

Applied Accelerated Artificial Intelligence
Prof. Satyadhyan Chickerur
School of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 10

DeepOps: Deep Dive into Kubernetes with deployment of various AI based Services
Session I - Kubernetes Part - 1

(Refer Slide Time: 00:20)

Applied Accelerated Artificial Intelligence

DeepOps: Deep Dive into Kubernetes with deployment of various AI based Services

Session I – Kubernetes Part -1

Satyadhyan Chickerur, PhD
Professor
School of Computer Science and Engineering
KLE Technological University
NVIDIA DLI Ambassador/Instructor

NVIDIA
National Supercomputing Mission
Center for Development and Advancement

(Refer Slide Time: 00:49)

Agenda

- Need for Kubernetes
- Demo –
- Deployments Demo – Load balancing

Hello everyone good evening. So, let us start today's session yes. So, today we are going to actually start with the Session 1 on Deep Dive into Kubernetes which will help us to understand how do we deploy right various AI based services on a cluster. So, the agenda for today would basically be the need for Kubernetes, what is Kubernetes? We will have demos and we will have a demo for load balancing.

So, the idea here is to link this session with the previous session which Bharath took who talked about scheduling and orchestration and he gave a brief overview of what Kubernetes is. In the sense he showed what Kubernetes is, but he discussed about slurm, scheduling and all of this right. So, you now understand the context of why a scheduling mechanism is needed when you use a shared hardware cluster, because there would be so many people trying to use that cluster.

You need to ensure that everybody gets ok some time on that common scheduled hardware that is why you use a scheduler right. So, this is the basic concept. Now, let us start with today's agenda and today's agenda is these 3.

(Refer Slide Time: 02:13)

Need for Kubernetes ?

Virtualized Deployment

App	App	App	App
Bin/ Library	Bin/ Library	Bin/ Library	Bin/ Library
Operating System	Operating System	Operating System	Operating System
Virtual Machine	Virtual Machine	Virtual Machine	Virtual Machine
Hypervisor			
Operating System			
Hardware			

Container Deployment

App	App	App
Bin/ Library	Bin/ Library	Bin/ Library
Container	Container	Container
Container Runtime Kubernetes		
Operating System		
Hardware		

- Containers are a good way to bundle applications.
- Need to manage the containers that run the applications and ensure that there is no downtime.
- if a container goes down, another container needs to start.
- Wouldn't it be easier/useful if this behavior was handled by a system?
- **Kubernetes**

Source: <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>

Let us start with a very very basic premise of what is the need for Kubernetes? We saw how traditionally people used to use the hardware ok and then it basically came up to a virtualized deployment and then it came to a status wherein we were talking of container deployments right and you understand that what a container is by now. And then you also know that since this container deployment needs to be done on a centralized

hardware and there would be people using lot of such containers, there has to be actually a mechanism by which these containers need to be managed ok somehow right.

Whether a person is managing it or there is a automated Kubernetes software which does it or there are certain other scheduling and orchestrating softwares right. But somebody needs to manage it ok. So, if you see that way containers are a good way to bundle the applications, because you have all your libraries your applications the dependencies everything in this container.

As we saw in the previous sessions if this is of a team a this container belongs to team a and this container belongs to team b each might be running different applications with different binaries and different libraries. But it is going to run on basically a common hardware right. So, containers by that mechanism are a good way to bundle applications.

Now you need to manage the containers that run these applications and ensure that there is no downtime. Now the concept is this you have got a very very high performance cluster it has to work 24x7 365 days and it need not have a downtime.

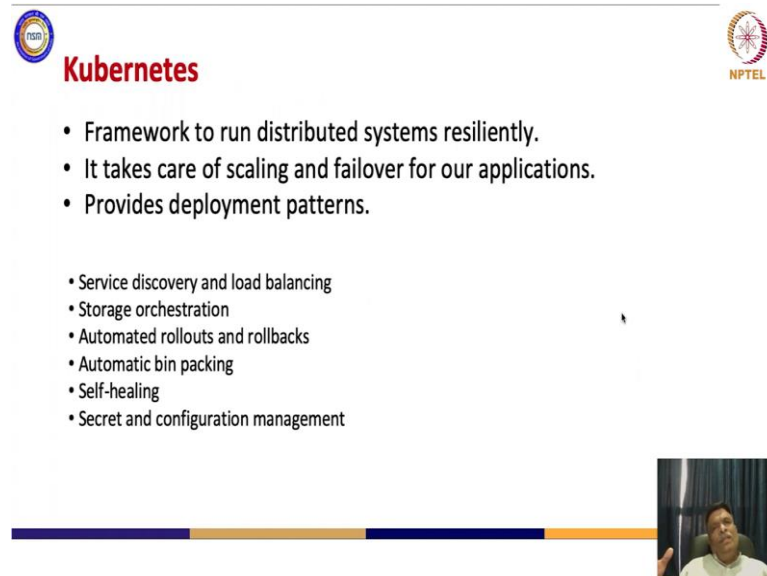
Downtime in the sense when you give some service to the client right or to a researcher he would have put in his container and it would be running for let us say 2 days and it has to finish in another 15 to 20 minutes, but assume the situation wherein some particular thing happens or something crashes and his work of execution of that container for 2 days is lost right.

So, how do you manage such things right such issues. So, this is where you should understand that if a container by chance goes down ok, you have to ensure that similar type of container or that container needs to again restart right. So, the question here what we should be asking ourselves is would not it be very easier and useful correct.

If this behaviour of downtime is handled by some system for us instead of a person monitoring it doing things right; would not it be very helpful and useful if this particular thing ok of ensuring that our applications always run on the hardware ok. Even if the node goes down or something happens we have redundancies into the system.

So, how is that it is managed right if we can ensure that ok that would be very useful and that is where this Kubernetes comes into play. Now Kubernetes is a framework to run distributed system resiliently.

(Refer Slide Time: 06:24)

A slide titled "Kubernetes" with a blue and yellow logo on the left and the NPTEL logo on the right. The slide lists several features of Kubernetes in a bulleted format. At the bottom right, there is a small video inset showing a man speaking. A decorative horizontal bar with blue and yellow segments is positioned above the video inset.

Kubernetes

- Framework to run distributed systems resiliently.
- It takes care of scaling and failover for our applications.
- Provides deployment patterns.

- Service discovery and load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration management

So, we are trying to now come up with a system which ensures right, that this type of a distributed system which has got lot of compute elements GPU CPU memory everything like this is going to run resiliently right. Even though you have got redundant nodes redundant compute stuff and everything like that ok it helps us to run the system ok without any faults.

It takes care of scaling and failure of our applications and it will provide deployment patterns. So, this is a very high level view of what a Kubernetes does. But when you talk of very specific things let us go into a bit of a very specific things which will help us to understand the importance of Kubernetes ok. The first thing is service discovery and load balancing let us try to understand what does it mean ok?

So, this service discovery for that matter is something like my Kubernetes is in a position right to actually understand or expose a container using either a DNS or it is own IP address.

And then it can rediscover all the services available based on the DNS name or the IP address and if the traffic to the container in the sense the container would be using

certain services ok or data or it has to send data or if that is the case if the traffic to the container is high Kubernetes is going to help us balance that load right. So, it will help us in load balancing and it will help us in actually distributing the network traffic. So, that the deployment is stable.

Now, this basically means that from the networks point of view you are trying to load balance your traffic. So, you have got let us say 5 nodes ok and everybody wants to get node 1, because everybody knows node 1 right. So, everybody gives that to node 1 and what effectively happens is node 2 node 3 node 4 are all free, if that thing load balances then your total workload is split across various nodes instead of just trying to run it on node 1. So, that is how the first thing of service discovery and load balancing is going to actually be served by Kubernetes.

The next thing is storage orchestration. So, what does storage orchestration essentially means? See when you try to work with Kubernetes in the days to come you will see that you are going to work with let us say lot of deep learning algorithms right. So, deep learning algorithms when you have a container that container will have all the things, but you will have to have a storage from where right you can pumping your data to that particular type of a model.

You will be working with huge amount of data because you are working with deep learning applications and each of these deep learning applications require lot of data for training right. So, if the data is not sufficient you do data augmentation, you create lot of data on your own and then try to use it for deep learning applications.

So, in such a scenario Kubernetes also allows you to automatically mount right and link the storage system of whatever choice you have let us say you will have local storage, you have storages from the cloud you have got storages from your own laptops or whatever right.

So, these type of systems what you say storages right also could be mounted ok that becomes one of the important features of Kubernetes, that it helps you to orchestrate storage also. Now when you talk of automated rollouts and rollbacks very important thing ensuring consistency ok across caches, across all the nodes all of this needs to be taken care of and Kubernetes does automated rollouts and rollbacks.

So, we can also have let us say a situation wherein we can describe a specific desired state ok for our container using Kubernetes ok, as you go along you will understand how do we do it. But for the time being just understand that there is a desired state right and you have deployed a container, we will see what a deployment means in the demo today, but we have deployed a container and that actually can be described to the Kubernetes.

And this Kubernetes can change the actual state to a desired state ok, in some fixed amount of time or maybe some control mechanism using some methodologies it can actually help us do that attain a desired state. So, we can automate also the Kubernetes to create new containers for our own deployment. So, we will show how do we create a container for deployment ok on a load ok and we can remove automatically existing containers, remove the dead containers, remove certain containers which are having some errors or something like that.

And all those resources which are used by which were used rather by those containers would again be actually given to these new containers. So, that particularly means that you are able to actually get back the resources which were being used by old containers which are not being used ok or which are having certain issues and those resources which are used by them are automatically given back to the system in general.

Now, another important point is automatic bin packing. So, let us say that since we are trying to use Kubernetes on a cluster of nodes right and we know that this cluster of nodes is going to actually run containerized tasks. And we tell now the Kubernetes that how much CPU and memory ok each container is going to need we can allocate that also. So, I tell that ok this particular container this particular user will actually require these many CPU and this is the memory which would be required by his or her container.

So, my Kubernetes is going to fit or allocate my container to those nodes which are actually going to make best use of the resources right. So, we will ensure that resources are used very efficiently. Now another important point is self-healing, so what does it mean right? Self-healing basically means that my Kubernetes installation or my Kubernetes setup is going to restart ok the containers that actually would have failed, we can replace the containers, we can kill the containers based on certain criteria right.

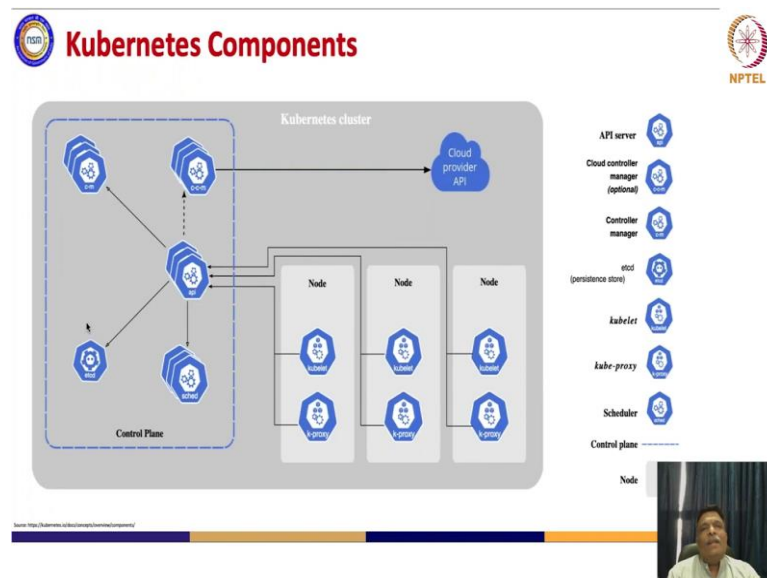
So, we will see what are those criteria's those are called as user defined health check things right and once the containers which is running and if you are providing some

services through those containers to various clients or users. If we define certain health check and ensure that if they are not satisfied then we are not going to advertise them to the other clients until they are ready to serve. So, you indirectly in turn ensure quality of service as well right.

And then the last and the most important is secret and configuration management. So, what do we mean by this? Kubernetes helps us to store and manage sensitive information's, password authentication tokens ok and SSH keys. So, these are all part of configuration management and this is all basically like ensuring these have to ensures security and secrecy for that matter.

And we can actually deploy and update secrets and secured information and application configuration technically, without rebuilding again the image containers and without exposing ok any of the secrets or whatever information ok in basically the stack configuration or something of that size right. So, this is why actually Kubernetes is very very good to be used ok for the applications which we are trying to run on the clusters which we are going to use in future ok.

(Refer Slide Time: 18:18)



So, let us try to understand the components of a Kubernetes right. So, when we deploy actually Kubernetes technically. What are we doing? We will get a cluster, so we assume that we have a cluster and we want to deploy Kubernetes on it. And you know the what

to say the setup of a cluster right like here we have this node we have node a node b node c whatever and then we have compute nodes we have got login nodes.

There are various types of nodes available ok and when I say I deploy in a cluster when we deploy in a cluster what effectively happens is we have to actually manage all these nodes. So, in this diagram you are seeing that there are 3 nodes here, now these nodes are supposed to be are supposed to be clusters ok. And then these nodes actually if you see are linked to the cloud through a control plane.

Please understand this concept that cloud is a service cloud basically is a service right, somebody is providing us that service and we have application programming interface from that service provider, so that we can use this particular cluster. So, what effectively happens is you are a user at this point, you access through the cloud this particular Kubernetes cluster in such a way that you are in a position to get the facilities which are running on this node.

So, technically speaking Kubernetes cluster will consist of a set of workers machines the terminology we use is worker nodes or worker machines that machine is going to run containerized applications and each cluster if you say a cluster it should have at least 1 worker node. So, assuming that this is my master node ok we will see in detail what this control plane and all these things mean what are all these terminologies.

But for the time being assume this control plane is running on another node or a computing element which can be considered as a master node ok. So, these worker nodes and this master node work in client server type of a mechanism correct. So, this is my master node and these are my worker nodes. So, I access this through these cloud provider API I submit in my job to this particular master it distributes it to these nodes ok or my nodes can get the workload to be executed from these ok and then I give back the result.

So, the worker nodes technically will host the pods that are components of the application workload. The control plane if you see this the control plane; the control plane manages actually the global decision ok this control plane manages the global decision about the cluster. For example, scheduling detecting and ensuring that the master replies to all the cluster events which are happening.

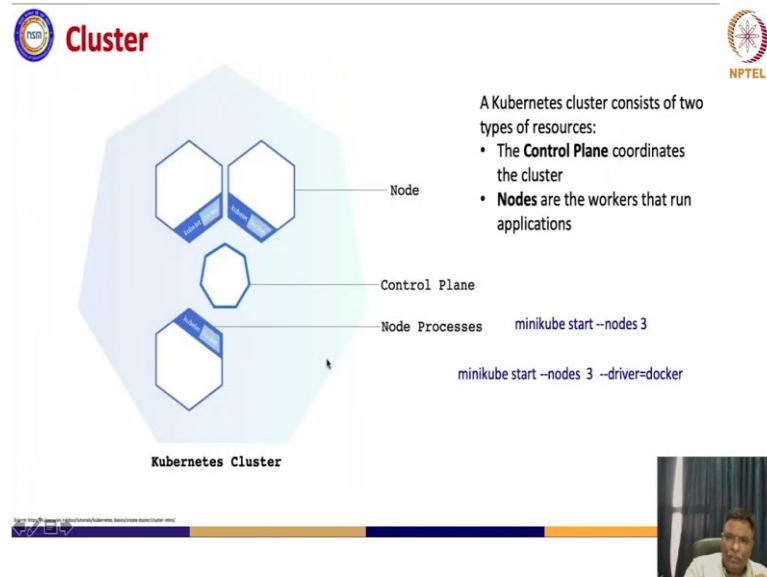
So, control plane components technically ok can be run on we assume right a control to be on and do not run user containers on this machine. So, you we call this as a provisioning node ok or a control node or a master node or whatever right. So, that is why when you actually log in to any cluster you have a login node right and login node basically does all of this and then the compute nodes are these nodes where you actually try to execute your workload on.

So, let us now try to see each of these ok things in a bit of a brief introduction right. You have something called as a kube API server right. So, this is a API server type of thing ok. So, what does it do? It actually is a component which is there in the control plane of course, that exposes the Kubernetes API and this API server actually ok. So, assume it to be a server you get your API from the cloud provider and you try to access it. So, this is a API server for that matter this particular master node. So, this is the front end for my Kubernetes control plane.

The main thing which this particular Kubernetes API server does is it can actually work with nodes ok for managing various parts or Kubernetes ok or something like this. So, this is a point from where you can coordinate ok with the whole of the control plane right and then here we have this etcd right, which is a persistent storage.

So, this is actually a persistent storage or a key value store ok which has got the information about whole of this cluster actually this has got information and this is actually is a backing up store or a backup store ok. So, all of this data which we keep the information about this all of that is stored in this persistent backup storage right.

(Refer Slide Time: 26:29)



And then you have this kube scheduler, this scheduler here ok is actually going to watch for the newly created pods ok with no nodes assigned to them. So, the basic idea is when we were talking of slurm and we were trying to tell that ok that is the scheduler which schedules on these nodes. Here Kubernetes does that work and checks for various pods which are created ok and those pods are supposed to be scheduled by these schedulers sorry this scheduler actually and then it selects a particular node ok any one of them. So, that pod runs on that particular node right.

So, how does it schedule? See I have created a pod. So, pod you can assume it to be a application right we will discuss what is a pod what is a Docker what you know already a Docker now. So, Docker has a application. So that application can be considered as a pod to some extent for the time being please assume that to be a pod ok. Now that particular Docker has to be run on some particular node. So, you give it to you do not know on what node you are going to run it on ok. So, that particular pod ok has to be scheduled.

So, this scheduler takes care and schedules that part ok on to some particular node. Now on what basis it takes a decision right? It takes a decision based on resource requirements, it has to check for the hardware, software and various policy constraints. Now there can be issues of affinity? Affinity basically means that it has to actually go to that particular node itself and anti-affinity specifications you can take care of data

locality in case of scheduling things and inter workload inference interferences or something like that right.

So, all of this are certain parameters which you can set when you set up your Kubernetes and then based on that actually scheduling decision right is taken for these pods. Now then there is a control manager. Now this control plane actually is a component that runs controller processes. So, it has to talk of controller processes and each controller technically speaking is a separate process.

So, you can have a controller which controls the nodes, you can have a controller which controls the job you can have end point controllers, you can have service account and token controllers. You can have various type of controllers right. So, ultimately there is a control manager. So, this control manager is going to either take care of your nodes. So, when you say a control manager is there which takes care of my node, so you call it as a node controller. So, what does a node controller? Do node controllers responsible for noticing and responding when the nodes go down.

So, we will see that example now today when one of the node goes down right how is that node basically is shifted to some the pod which was working or the Docker which was running on that node ok, gets shifted to some other node automatically this we will see in the demo today ok.

So, this is how basically a control manager works. Now then there is a cloud control manager here right. Now this cloud control manager is a component which interfaces right or links to your cloud provider API and it basically is what to say a connection right or it helps you to interact between the cloud platform ok and the various components which are there in the control plane.

Not going into the details of it, but then there are certain issues which are very specific; for example, these controllers right when we say cloud control managers and you are trying to link it up with or you are trying to interface it with your cloud provider APIs right you need to have node controller again you need to have root controller and you need to have service controller.

So, you can actually use this technically for creating and updating right, your cloud provider information load balancing at that point in time for setting up roots in

underlying cloud infrastructure all of this can be taken care by the CCM right and now coming down to these nodes right. So, we have this node component ok and these node components actually run on various nodes. So, we have got lot of node components here right we have got two specifically now shown which is of Kubelet and a k-proxy.

So, now a Kubelet is specifically considered to be an agent; a local agent for that node and that runs on each of these nodes as a what to say as a pod let us say ok. So, it is an agent that runs on each node on the cluster and it makes sure right the Kubelet makes sure that the containers are running in specifically a pod ok.

Now if you say from the name spaces point of view and all I am not going into the details, but the Kubelet takes a set of pod specs ok provided through various mechanisms and that particular container right describes in those pod specs are running correctly or not healthy or not. So, when we talk of pods being run containers being run we talk of in terms of health of the container right, whether they are healthy or not or something like that.

So, but Kubelets generally do not manage containers, which are not created by Kubernetes this is one of the very important things. We should remember Kubelets will not or will not manage containers which were not created by Kubernetes right. So, the idea is this if you have a Kubernetes cluster you should make sure right for achieving efficiency and ensuring that Kubernetes manages all the containers, you should ensure that the no the containers right should be created using Kubernetes ok. And then there is something which is called as k-proxy.

So, this k-proxy basically maintains the network rules on the nodes. So, there has to be some rules right. So, these rules allow network communication between our pods for the network sessions ok or communication inside or outside to our cluster. So, this proxy is going to help us communicate with this control plane are in directly to the outside world through a cloud provider or whatever ok.

So, Kubernetes of course, supports runtime containers which we will discuss maybe in future. But this is how a broad idea of what a Kubernetes cluster actually is right.