$ aws configure
```

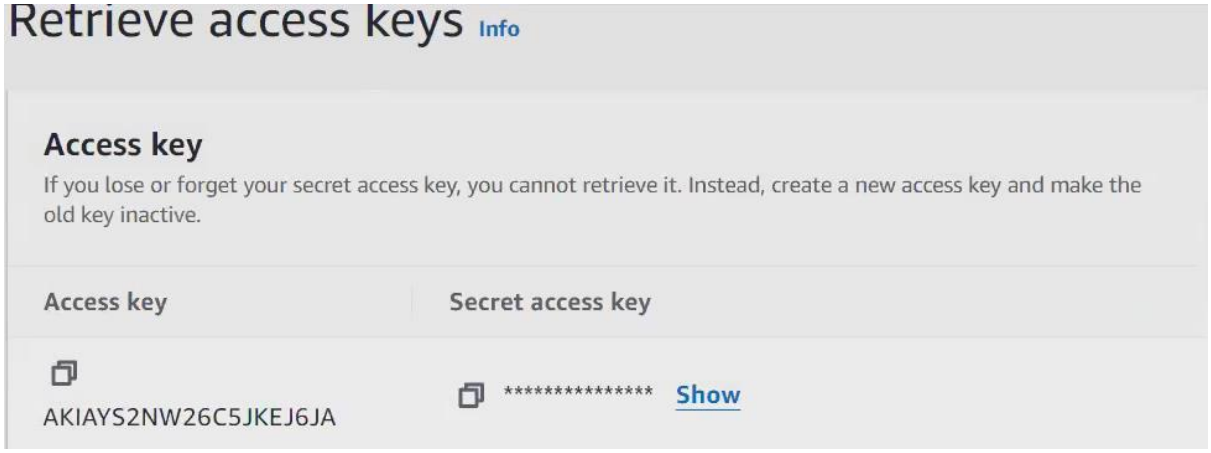
How to generate access key and secret access key:





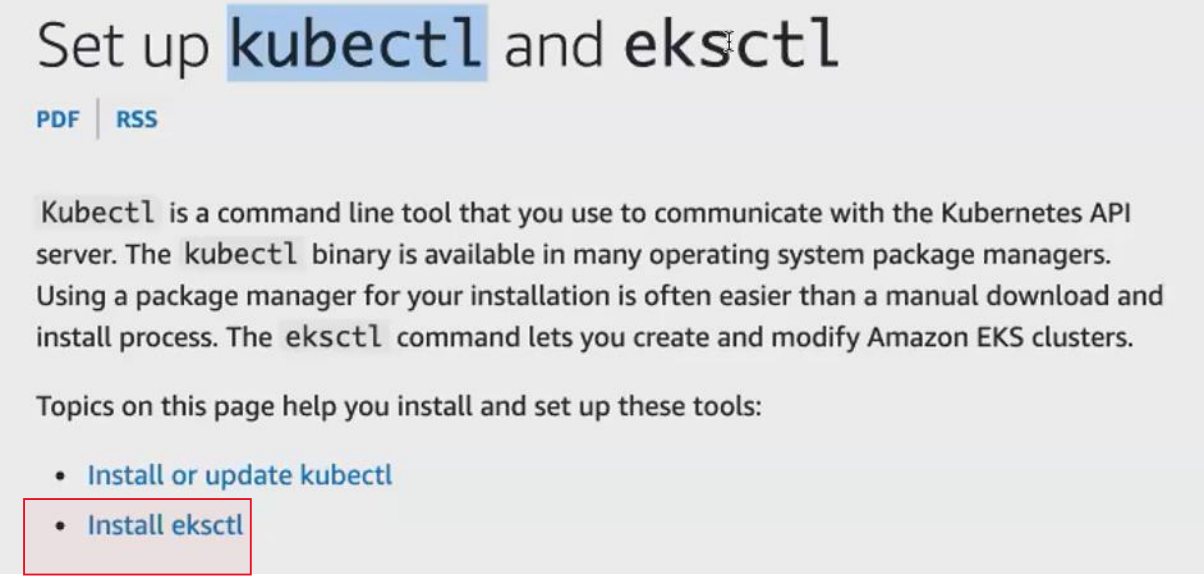


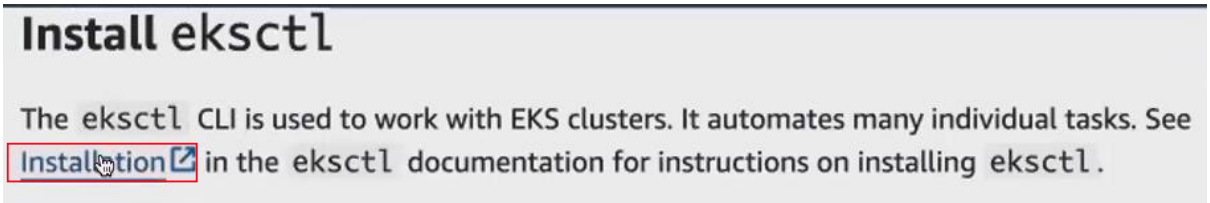

Set description tag - optional Info
The description for this access key will be attached to this user as a tag and shown alongside the access key.
Description tag value
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.
Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @
Cancel
Previous
Create access key


Retrieve access keys Info
Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.
Access key
Secret access key
AKIAYS2NW26C5JKEJ6JA
********** Show

How to install eksctl and kubectl:

To communicate with your cluster. As I already told you we use kubeCtl, your cluster, you can create it in different way. You have different methods to launch your cluster. But to communicate with your cluster. We only use a tool called kubeCtl.





https://eksctl.io/installation/ [below cmd in this doc link]

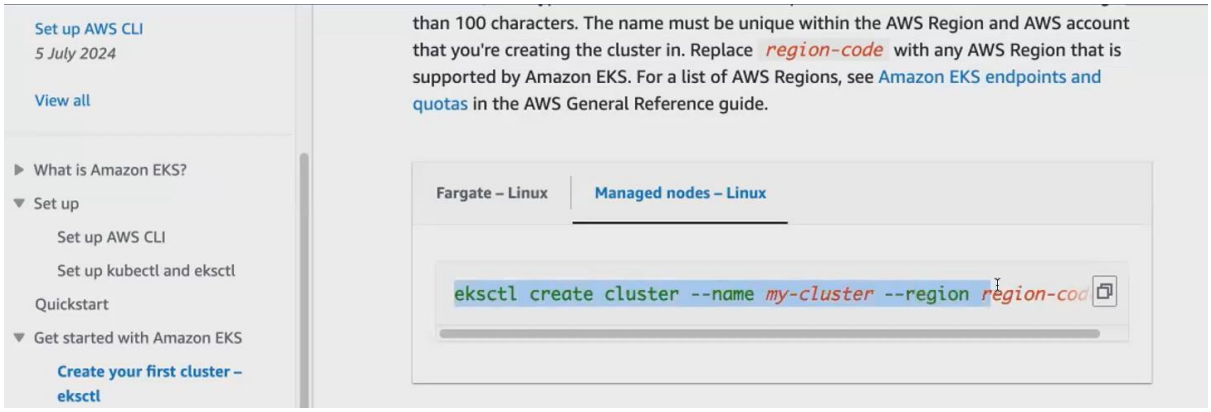
https://docs.google.com/document/d/1w4DZAbVGEZLe4pnzGAzi-d3LcNh5wZtQ/edit?pli=1#heading=h.gjdgxs [all commands in this doc]

```
ubuntu@ip-172-31-38-243:~$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz"
tar xz -C /tmp
ubuntu@ip-172-31-38-243:~$ sudo mv /tmp/eksctl /usr/local/bin
ubuntu@ip-172-31-38-243:~$ eksctl version
0.187.0
```

```
ubuntu@ip-172-31-38-243:~$ curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
 60 57.4M   60 34.5M    0     0  4600k      0  0:00:12  0:00:07  0:00:05 6342k
```

```
ubuntu@ip-172-31-38-243:~$ chmod +x ./kubectl
ubuntu@ip-172-31-38-243:~$ sudo mv ./kubectl /usr/local/bin
ubuntu@ip-172-31-38-243:~$ kubectl version --short --client
Client Version: v1.19.6-eks-49a6c0
ubuntu@ip-172-31-38-243:~$
```

Create a cluster by using eksctl:




```
ubuntu@ip-172-31-38-243:~$ eksctl create cluster --name demo-guvi --region ap-south-1 --node-type t2.micro --nodes-min 2 --nodes-max 10
```

eksctl delete cluster --name demo-guvi --region ap-south-1

1. Name of cluster
2. Region
3. Node type
4. Min no of nodes -2 [always maintain 2]
5. Max no of nodes -10

https://docs.google.com/document/d/1w4DZAbVGEZLe4pnzGAzi-d3LcNh5wZtQ/edit?pli=1#heading=h.gjdgxs [above all commands in this doc]

cluster created

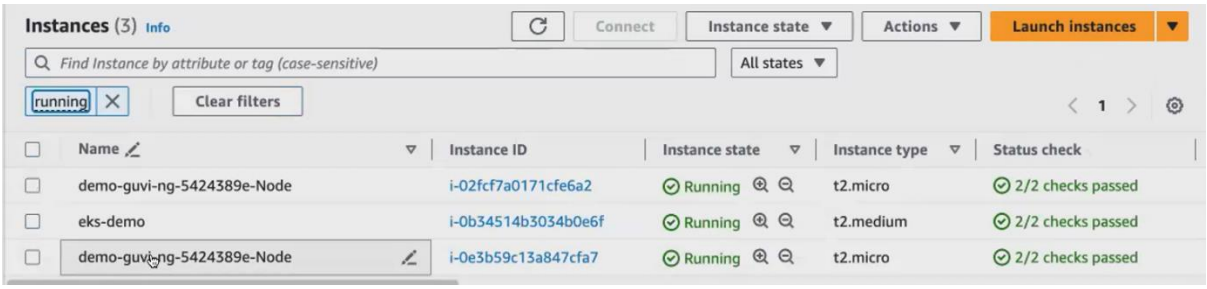
```
2024-07-18 12:00:43 [ℹ]  successfully created addon
2024-07-18 12:00:43 [ℹ]  creating addon
2024-07-18 12:00:44 [ℹ]  successfully created addon
2024-07-18 12:02:44 [ℹ]  building managed nodegroup stack "eksctl-demo-guvi-nodegroup-ng-5424389e"
2024-07-18 12:02:44 [ℹ]  deploying stack "eksctl-demo-guvi-nodegroup-ng-5424389e"
2024-07-18 12:02:44 [ℹ]  waiting for CloudFormation stack "eksctl-demo-guvi-nodegroup-ng-5424389e"
2024-07-18 12:03:15 [ℹ]  waiting for CloudFormation stack "eksctl-demo-guvi-nodegroup-ng-5424389e"
2024-07-18 12:03:47 [ℹ]  waiting for CloudFormation stack "eksctl-demo-guvi-nodegroup-ng-5424389e"
2024-07-18 12:04:56 [ℹ]  waiting for CloudFormation stack "eksctl-demo-guvi-nodegroup-ng-5424389e"
2024-07-18 12:06:38 [ℹ]  waiting for CloudFormation stack "eksctl-demo-guvi-nodegroup-ng-5424389e"
2024-07-18 12:06:38 [ℹ]  waiting for the control plane to become ready
2024-07-18 12:06:38 [✔]  saved kubeconfig as "/home/ubuntu/.kube/config"
2024-07-18 12:06:38 [ℹ]  no tasks
2024-07-18 12:06:38 [✔]  all EKS cluster resources for "demo-guvi" have been created
2024-07-18 12:06:38 [✔]  created 0 nodegroup(s) in cluster "demo-guvi"
2024-07-18 12:06:38 [ℹ]  nodegroup "ng-5424389e" has 2 node(s)
2024-07-18 12:06:38 [ℹ]  node "ip-192-168-22-40.ap-south-1.compute.internal" is ready
2024-07-18 12:06:38 [ℹ]  node "ip-192-168-42-4.ap-south-1.compute.internal" is ready
2024-07-18 12:06:38 [ℹ]  waiting for at least 2 node(s) to become ready in "ng-5424389e"
2024-07-18 12:06:38 [ℹ]  nodegroup "ng-5424389e" has 2 node(s)
2024-07-18 12:06:38 [ℹ]  node "ip-192-168-22-40.ap-south-1.compute.internal" is ready
2024-07-18 12:06:38 [ℹ]  node "ip-192-168-42-4.ap-south-1.compute.internal" is ready
2024-07-18 12:06:38 [✔]  created 1 managed nodegroup(s) in cluster "demo-guvi"
2024-07-18 12:06:39 [ℹ]  kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2024-07-18 12:06:39 [✔]  EKS cluster "demo-guvi" in "ap-south-1" region is ready
```

2 nodes created

```
ubuntu@ip-172-31-38-243:~$ kubectl get nodes
NAME                                            STATUS   ROLES    AGE     VERSION
ip-192-168-22-40.ap-south-1.compute.internal    Ready    <none>   3m52s   v1.30.0-eks-036c24b
ip-192-168-42-4.ap-south-1.compute.internal     Ready    <none>   3m55s   v1.30.0-eks-036c24b
ubuntu@ip-172-31-38-243:~$
ubuntu@ip-172-31-38-243:~$ eksctl create cluster --name demo-guvi --region ap-south-1 --node-type t2.micro --nodes-min 2 --nodes-max 10
```

- Ng is nothing but Node Group. Node group means so your cluster, what it will do to create a new group inside the Node group. It will play all your worker nodes. Okay, you have a node group.
- **worker Node one and Worker Node 2 worker nodes got created. It's nothing but your Ec. 2 only it acts as a slave machine of your Kubernetes. It is cluster.**
- **In the Kubernetes**

    1. **master Node [taken care by aws]**
    2. **worker node [we have to take care]**

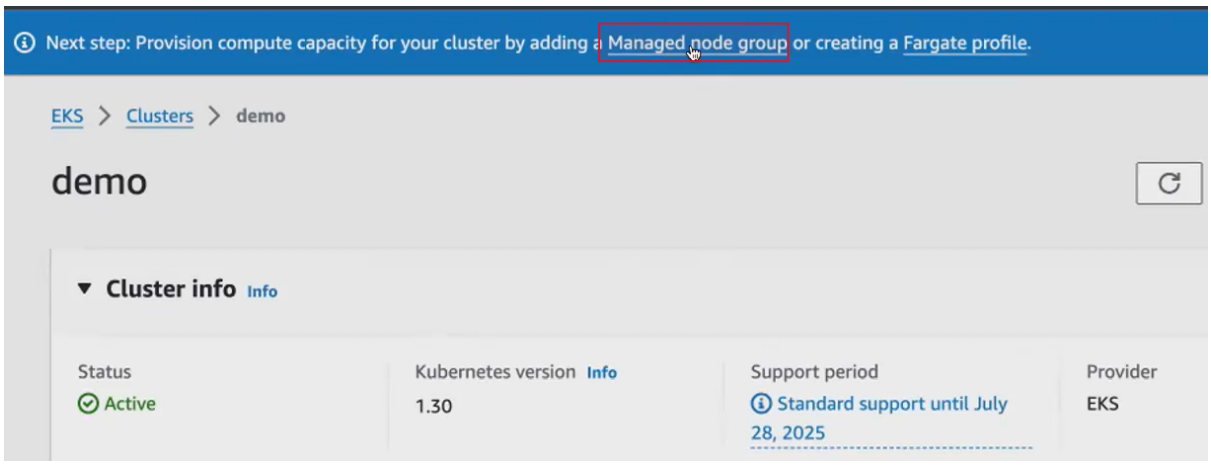


- coming to your Kubernetes cluster, we have an object called Pod, so we will be creating pod within your pod.
- We will be hosting **containers. We will be hosting our application containers, so you cannot directly launch your containers inside the Kubernetes cluster. Instead, you will create a** component called pod within your pod, we will be running the containers.
- So, if you see here, you have a node, which means you have a Kubernetes cluster inside the Kubernetes cluster. Now we have 2 worker nodes, so I assume that you have 2 worker nodes within your worker Node. You create multiple pods. Every pod. You are running your Kubernetes. You're running your docker container inside the pod.
- So you create a port and then you launch your container inside the pod, a pod can have one or more container. It is not necessary that your port can only host one container your pod can have one or more containers running inside the pod. Okay, so what we do, we will be launching this pod inside the pod. You launch your containers.

create your worker node:

you have created your cluster, so only your master Node will be created. So you have to launch your worker node. You have something called this Manage node group, you click on it, and you create your worker Node.

Worker node means you can have multiple worker notes. That is your slave machine for your Kubernetes cluster, and you will be launching it. It will create. You'll for this cluster. You have a master node, and you have your worker notes so you can create multiple worker nodes and attach it to your Kubernetes cluster. This is one way of configuring your Kubernetes cluster.



Pod :

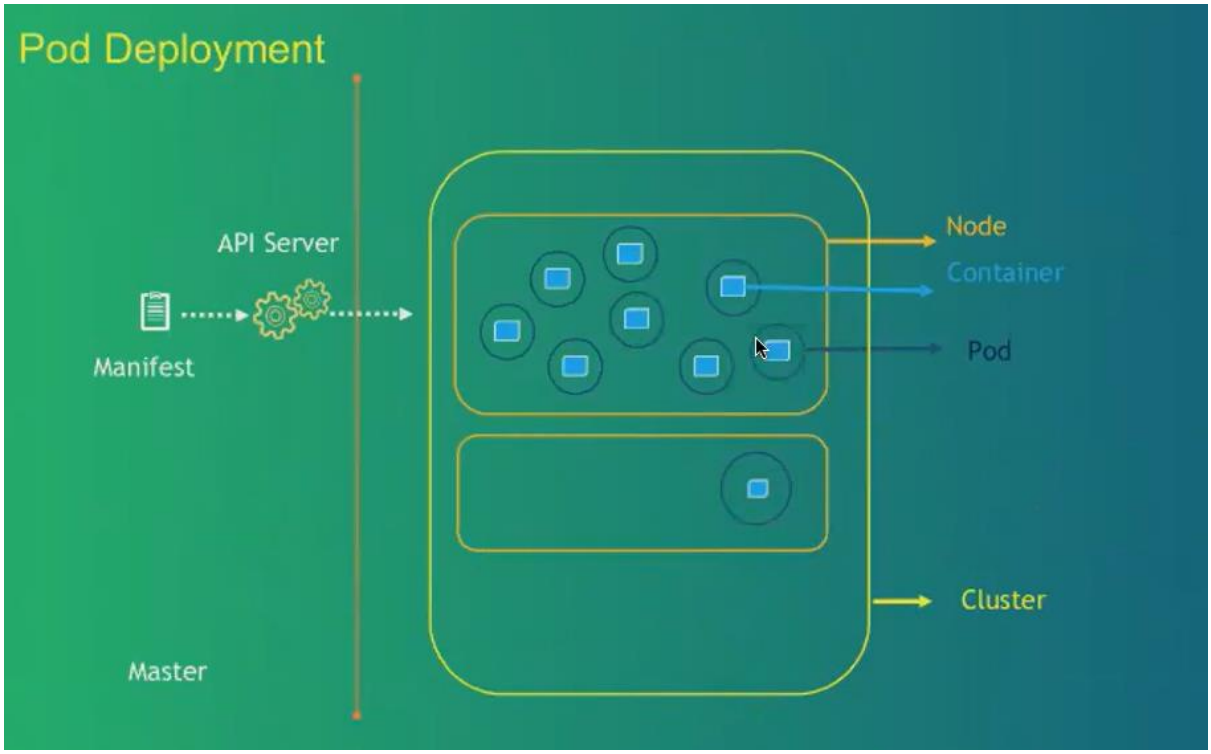
In Virtualization, basic scheduling unit is VM

In Containerization, basic scheduling unit is container

In Kubernetes, what is basic scheduling unit ?

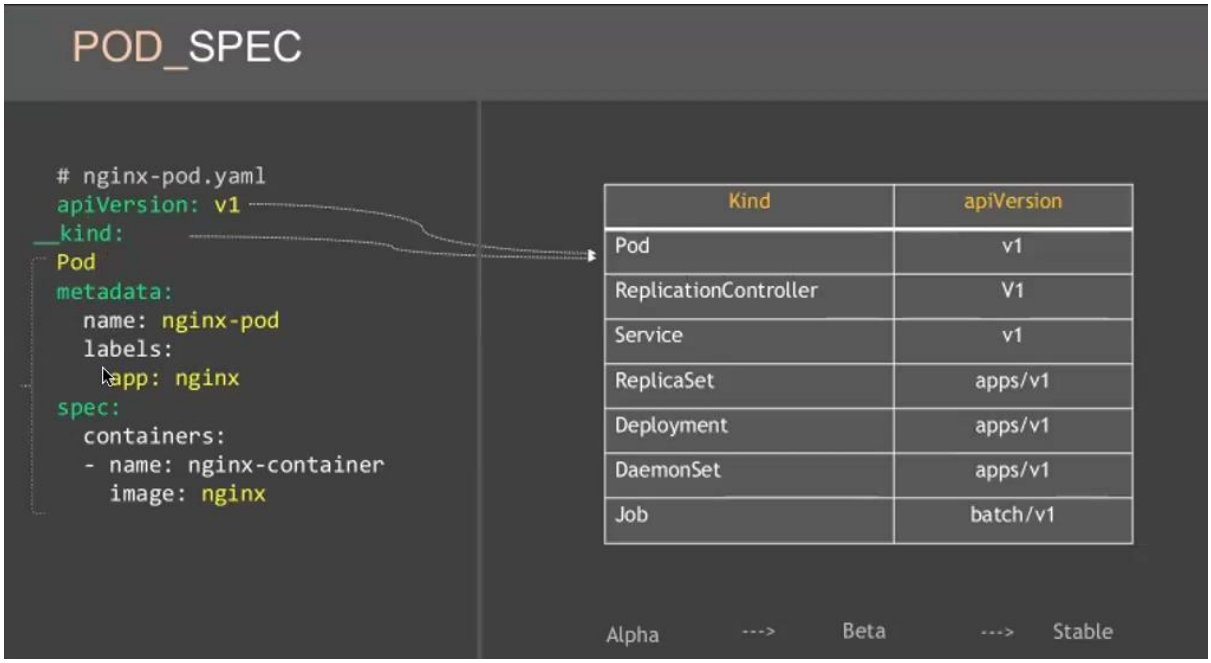
What is Pod?

Atomic Unit of Scheduling

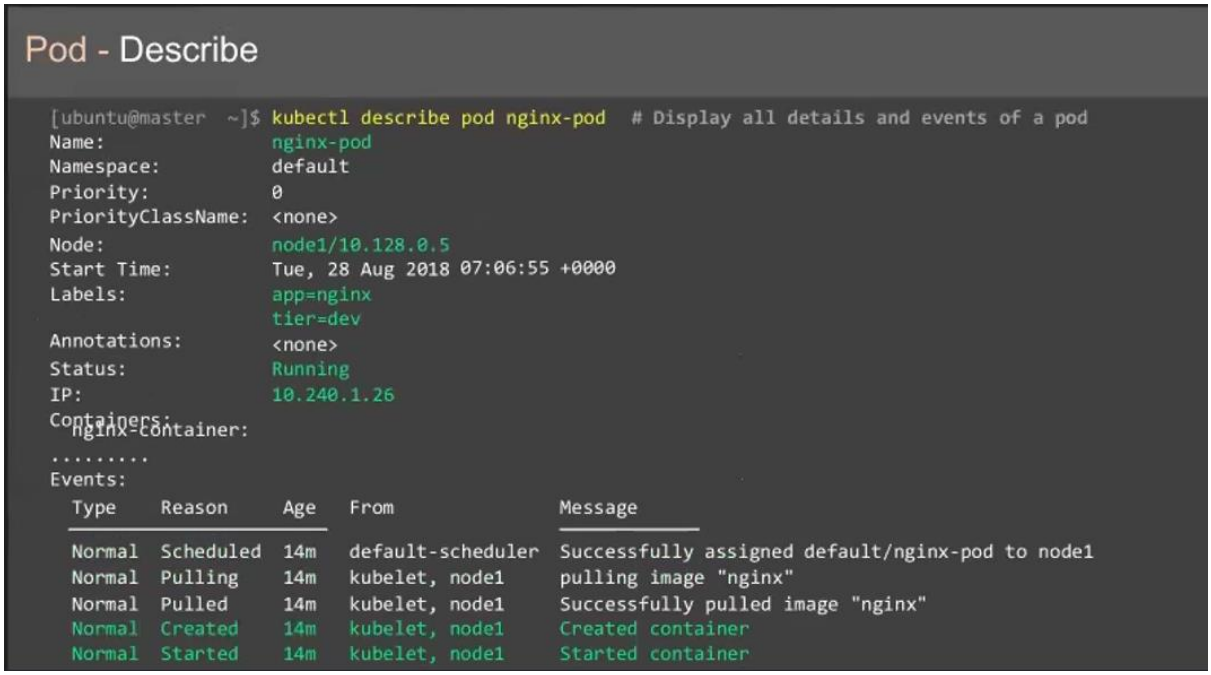


we will be creating a manifest file via that manifest file, your Kubernetes resources, or pod. Pod is one of your Kubernetes resource. Okay, can be created inside your cluster.

So, this is how your manifest file will look like it is. It is in the Yaml format. So, throughout your journey you will be writing this Yaml. You'll be writing lot of configuration files in the format.

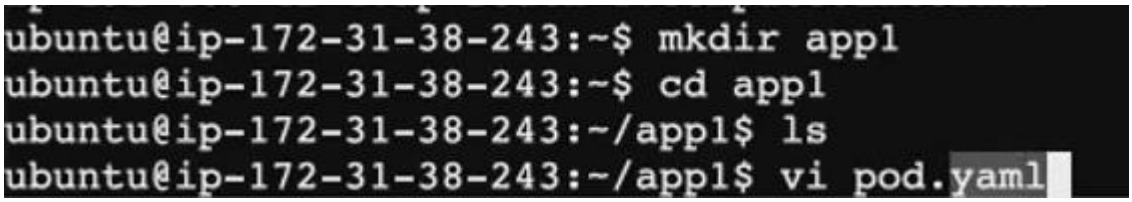


It will create a container, and this container will be launched inside your pod. A pod will be created inside the pod container will be created. A container will be launched with the image that you have specified.



Create a pod manifest file:

For manifest file, So always your pod manifest file will have an extension of Yaml. We are using Yaml format to create that manifest file.



Pod.yaml file:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
   app: web
spec:
 containers:
   - name: nginx-container
     image: nginx
```

Using kubectl we can launch pod resources inside Kubernetes cluster. kubeCtl is going to communicate this to the cluster.

```
ubuntu@ip-172-31-38-243:~/app1$ ls
pod.yaml
ubuntu@ip-172-31-38-243:~/app1$ kubectl create -f pod.yaml
pod/nginx-pod created
ubuntu@ip-172-31-38-243:~/app1$ kubectl get all
```

-f --- file name

Kubectl get all --- give all files inside the cluster

Kubectl get pod – shows pod creation

```
ubuntu@ip-172-31-38-243:~/app1$ kubectl get pod
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          61s
ubuntu@ip-172-31-38-243:~/app1$
```

you get the complete detail of the pod, how the pod is being created. Everything, all the details. You get it in the Yaml format.

```
ubuntu@ip-172-31-38-243:~/app1$ kubectl get pod -o yaml
```

now we have created a pod which is running our container and inside the container, you have your application running inside it. So, in case if you want to expose your pod, if you want to

access your pod outside the cluster. So now I have my Internet's application running. And I want to access my application through my browser.

since it's running inside the pod and your Kubernetes cluster is very secure. You cannot simply access the resources that is running inside your Kubernetes cluster. You have to create a file, you have to create a resource called service via service, you should be able to access the application that is running inside your pod.

In the real time we don't directly run the pod. We don't directly run the pod straightaway inside the Kubernetes cluster because pods are ephemeral in nature. your pods are ephemeral, which means it can be easily deleted.

create a resource called deployment:

what will happen if your pod gets deleted all of a sudden? Because now I have my application running in one of the pod. I have the application running only in this pod It's completely dependent on this pod. Something happens to this pod. My application will also be unavailable. In order to avoid that, what we do, we create another resource just to manage this pod. We **create a resource called deployment.**

So, you don't run your pod separately inside your cluster. Rather, you create your deployment. And inside the deployment you launch your pod and you run. You create your pod. Okay? So what we do now, we will go and create the deployment.

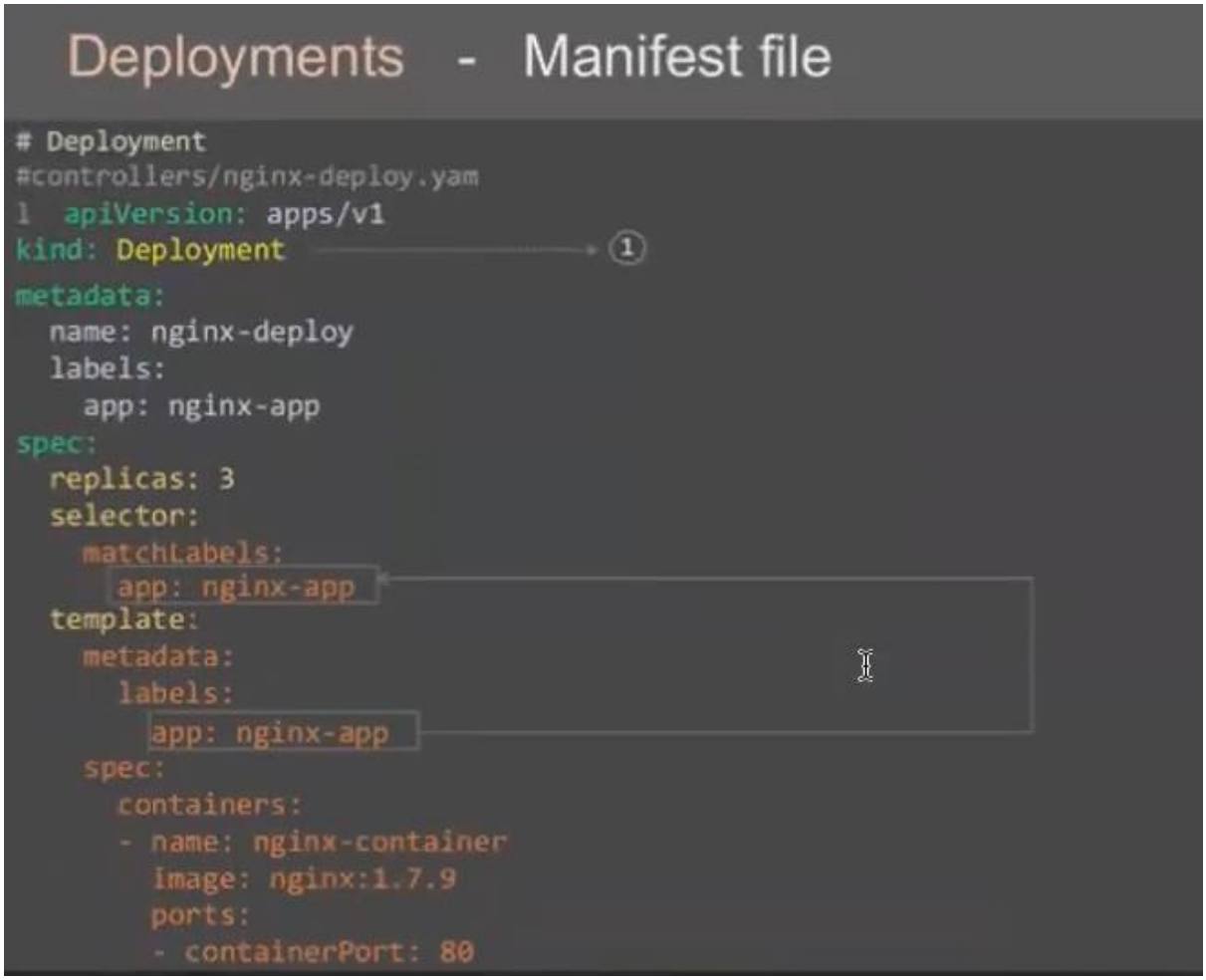
So deployment is like a super set to have a deployment inside the deployment to create your pod inside the pod You have your container. Okay? Deployment is also a resource of your Kubernetes cluster.

Replica set takes care of your replication of your pod. So, you can maintain multiple pods, multiple similar pods with the help of replica set. So, your deployment will take care of the replica set and the pod.

your deployment will create multiple replicas of your pod, you will. You will be able to upgrade your application. If you want to upgrade the nigix application. Now you're using the version 1.5, and tomorrow you want to upgrade it. Then to a different version that is also possible with the help of deployment.

but with the pod you cannot do it. So, with the help of deployment, you can upgrade your application, and in case you don't want the current version, and you want to roll back to the

previous version itself. You can also roll back your application to the previous version with the help of rollback command that we use with the deployment you can scale up and scale down based on the applications requirement, and you can also pause your application and assume your application.



- You either call it as a manifest file configuration file or a yaml file to create your deployment. We call it as deployment manifest file. So, as I told you, any manifest file that you use inside your cluster will only have 4 components. That is your Api version, kind metadata and spec for deployment.
- Your Api version is going to be apps/V1/ You mentioned the kind, the type of the resource that you're creating. You're creating a deployment and you give name of your deployment. And how do you identify the deployment inside? The Kubernetes cluster with the help of labels.
- here only the only your spec will be different. So your podspec was completely different. So in the podspec what you had, you had only the container, because in your inside the pod you only have your container. But now, inside your deployment, you have a pod within the pod. You have your container running.
- while creating your pod , same template, same things that you mentioned inside your template? So under your specification of your deployment. You mentioned the replicas

selector and template under your template parameter. You mentioned all your port details, you give all your port details. You mentioned the label of your pod.

Deployment.yaml file :

```
ubuntu@ip-172-31-38-243:~/app1$ vi deploy.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deploy
  labels:
    app: web
spec:
  replicas: 3
  selector:
   matchLabels:
    app: web
  template:
    metadata:
     labels:
      app: web
    spec:
      containers:
        - name: mycontainer
          image: nginx:1.7.9
          ports:
          - containerPort: 80
```

```
ubuntu@ip-172-31-38-243:~/app1$ kubectl create -f deploy.yaml
deployment.apps/nginx-deploy created
```

Pod created via deployment:

We gave 3 replicas so 3 pod created

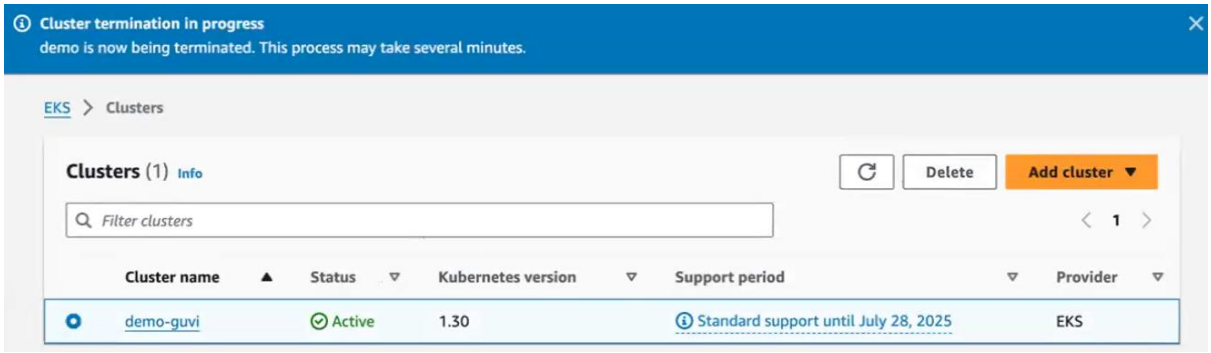
```
ubuntu@ip-172-31-38-243:~/app1$ kubectl get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/nginx-deploy-77cd89c769-qgbgd    0/1     Pending   0          11s
pod/nginx-deploy-77cd89c769-shtgw    0/1     Pending   0          11s
pod/nginx-deploy-77cd89c769-sqt7f    1/1     Running   0          11s
pod/nginx-pod                        1/1     Running   0          23m

NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.100.0.1    <none>        443/TCP   59m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deploy        1/3     3            1           11s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deploy-77cd89c769        3         3         1       11s
ubuntu@ip-172-31-38-243:~/app1$
```

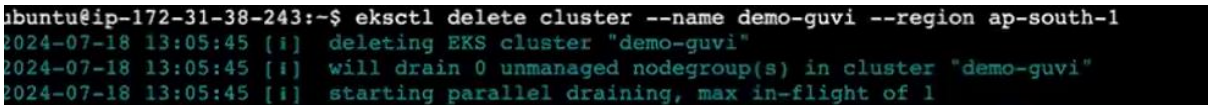
```
ubuntu@ip-172-31-38-243:~/app1$ kubectl get deploy
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deploy   1/3     3            1           103s
ubuntu@ip-172-31-38-243:~/app1$ kubectl describe deploy nginx-deploy
Name:                   nginx-deploy
Namespace:              default
CreationTimestamp:      Thu, 18 Jul 2024 12:56:43 +0000
Labels:                 app=web
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=web
Replicas:               3 desired | 3 updated | 3 total | 1 available
 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=web
  Containers:
   mycontainer:
    Image:        nginx:1.7.9
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
```

https://docs.google.com/document/d/1w4DZAbVGEZLe4pnzGAzi-d3LcNh5wZtQ/edit#heading=h.gjdgxs

[pod creation document]

Delete the cluster immediately:





eksctl delete cluster --name demo-guvi --region ap-south-1