Kubernetes Architecture

As of now we have understood run the applications in the monolithic fashion, containerized model

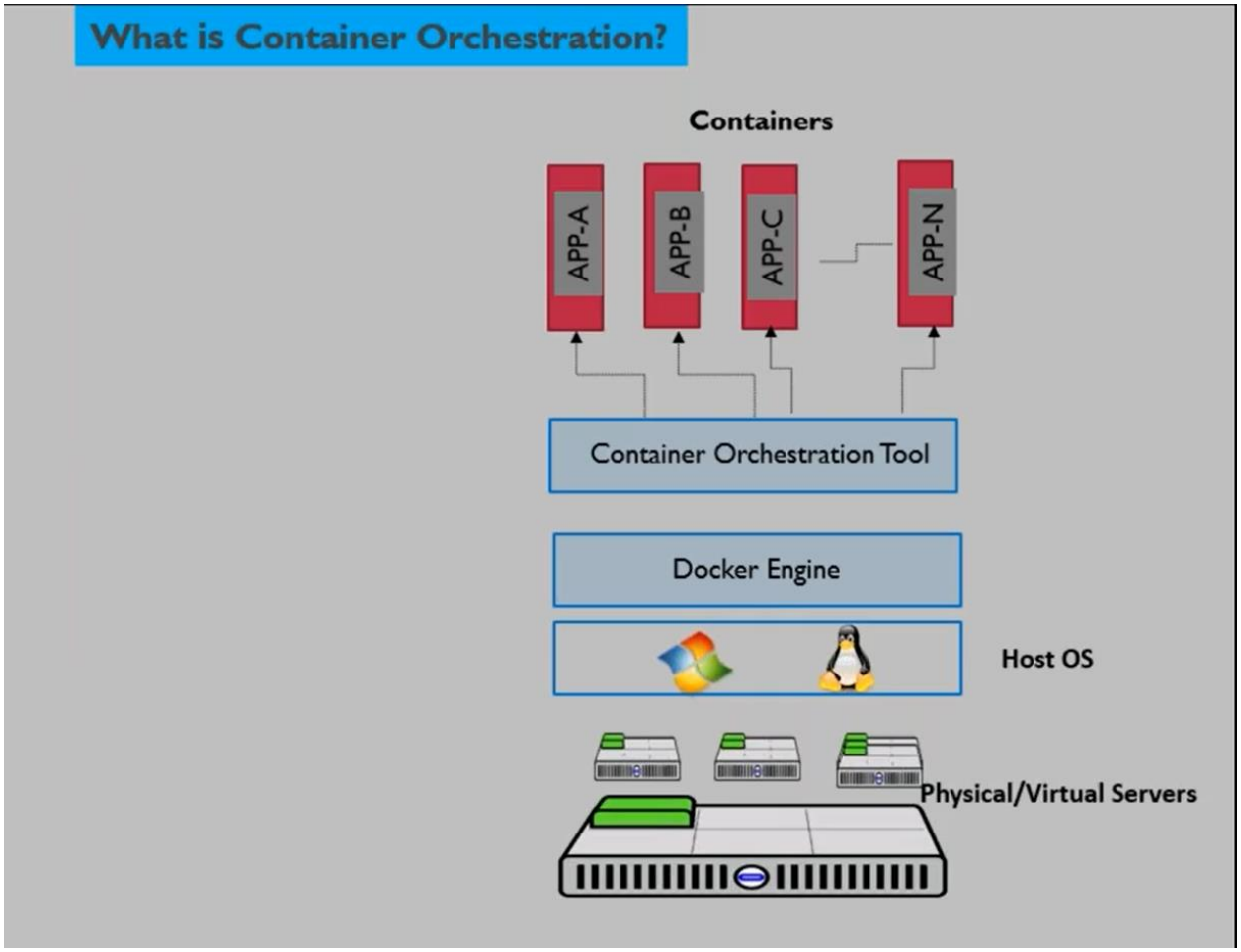
First will see what the challenges we have on containerized model are

When we are using the docker engine to manage the containers, we are not achieved the auto scaling, high availability, load balancing

To achieve all these challenges on top of docker engine now am going to install a one more tool Containerization Orchestration Tool

The Container Orchestration Tool is using on top of the docker engine help us to manage the container life cycle

Container Orchestration Tool will take of life cycle of containers [managing the containers]



In order to fix the challenges the container orchestration tool doesn't have the capability to create and manage the container life cycle

The Container Orchestration Tool always depends on the container engine only to manage the containers life cycle

Managerial decisions now will take care by the container orchestration tool, the

COT will take care of monitoring the application if not healthy that will be replaced with new server

COT only take care of wise decisions what needs to be done however the actual container creation process is not the responsible of COT that scenario still be take care by the container engine only

Decisions taken by the COT

Actions do by docker engine

Without docker engine the COT not do anything

The instruction sends by COT accordingly Docker engine will do the necessary things

As of now will see the challenges again

Mainly

Self healing

High availability at container level

Load balancing

High availability at host level

Auto scaling

COT will take care of provision and deployment of Containers

COT will take care of Redundancy and high availability of containers

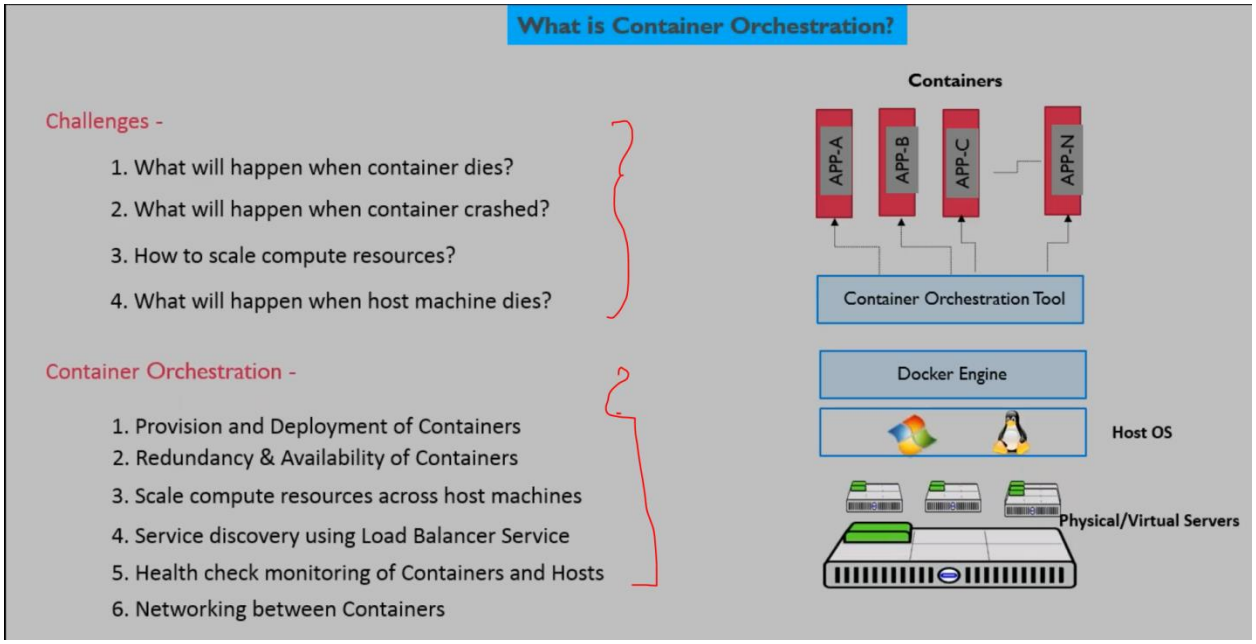
COT also take care by the auto scaling

Service Discovery is the concept which will take care by the load balancing

COT tool also do the scaling up of the computer resources of host machines.

COT tool also take care of heal check if anything not working it will replaces

COT tool also manages the networking between the containers



There are multiple orchestration tools are available in the market

Kubernetes is the one COT tool as well as Docker Swarm engine also available

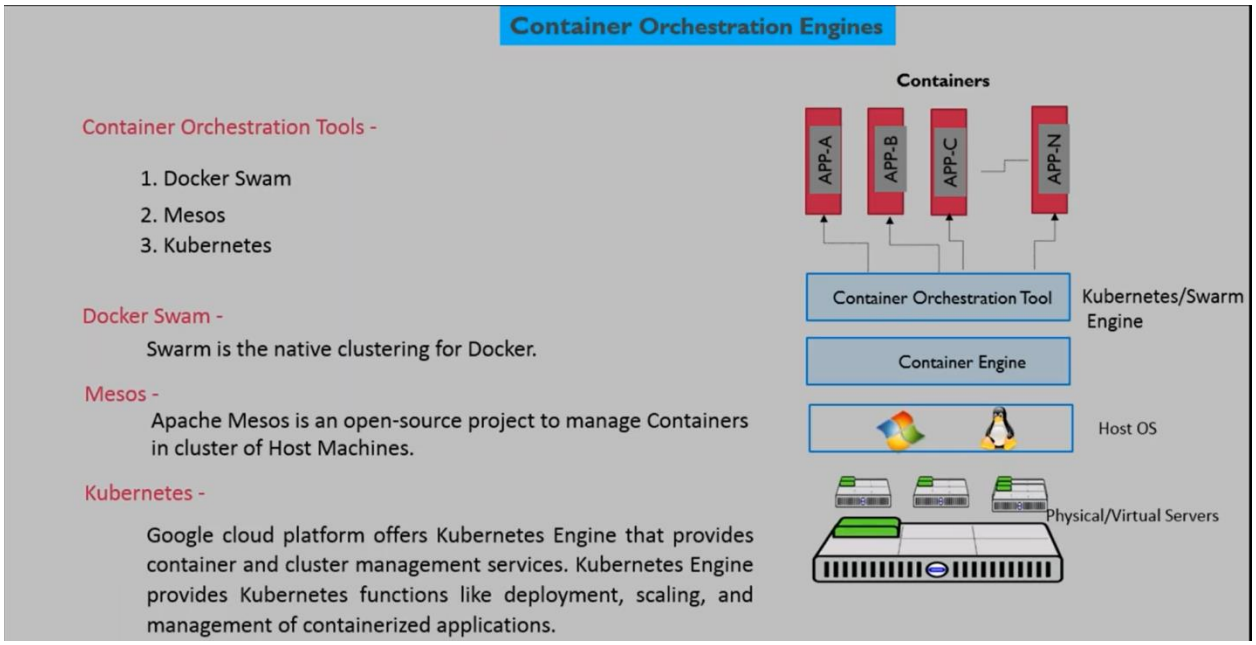
Will see some of COT tools

1. **Docker Swam**
2. **Mesos [from Apache people]**
3. **Kubernetes**

Docker Swam don't have much features

Mesos – not a great architecture.

Kubernetes:



Kubernetes is also the open source from GOOGLE

Now will kubernetes

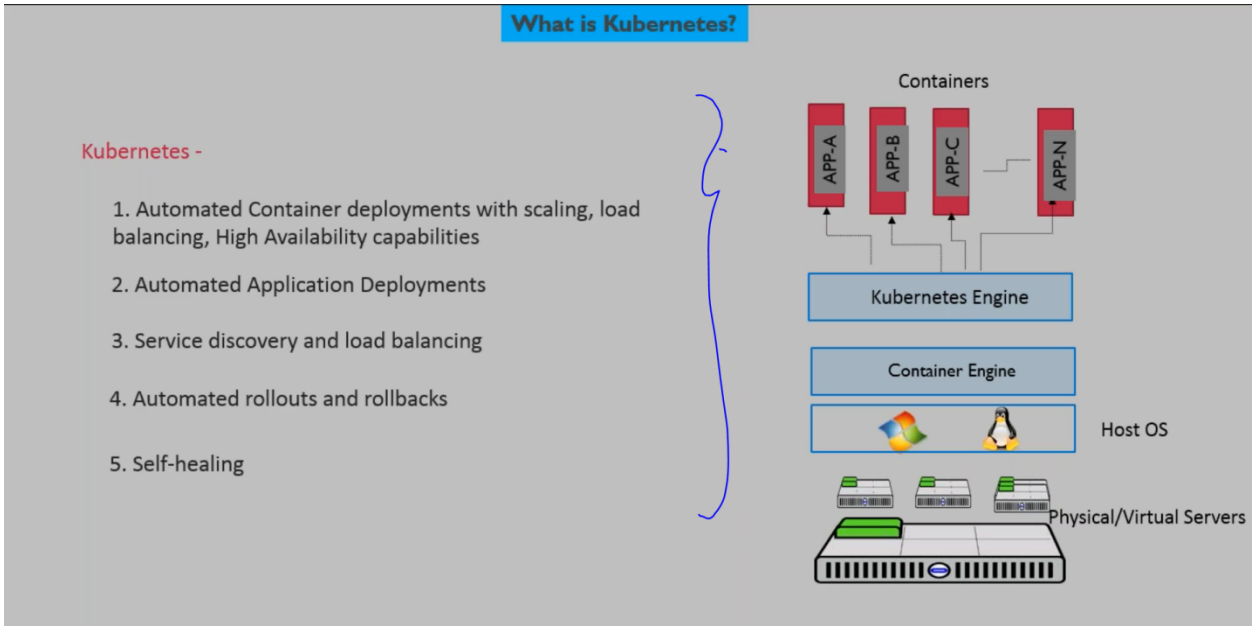
What is Kubernetes?

Kubernetes is the container orchestration engine

Kubernetes engine doesn't have the capability to create and managing the containers directly, Kubernetes always act as the Manager actual thing did by the docker only

Kindly see the below what kubernetes did

Self healing if the server is not healthy then the kubernetes will do the replacement of the server



Kubernetes also managing the secrets and configurations

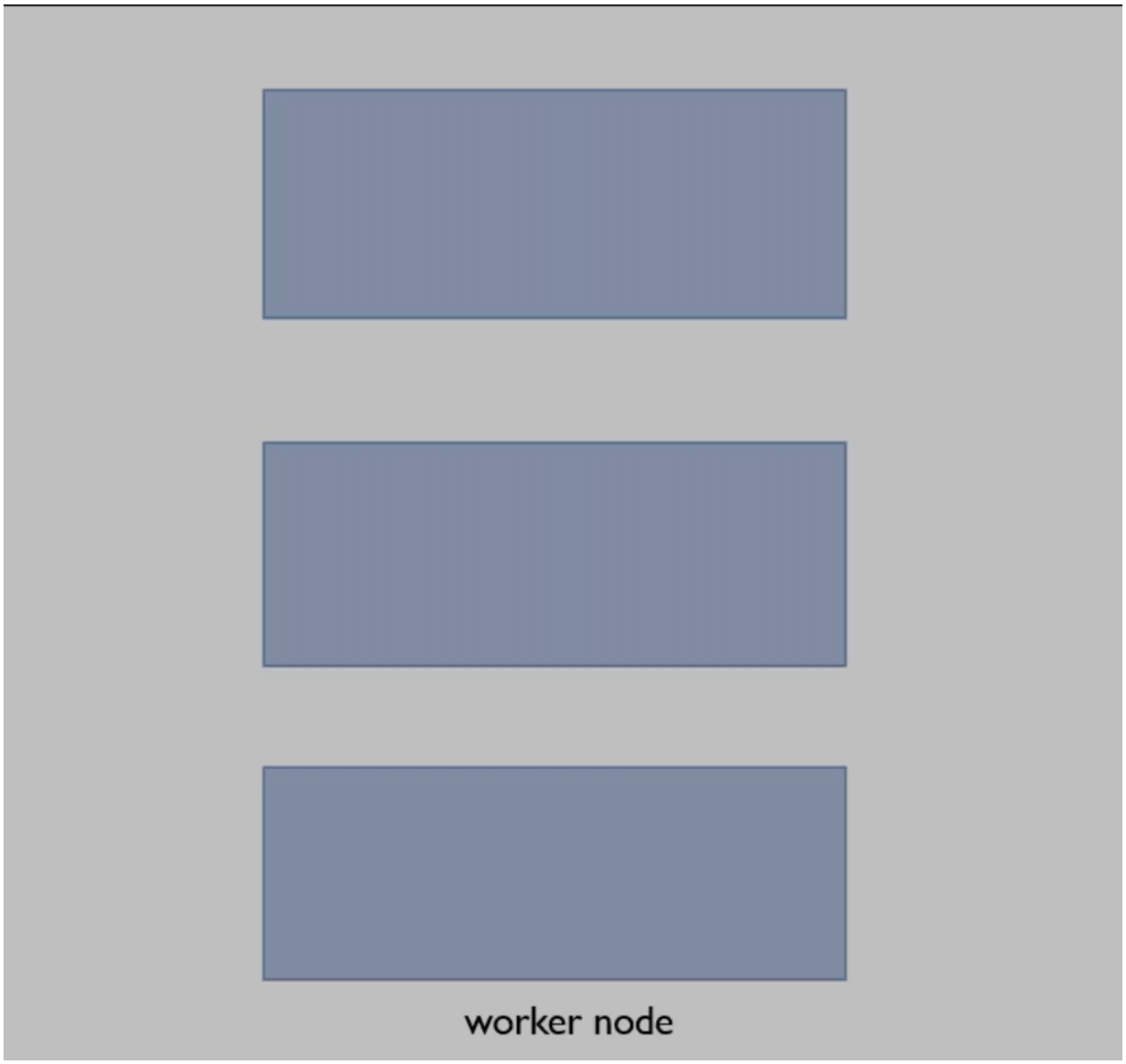
Now we have to understand the how kubernetes architecture will work

As we know we have to provide the high availability at host machine level

I have to maintain the multiple host machines rather maintain only one host machine

Host machines now are calling as WORKER NODE

In my architecture have the 3 worker nodes if incase one worker node fail automatically another worker node will come and use it



The number of workers node need that depends on the size of the application

You can take how many worker nodes you needed

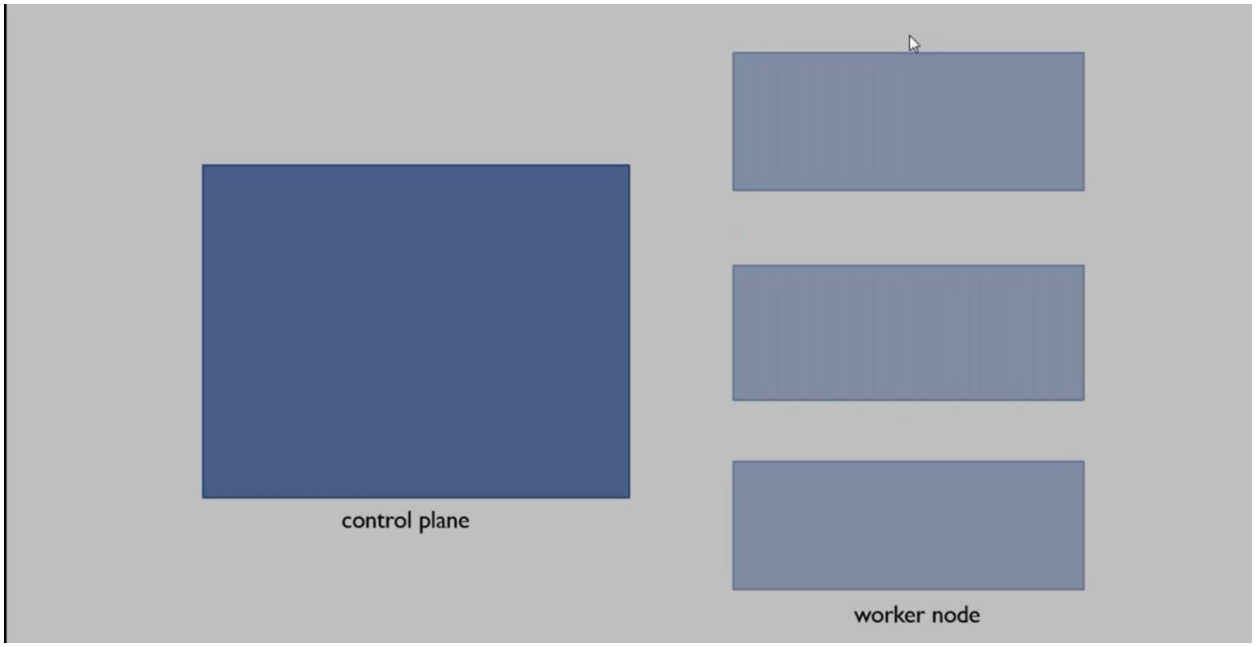
Typically 3 worker nodes are ideally any kubernetes set up

When application is running on these worker nodes these worker nodes need to be manage someone [let say one worker node is crashes I need to be run the same application on 2nd worker node] who is taking care of monitoring , who is taking care of monitoring fail , who is moving your application from one worker node to another worker node

The worker nodes here I was taken just to run the applications the actual management of worker nodes taking care by the one more instance [CONTROL PLANE]

CONTROL PLANE will take care of WORKER NODES and take the necessary action if anything fails

As of now I have the 4 servers 1 is control plane and another 3 are worker nodes



Q> where the business application will run?

Is it running on the control plane or worker nodes?

Always business application will run on the worker nodes only it could be any worker node

The control plane just take care of monitoring of your application

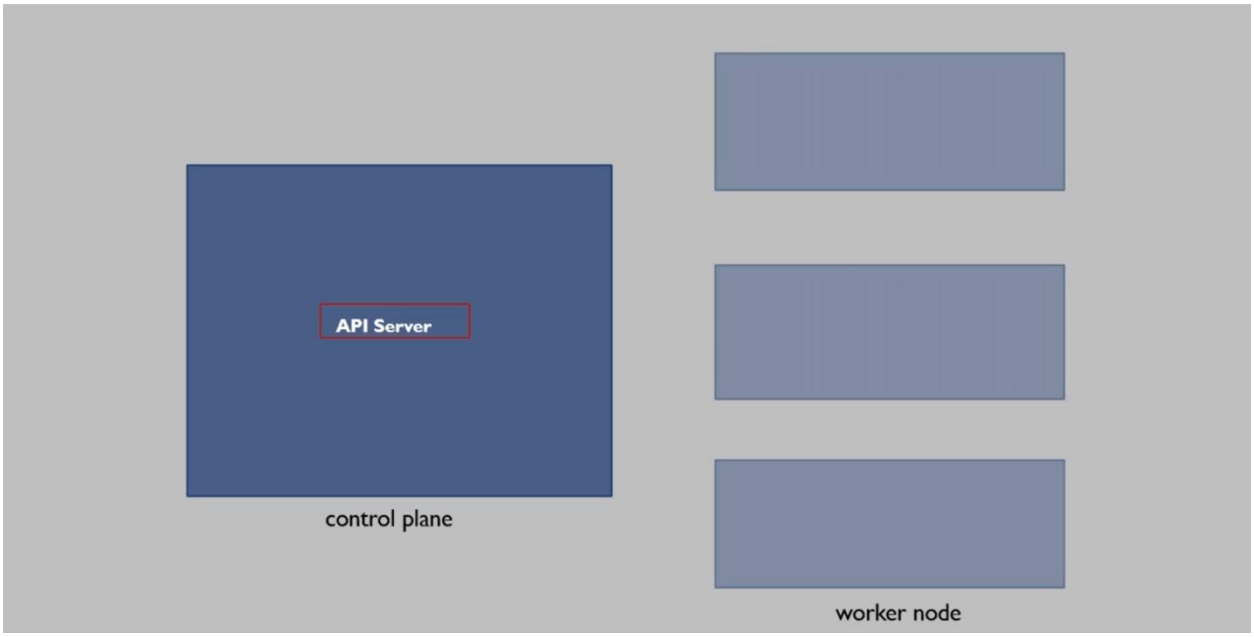
In Control plane I need to install one service that service take care of the entire operation

The entire service taking care by the service called API server

The control plane is the set of components that "make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied)."

The API Server is a service that is running on the control plane, the API service managing the entire operations

API server is not enough to take care of everything

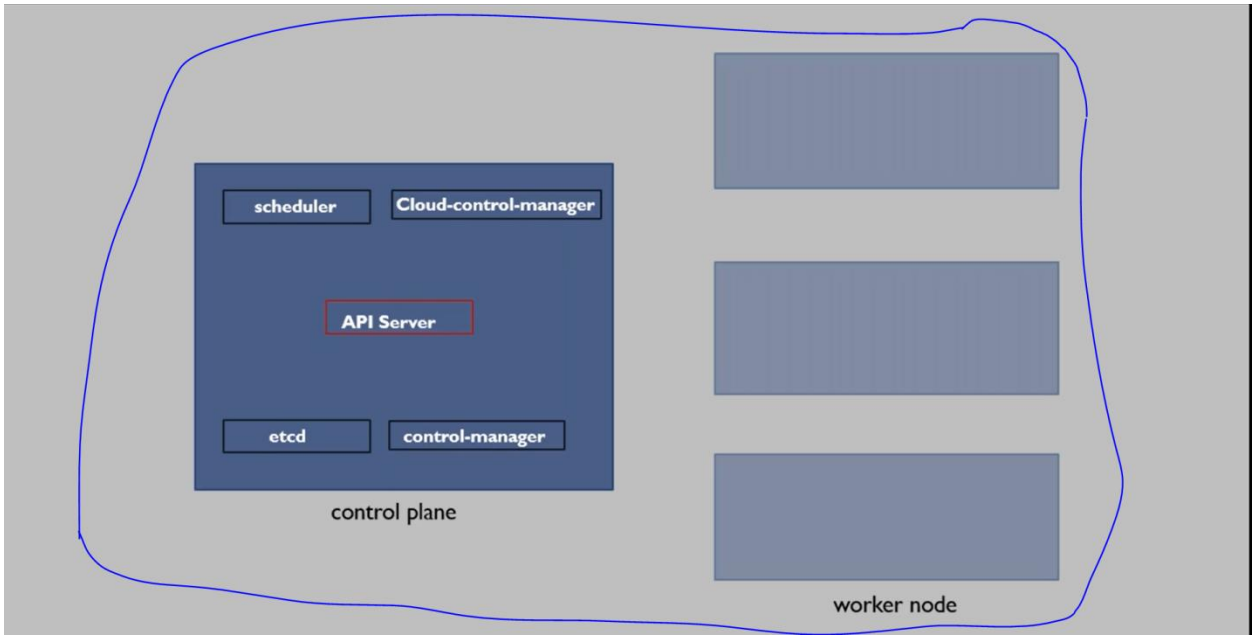


API server need to take of Monitoring the worker nodes, application, maintain the high availability , service discovery, how to recover the worker node, load balancing, how to recover the container node there are multiple things all these this not only take of API server

In your control plane server we have one more service called scheduler service, scheduler service is the another service you must be installed and configured in the control plane

Scheduler service responsibity is scheduling your application on your worker nodes [let say I have to be run the node application your node application which worker node need to be run out of 3 scheduler will take the decision by default

the scheduler will check the what is current computing utilization all these worker nodes whichever the worker node has the high amount free amount of computing available it will going that worker node]



Scheduler will follow the default scheduling policy stating that whichever the worker node has the free high computing resource that going to be pick up also you can send the your own instructions to your scheduler

Scheduler just will take the decision like which worker node your application need to run

One running on the worker node who is the responsible guy to watch the application whether your worker node is healthy or not, your container healthy or not

There is a control manager service that is running inside the control plane, this control manager is the responsible in managing the running workloads, workload another new terminology here am using whenever I say workload mean your application

Control Manager will monitoring your running application and reports if anything failure

Etcd:

Now I would require a database here

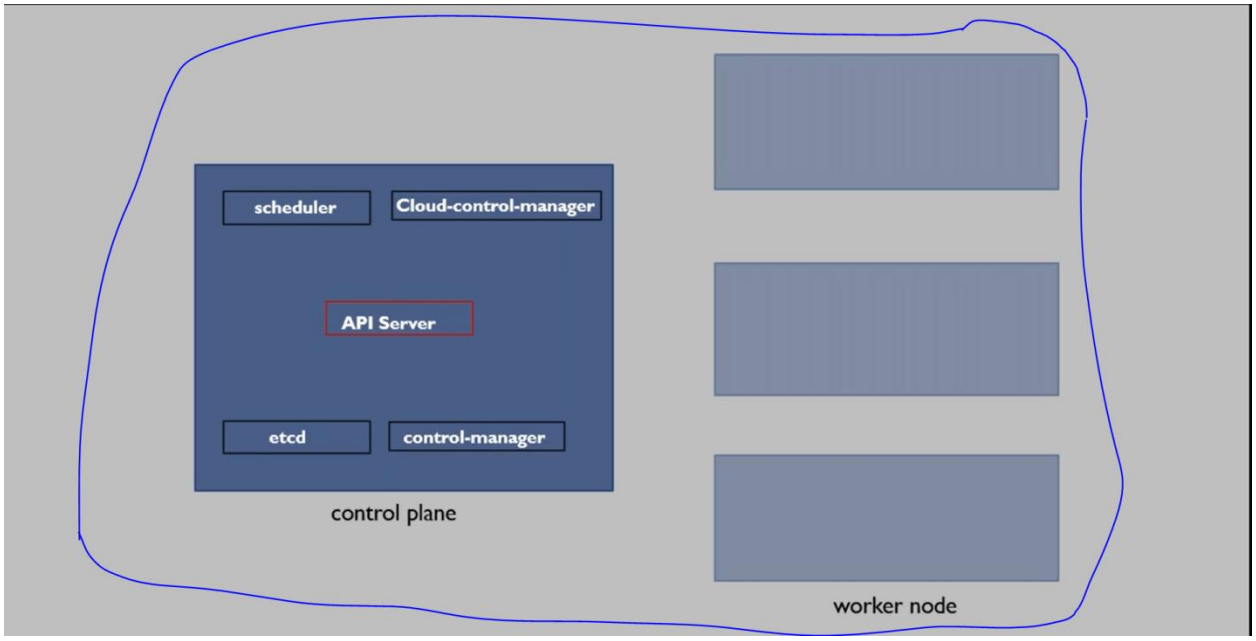
One guy definite have the all data like how many containers are running, in which container the application is running etc…

Etcd is the database where the metadata is stored this entire eco space means in the etcd database had the data how many worker nodes are there , what are the ips of worker nodes , how many containers , where the application is running

In one word the Meta data about your entire cluster will be now in the etcd

You have one more service called Cloud-control-Manager

Cluster -> Control plane + worker nodes



Cluster is the group of servers

The Kubernetes cluster metadata will stored in the etcd database

This Kubernetes cluster can be done which type of servers?

The kubernetes cluster can be set up at anywhere [may be physical/ virtual servers]

You can kept in the on premises server also

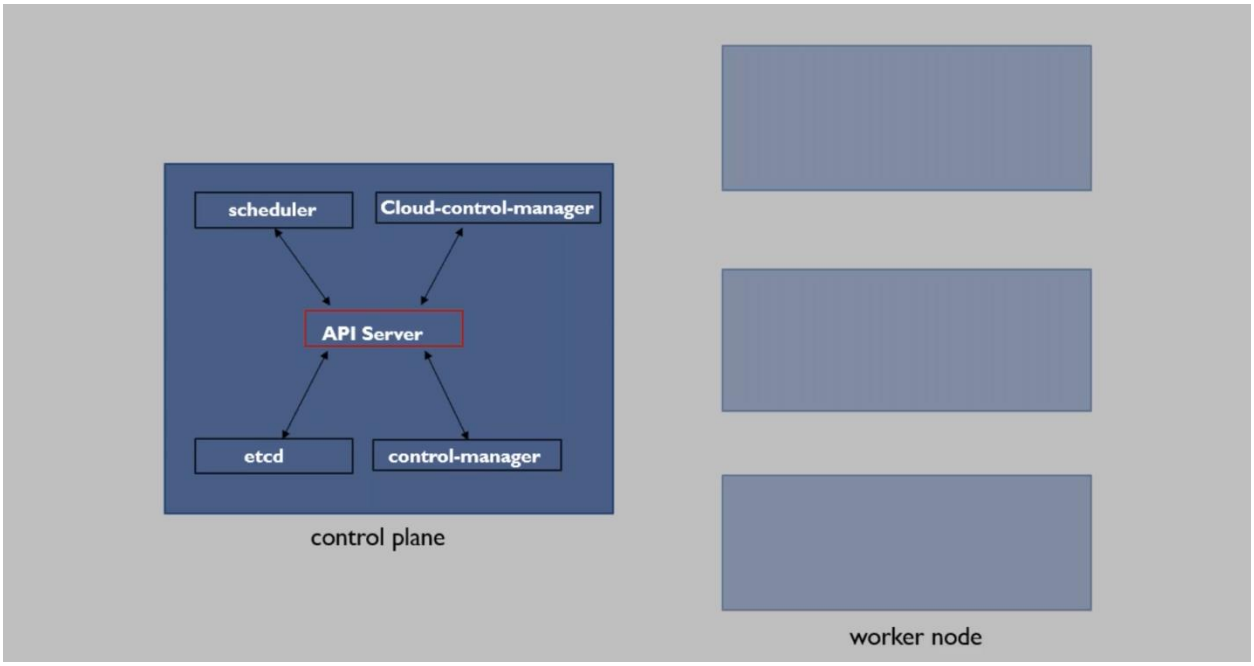
The Cloud control Manager that is available in control plane node that cloud control manager having the required dependencies to support running this cluster into your cloud platform

Let say I want run the kubernetes cluster on aws level, the CCM has the required dependencies to support you to run

Cloud related things provided by the CCM

If you are running kubernetes cluster on-premises you no need of cloud control manager

Whatever we take those services reported to the API server



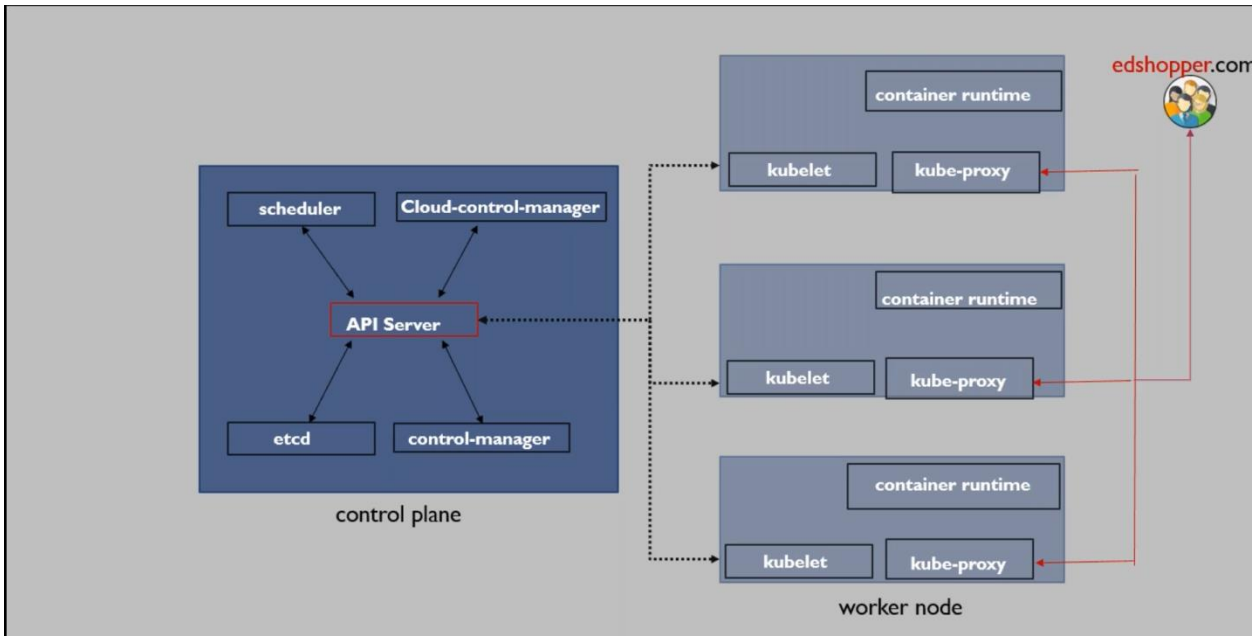
The scheduler will schedule the worker node which worker node needs to run for the particular application information will give to API server

Control Manager Keep on monitoring your work load [application] if going wrong and that report will give to API server

API server has the permission to read the information will update in the database accordingly

API server is the actual manager

API server responsibility to inform the action item to the worker node Only API server will go and talk with the worker nodes whenever some changes would be required



In the worker node side there is one more service called kubelet, kubelet is the service and must be installed and running on the worker nodes

The kubelet is responsible to listen to the request given by the API server then the kubelet responsibility to implement that action item on the worker nodes

Let say scheduler decides that this needs to go worker node 3 then the API server will go and talk to worker node 3 will pass the instruction what needs to be done to the kubelet

As I said before kubernetes is just the orchestration tool only the managerial role to maintain the complete operation to ensure that your application should be highly available, scalable

The kubernetes doesn't have a direct control to go and start creating the services as well as maintain the life cycle of containers

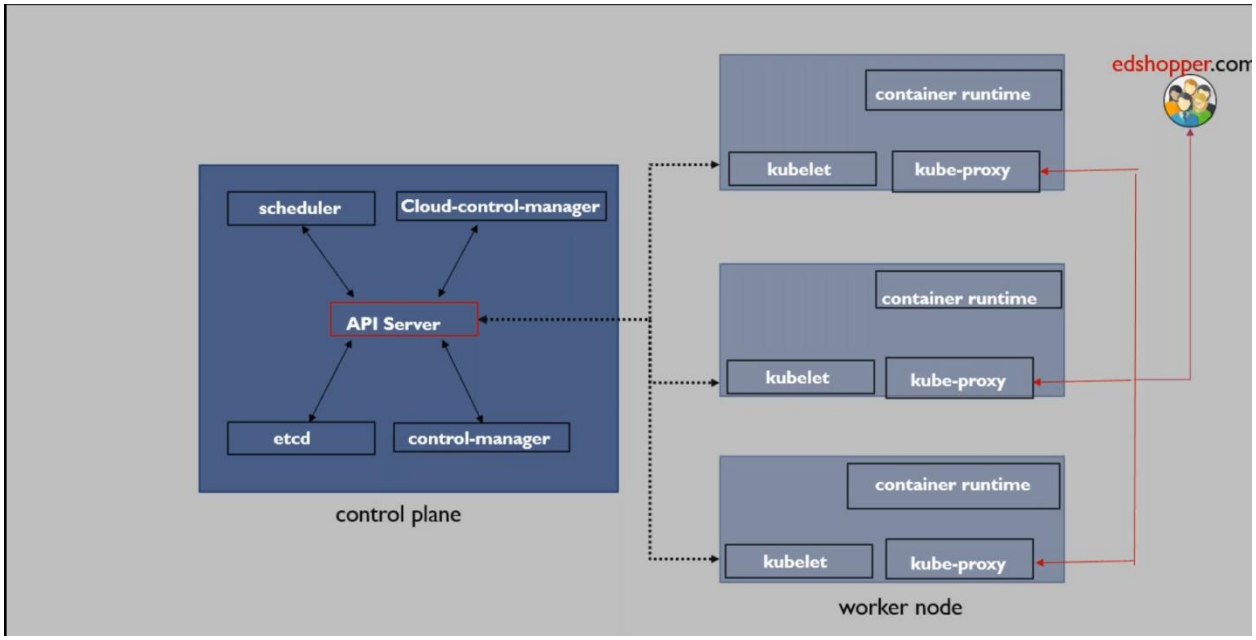
In your worker node should be the container runtime is should be installed, container runtime or container engine must be running in your worker node

What is the container need to run will it be a docker container or will it be a containerized?

As I said before containerized will take of all the things about container, here you have to install containerized not the docker engine

I am not using the docker container as the container engine here, I am going to use container d only on top of containerized am going to set up the kubernetes cluster

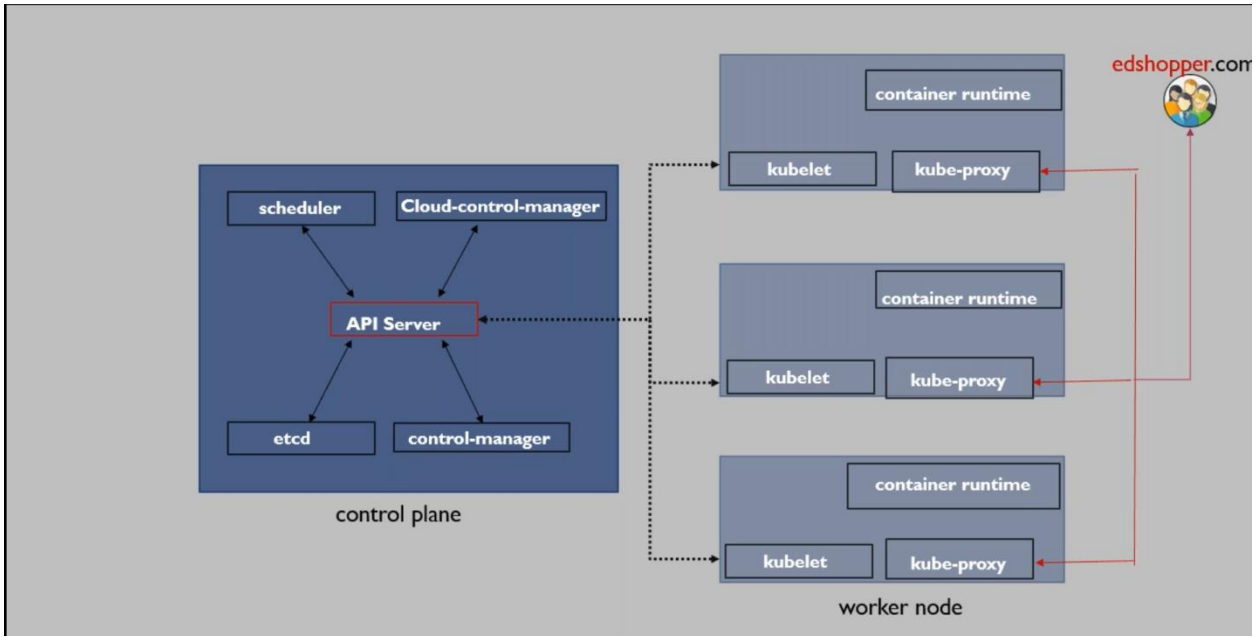
At worker node side there is one more service called kube-proxy



The kube-proxy is the server is must be running on your worker node side, the kube-proxy is responsible in providing you the connectivity between the workloads running across the different worker nodes

Let say I have one application running on one worker node, another application is running on the another worker node how there two application will work together kube-proxy will help for this

Kube-proxy not only give you the connectivity between the applications, kube-proxy also provide the externalize access to the end users whoever accessing the your application



Let say this is my kubernetes cluster one control plane and 3 worker nodes , I wanted to run a node application where should I be running my node application, earlier it was one host machine I will go to host machine and make sure that the docker engine is there in running state probably I will use the command called docker run to create the container so easy before but now typical cluster architecture having multiple worker nodes I want to run my application where I need to be run should I need to go each worker node and start the application or let my scheduler taking care the decision on which worker node

Control plane will take care of it you don't need connect to worker nodes for anything if anything need to do work you have to send the instructions to control plane all instructions will go to the API server and API server will talk with the worker nodes

As you a devops enginer to do the deployment you has to connect to the API server

In order to connect to API server you should require the some command line client software [kubectl]

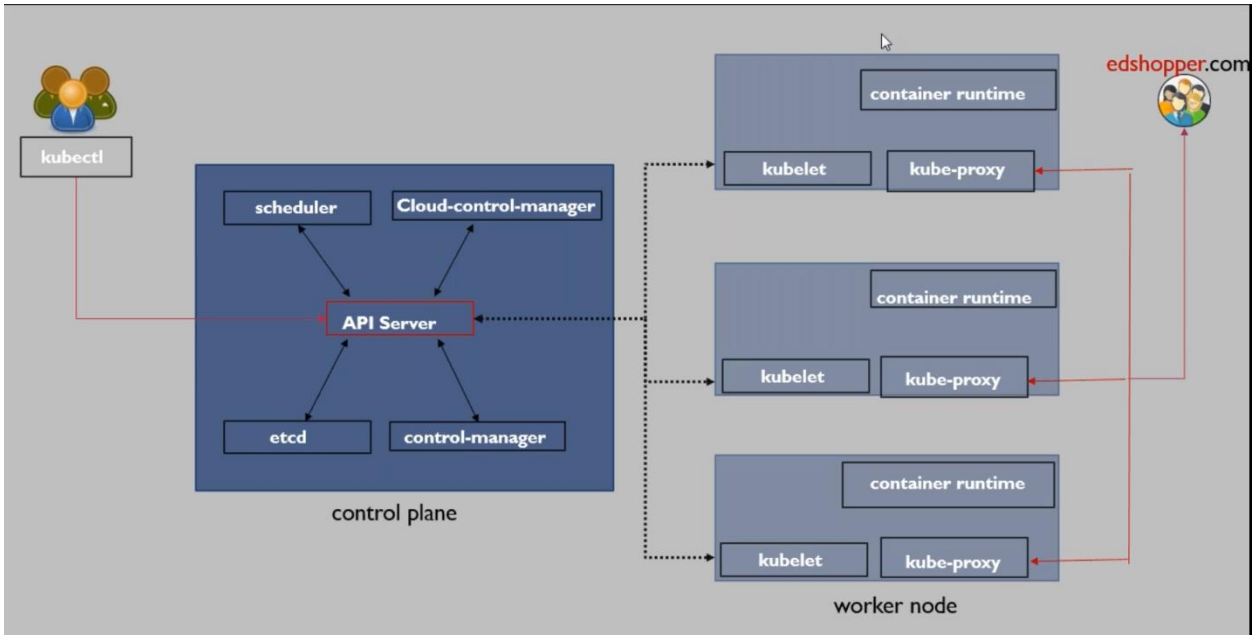
Kubectl is the kubernetes client which has the capability to talk with the API server

You will always run the kubectl commands that kubectl commands will talk to the API server will give the necessary input to the API server, the API server will take care of running your application any of worker node based on the scheduler policy

Means once the kubernetes cluster is ready, you need to talk with API server do the necessary things what you needed

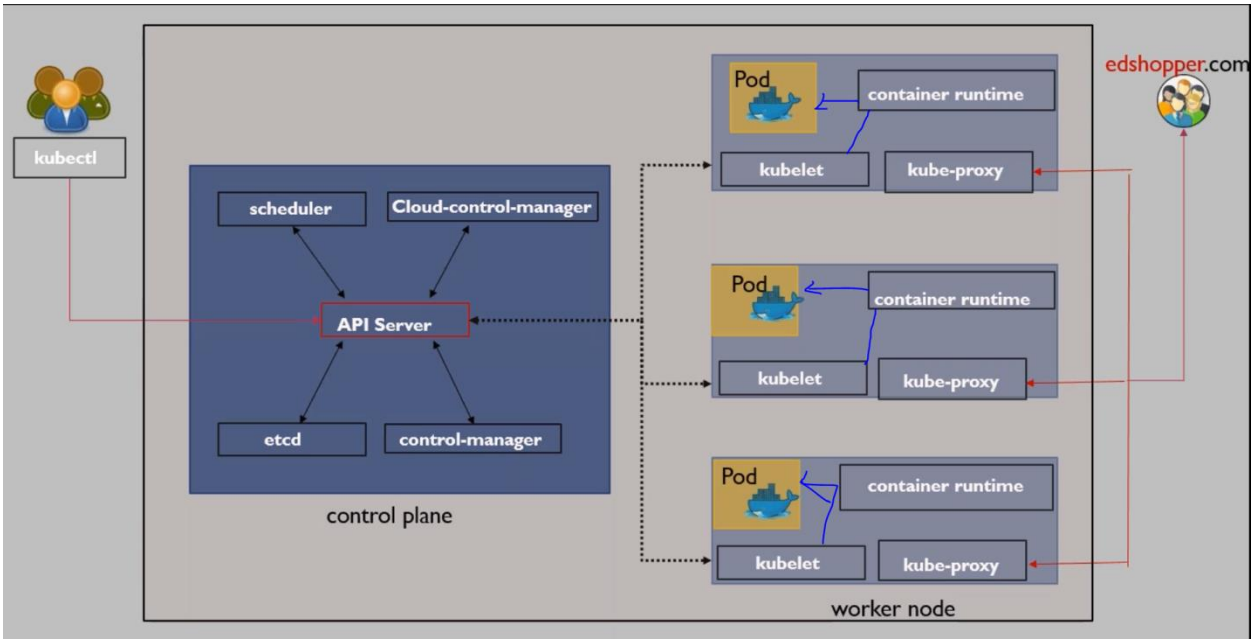
Can I say now kubernetes helping the automated deployment?

Yes because I don't to look into which worker node need to run my application, only need to inform to the API server



Let say using the kubectl you giving the instruction to the API server stating that hey this is the workload I wanted to run the API server now actually taking care of doing the deployment

The API server actually informing to the Kubelet in the worker node the kubelet is the service which is taking care of this , the kubelet will receive the instruction , the kubelet will go and talk to the container runtime then your container will get be started



During the container creation time as I said kubernetes don't have the permission directly to create the container still go through the container runtime only

Since my kubernetes don't have the permission always should be need to go via container runtime it is a tricky process right hence the reason kubernetes will create the one more wrapper here around

Whenever the your application running as container, my kubernetes need details of the container

Kubernetes will create the one more wrapper around your container, whatever the kubernetes creating the around the container that is called as POD

POD is the smallest executable compute in it in the kubernetes [one extra layer will be created around our container to export the container properties as the POD properties

Means now onwards if my kubernetes wanted to see the status of your container the kubernetes will see the status of POD, POD is name where extra layer will be created on top of the running container now all the characteristic available as the POD characteristics

So if my kubernetes wanted to see what happening inside my container it always look at POD level, POD is within the control of kubernetes

The container not actually control of kubernetes, container in the control of container runtime

To increase the flexibility of container the kubernetes actually creating one more around it whatever the properties contain the container those properties inherited to POD

This kubernetes work as a pod level

From kubernetes perspective let say I want to run an application should I be run as a container or should I be running by application as the pod

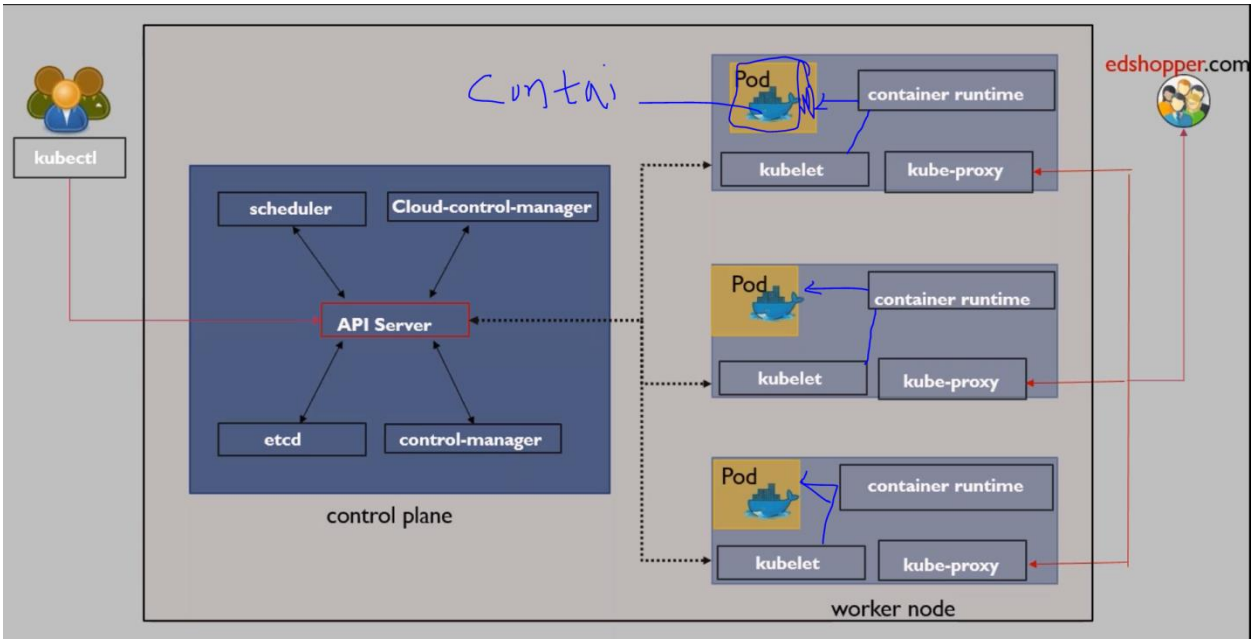
Which will be the right terminology as per the kubernetes if POD

So my statement will be I need to run a application as POD, of course when I say pod inside container only

POD is another term

1 Pod always had one application in it

You cannot have a multiple containers in a single POD



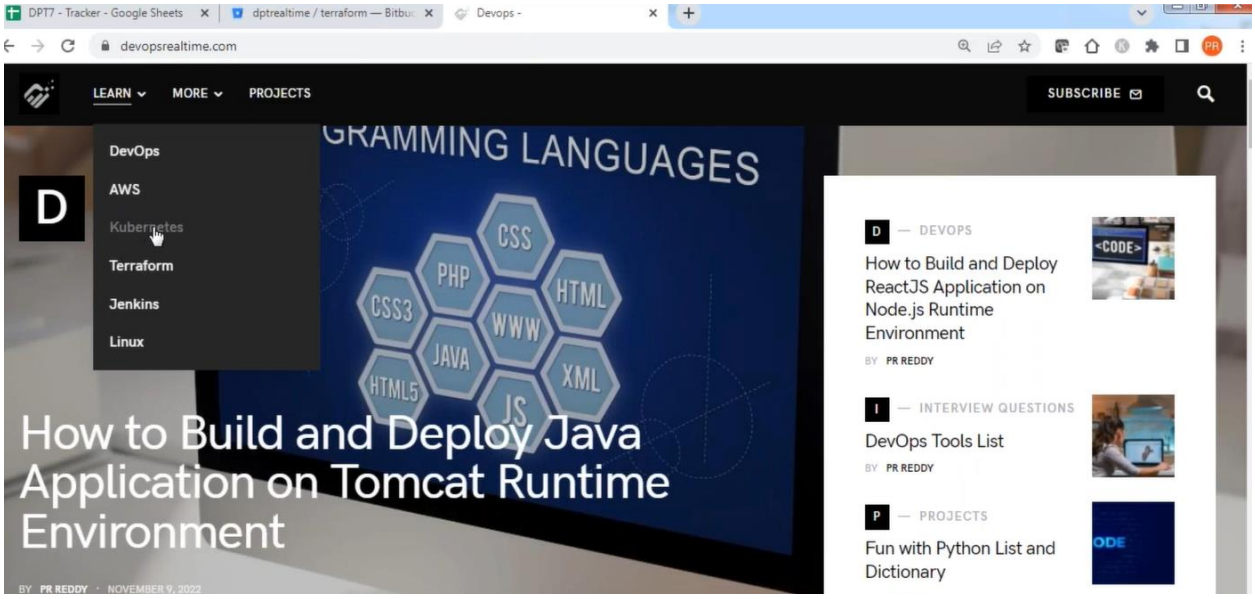
Whenever the developer will give you the application as a devops engineer end goal is you have to run the application as a pod

Now onwards called it as POD only not as container

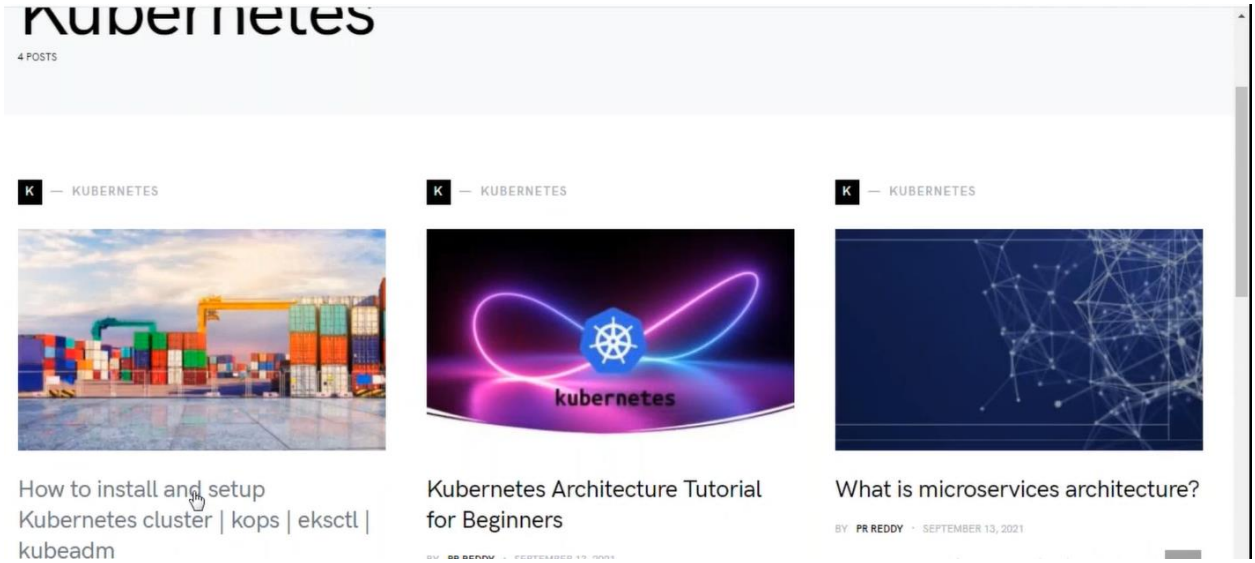
Container life cycle done by the container runtime , once the container is created then container properties inherited as a POD level , the kubernetes always looking as pod level only for doing the a operations

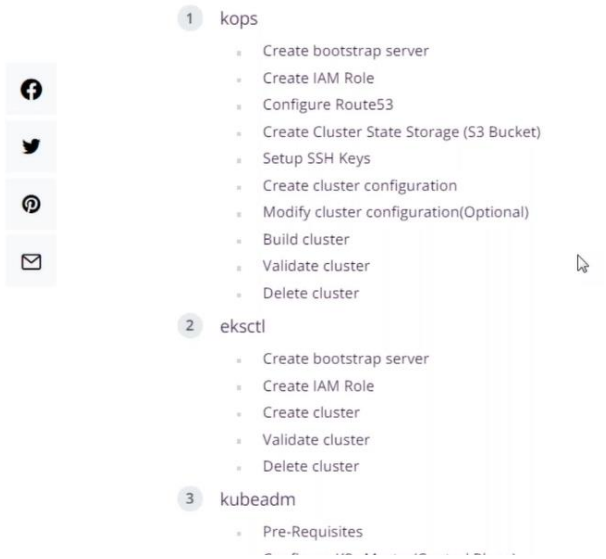
Setting up kubernetes cluster from SCRATCH is very tricky

https://devopsrealtime.com/how-to-install-and-setup-kubernetes-cluster-kops-eksctl-kubeadm/



There are different ways are there how you are setting up of the kubernetes cluster





Top 3 methods that you can set up your kubernetes cluster

1. **Kops –easy way**
2. **Eksctl – less steps**
3. **Kubadm –**

Hope you set up the kubernetes cluster and working as fine what is the case if control plane is down, is there any impact to work load

The running worknode there is no problem because the application is running on the worknodes incase control plane is got terminated but without control plane you will not do a managerial operation.

Let say any application is failed it will not be recovered because your manager [control plane] is not available

Let say one of the worker node is failed is there any impact to business?

Yes, impact is less because the other 2 worker nodes will help

My worker node should be high available, if in case worker node fail have to do worker node replacement as soon as possible

The business not happy to install and do the things from scratch onwards rather business will managed kubernetes services

Hope you know what managed [could will provide you just need to use]

And self hosted [you install, you configure, you manage]

When I go for cloud entire kubernetes architecture will now be managed by taking care of cloud vendor itself [1:20]

Amazon Elastic Kubernetes service (EKS)

Azure Kubernetes service (AKS)

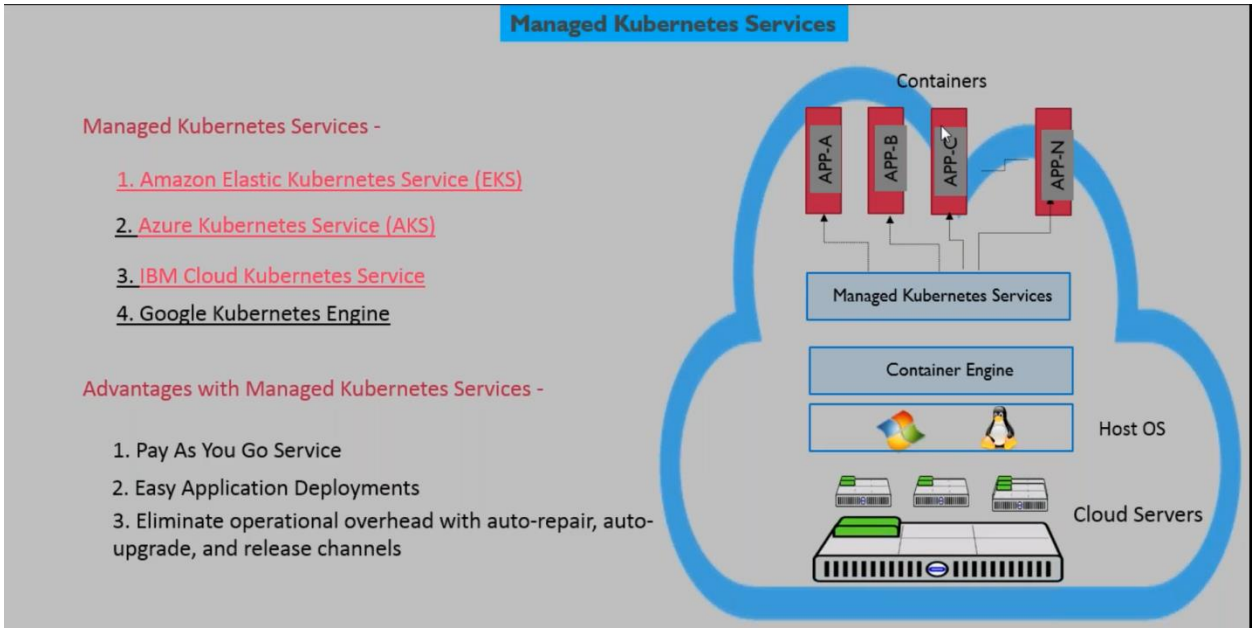
IBM Cloud Kubernetes service

Google Kubernetes Engineer

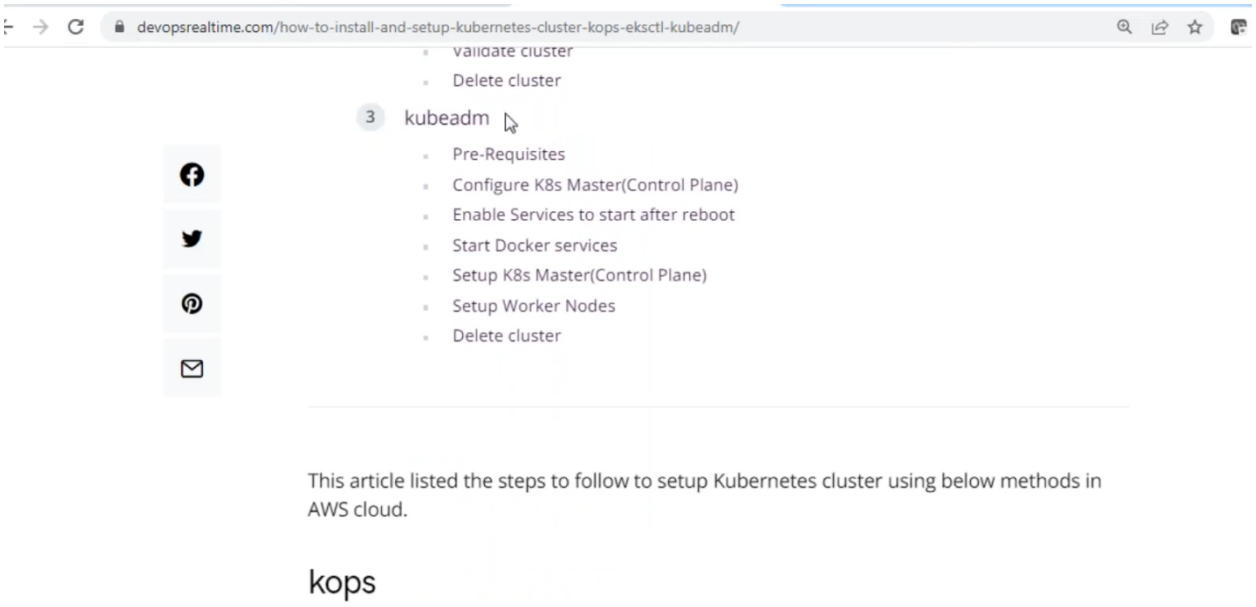
These top kubernetes service according to the cloud

In our case we will be use the EKS

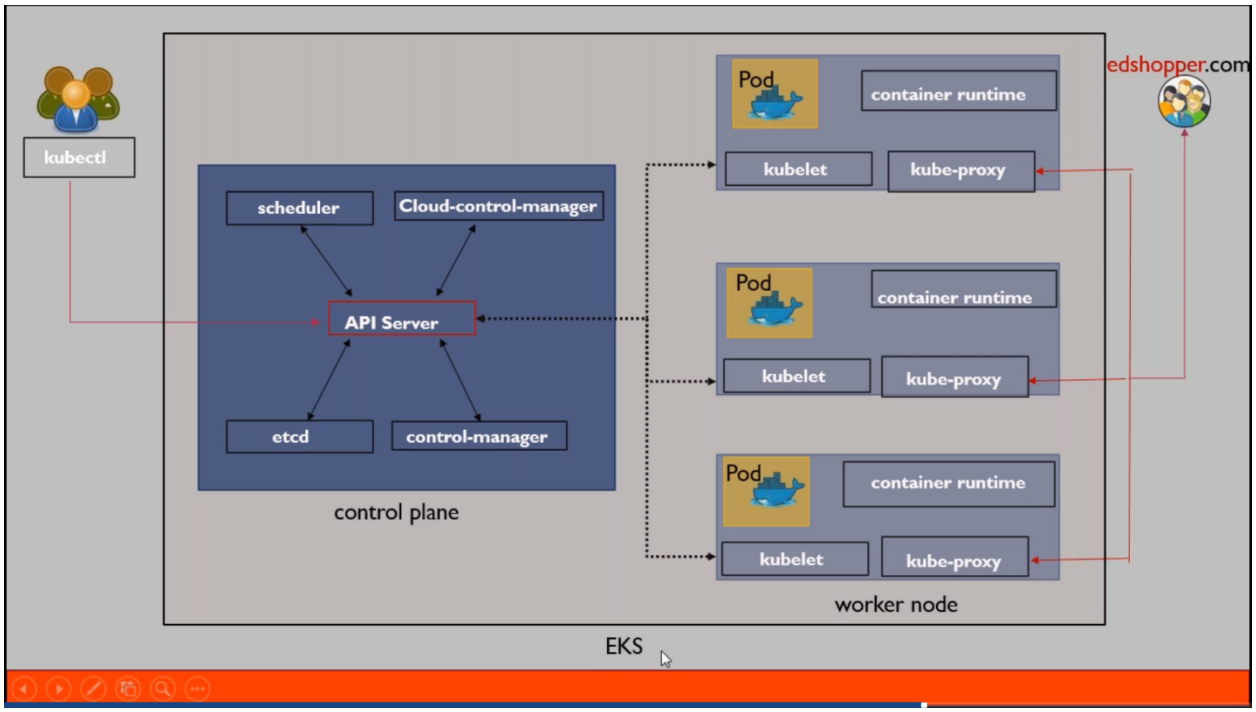
We have lot of advantages of using the EKS [pay as you go, easy application deployments]



If you are interested to take the self hosted refer the documentation



Now will talk about how managed kubernetes clusters helping or making life easier



How EKS people will managing the when worker node is down AWS will take care of high availability of your worker nodes by going through the auto scaling groups

One good thing by using the you no need worry about installing and configuring the services whatever you are seeing in above image

Do not worry about how control plane works in EKS you just need the API server details how you are connect to the API server , use your kubectl command you just inform to API server then it will take care of everything

Means when it comes to EKS Managed Kubernetes cluster you don't even see a control plane server in AWS; total control plane server will take care of AWS

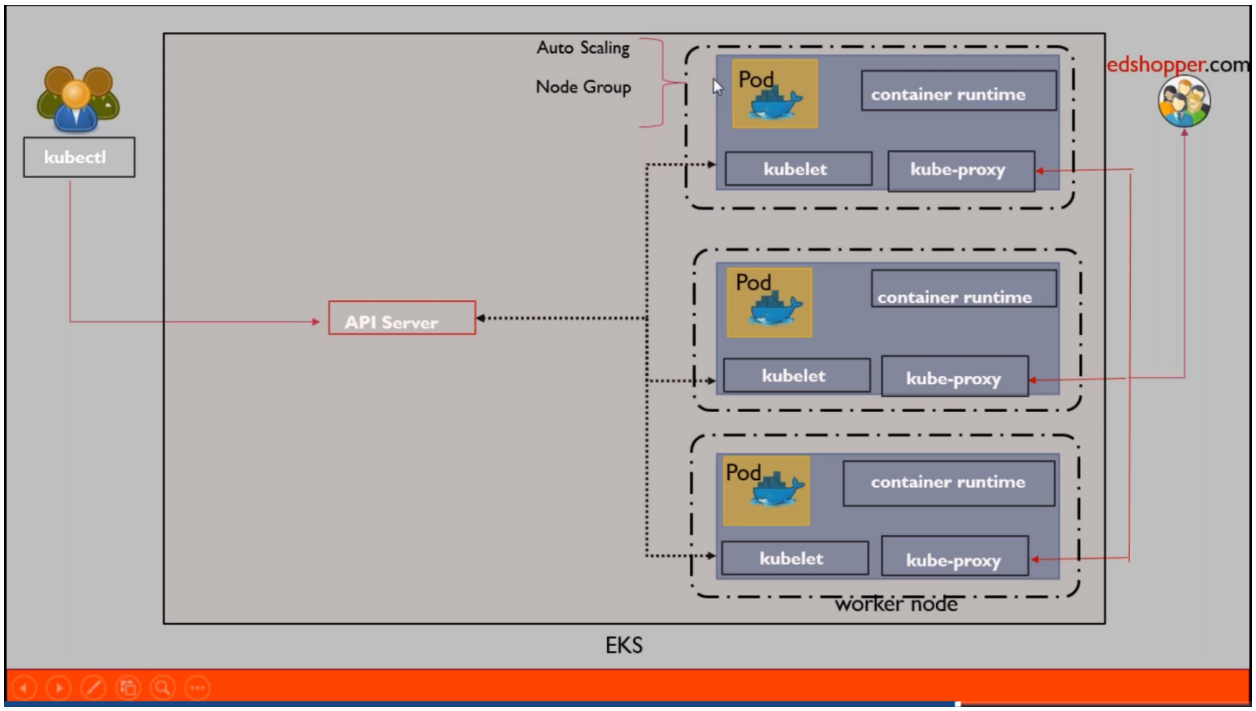
When you set up the EKS cluster you can't see the control node itself just AWS people will tell you this is the API server details just you need to connect to API server and send the details , how the API server doing the deployment how it is connecting to worker nodes everything taking care by AWS only that's why managed kubernetes cluster

If incase control plane down some time should I be worry about it [no, AWS know what need to be done they will take care of it]

You don't have visibility to the Control plane

EKS people will take care of auto scaling and everything

You no need to worry when it comes too managed



Now I may be running a workload in kubernetes cluster workload may be a related to dev environment , work load may related to production environment , may be a database how should we be controlling that when you are using the EKS clusters

Yesterday I listed all the environments in one single server that's not a good practice

Should I use the more computing resource to my dev env or to the production environment?

Production need to have the more computing resources

The problem here am seeing is API server will take care of automating the deployments let say you want to deploy the application for DEV purpose ,for production purpose both are not the same priority production will be the high priority

How will you handle that now?

Let above worker nodes in the below image has the low computing capacity [t2.micro] and below worker nodes has the high computing capacity [m2.micro]

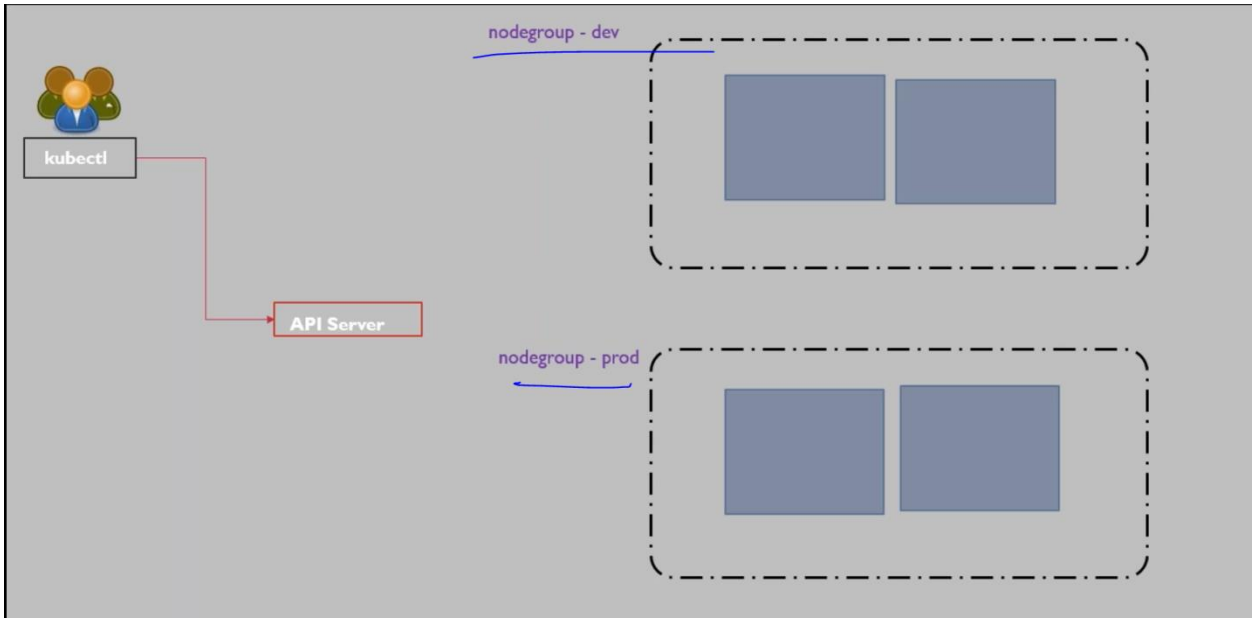
If the application needs to deploy on the production then have to go with below worker nodes

To achieve this in the EKS world we have the concept called nodegroup

Whatever the worker nodes we have we are grouping based on the priority and naming of it

Hope now it is easy to do this

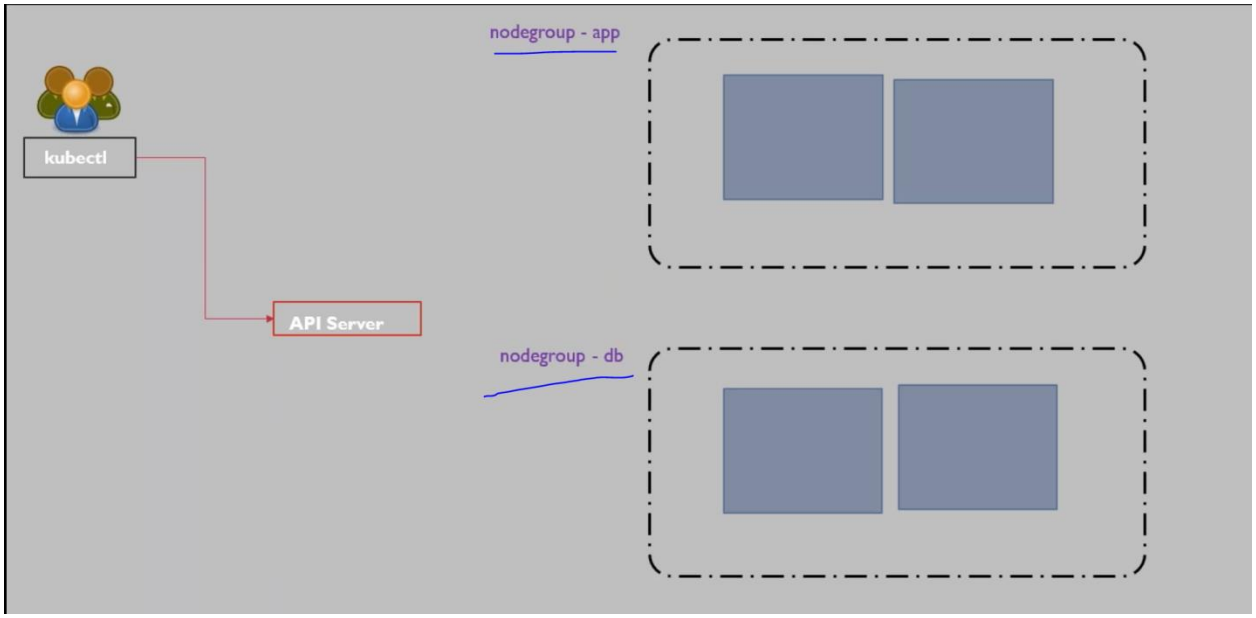
When I was informing to the API server my application this is for dev purpose please be ensure that dev nodegroup, then APIserver [scheduler will ensure that need to go on the dev node group]



This time requirement is all the application need to go app nodegroup and the entire database related things need to go db nodegroup

Node group is just a logical grouping of your servers

You can use the nodegroup as per your requirement [en or db or app]



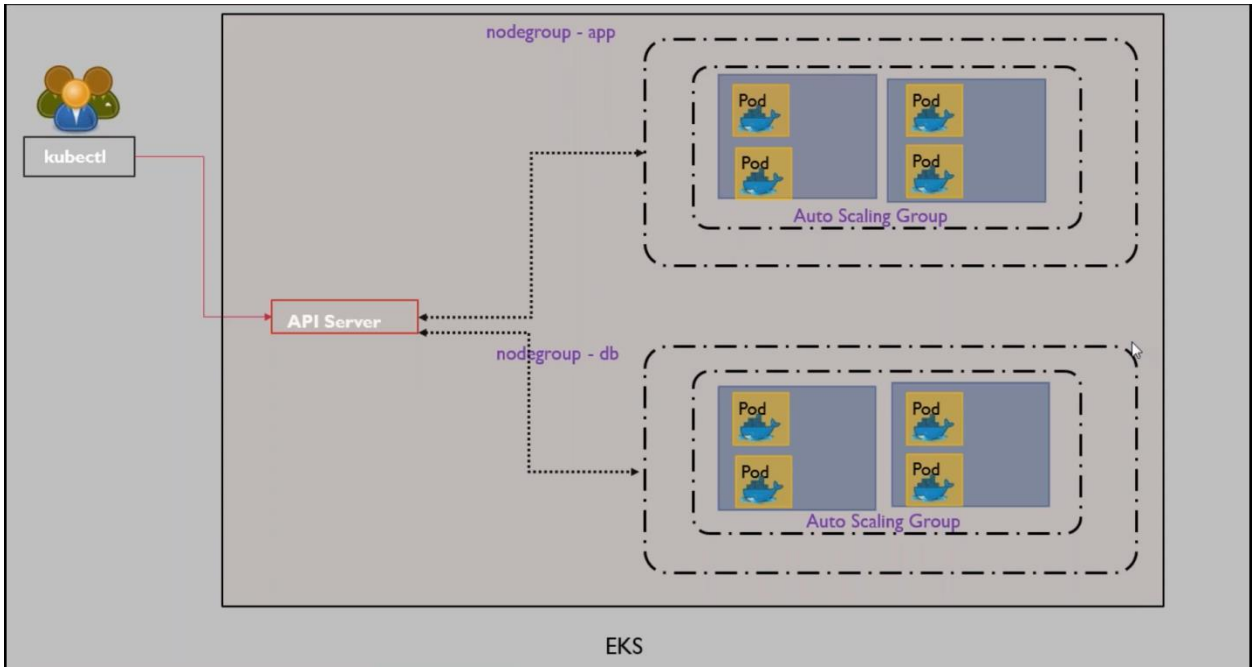
In real-time all front end application will go to the one node group

All back end applications will go to the another node group

All databases will go to the another node group

Like this will divide

Each node group has the auto scaling group; this scaling group will do the all necessary actions



As a summary we are using the orchestration tool for managing the containerized applications get the high availability, auto scaling self healing capability

Kubernetes will work on the top of container D only

You no need take care doing the control node and worker nodes manually taking care by the EKS [Managed cluster]

By looking at the kubernetes architecture we are going to work at the docker level because container intern will take of contained model hence the reason you don't need the deeper and deeper in docker level

In docker level you just need how to write the docker file and create the images because running the application actually in the kubernetes level and just more insight able information about the kubernetes only

If the people are not using the kubernetes still they are in traditional docker level only then only you need to be worry how to create the container and how to manage the container

If you are using the kubernetes you will never be creating a container by using the docker run command, you will be never working as a container level you will be always a pod level

Managed host is little bit less cost when compare to the self hosted