**WSD India DoL\_ Azure Learning Series - Introduction to Kubernetes-20240131\_115006-Meeting Recording**

0:03  
So hi everyone, welcome to the Devil session on Kubernetes overview.

0:07  
I'm Arvind Sidoju from UPS SST, UPS team and I'll be walking you through today the bigger picture of Kubernetes.

0:14  
So as part of agenda, I'll be covering what Kubernetes is and how it differentiates from Service Fabric.

0:21  
We'll see like what flaws of Service Fabric eventually let Kubernetes evolution and why people are migrating to Kubernetes.

0:28  
And we look into Kubernetes architecture as well as the containers and parts there are like different components or like entities within Kubernetes environment.

0:40  
So.

0:42  
So before I even start talking about Kubernetes, I want to throw some light on different areas of deployment.

0:48  
So traditional deployment like this, this is where like we used to or every organization ran applications and physical servers.

0:56  
So there was no way to define resource boundaries for applications in physical server and discuss resource allocation issues.

1:03  
For example if multiple applications runs on physical server so there can be instances where one application would take up most of the resources and as a result the other applications were underperform.

1:15  
The solution for this would be to run each application on a different physical server.

1:19  
But this did not scale as resources for under replaced so and it was expensive for organizations to maintain many physical servers as well as the solution virtualization came.

1:31  
So this is the middle era.

1:32  
Like it allows you to run multiple virtual machines on a single physical server.

1:36  
CPU virtualization allows applications to be isolated between VMS and provides a level of security as the information of application cannot be shared across 2 multiple VMS running on the same OS.

1:49  
The solution of application for VMS solve the isolation problem with which with whatever is conflicting the dependencies and share resources problem which we have when multiple applications are running on the same OS but it is but it wasted a lot of resources like CPU and memory like.

2:06  
This is because the VM runs not only your application but also full operating system that needs resources too.

2:13  
So less resources would be available for your application to use.

2:16  
So with the evolvement came the container deployment era.

2:20  
So containers are again very similar to VMS, but they have relaxed isolation properties to share the operating system among the applications.

2:28  
Therefore containers are considered lightweight.

2:30  
Similar to a VM, a container has its own file system share of CPU, memory, access and process space and model.

2:38  
As they are decoupled from the underlying infrastructure, they are portable across cloud and OS.

2:45  
Popular container tech provider is an example Docker.

2:49  
So Docker is a tech provider which is revolutionized the container technology and which is leading the container technology today.

2:59  
Now let's take a look closer look at the container deployment and how, how what what actually are containers?

3:06  
So containers are again just lightweight packages of your application code together with dependencies like programming language, runtimes, libraries required to run your software services.

3:17  
So containers effectively virtualize the host operating system and isolate an application's dependencies from other containers running on the same machine.

3:27  
So how containers solve the problem?

3:30  
So they have a container engine as well as a container image.

3:34  
So the container image is a package of an application and its dependencies and the container engine runs the applications in containers isolating it from other applications running within the same container within the same host operating system.

3:47  
So this removes the need to separate operating system for each application and allowing for a high resource utilization and lower cost.

3:58  
So now with little introduction and like little creep into like what containers is and what what deployment went in.

4:05  
So containers are good, they are good way to bundle and run your applications.

4:10  
Now we all agree looking at different errors of deployment and how efficiently containers run your applications, right?

4:17  
But in a production environment you need to manage the containers that run the applications and ensure that there is no down there.

4:25  
So for example, if a container goes down, another container should needs to start.

4:29  
Would it be?

4:30  
Wouldn't it be easier if this entire behavior is handled by a system which which kind of manages these containers, which kind of manages the deployment, scaling and everything.

4:40  
So Kubernetes comes to rescue.

4:42  
So Kubernetes provides you with a framework to run that distributed systems resiliently.

4:47  
It takes care of scaling and failover for your application and provides deployment patterns and more.

4:54  
So for example, Kubernetes can easily manage a Canary deployment for your application with.

5:01  
With that slight introduction on like what is Kubernetes and why we need it for running containers, let's take a closer look at like what exactly is the definition of Kubernetes and how it stands.

5:15  
So Kubernetes as I said like is a distributed systems platform.

5:20  
It's an open source platform for automation, deployment, scaling and management of containerized applications.

5:27  
So it facilitates both declarative configurations and automation.

5:31  
So it it provides the ability and extensibility for a dev to provide a declarative configuration on what and how the containers to be run on a deployment system and how they should scale up so it's that extensible.

5:45  
It has large and rapidly growing ecosystems.

5:47  
Since it's an open source with huge and active dev community Currently contributing to the Kubernetes in GitHub, I really don't want to go into a brief history on like how contain how Kubernetes evolved over the time.

6:02  
But as a need for info like Google open sourced Kubernetes so it's a in house product from Google, the open source in 2014 and Kubernetes comments over like somewhere around 15 years of Google's IT workloads, its scale and IT with the best of breed ideas and practices from the community.

6:24  
So it's been like evolving and like it's been continuously getting contributed from throughout like from 2014 to right now.

6:33  
So I'll, I'll stop here to see if there are any questions.

6:37  
If not, I'll I'll just proceed and talk further more about Kubernetes.

6:45  
I take that as no.

6:47  
So yeah, now coming to the future features, Kubernetes provides a ton of modern features that you expect from a distributed systems platform.

6:56  
So it it it manages the service discovery and load balancing for you, It orchestrates the storage, it automates the roll, insurance and rollouts and rollbacks for you so you don't have to really care about how container application gets scaled out and rolled out during the deployment.

7:15  
So Kubernetes will help you with that.

7:16  
It it provides a self healing capabilities.

7:18  
So whenever a container or a part dies, Kubernetes spins up the new ones and make sure that the self healing happens automatically in the background without any like disturbance to the devs or or application print end point of view.

7:32  
So it supports the batch execution properties, it it supports horizontal scaling and it it it it can go as expected to the dev who provides the declarative configuration to it.

7:42  
And it is designed for extensibility.

7:44  
So you can write your own extensible code on top of Kubernetes main source code and you can still use that if you want to have any particular features that you want to implement within Kubernetes and it also provides automatic bin packaging it.

7:59  
It also provides service secret and configuration management.

8:02  
And these are like again the main features that you would expect from a deployment and distributed system to provide.

8:08  
And there are like again ton of more features but I just highlighted like a high level like what all it provides and why it is like really needed for us.

8:19  
So it it provides a way to.

8:24  
Oh, it provides a way in which you can provide the bin of a container and the Kubernetes takes care of using that the packages that you provide.

8:34  
And it it it takes care of deploying it and distributing it on the cluster.

8:38  
When I say cluster, it's a mission so you don't really need to care about how the bins are getting created.

8:44  
All you need to provide is a configuration on hey this is my runtime and this is my project and this is the CS project and so and so and you can and Kubernetes will take care of like packaging the binaries and then creating the runtime instances of it and like distributing it.

9:01  
So yeah those are like very high level features of Kubernetes of the definition and features that I have described so far.

9:09  
So you might be already thinking about something else like we already have a platform, are you already if you are a someone who is developing micro services in Microsoft, you would already have some experience with Service Fabric like which is also again a distributed system platform which runs microservice applications and takes care of scaling, deployments and pretty much everything.

9:34  
So Service Fabric as I just mentioned again is an another alternative distributed system offering to run our application.

9:41  
And again a slide brief history of Service Fabric is is distributed systems platform that makes it easy to package, deploy and manage scalable and reliable microservices like it.

9:53  
It pretty much like stands for the same definition but the features and the way in which it handles and how it defines its microservice is entirely different which we'll talk about in the next slide.

10:05  
Before that like just want to throw some light on like how extensively used to Service Fabric in Microsoft, Microsoft.

10:13  
So pretty much all of our internal teams within Microsoft use Service Fabric there is no doubt about that and and we use to develop the cloud native micro service application.

10:25  
So pretty much anyone who is building a distributed scalable micro services and Microsoft is currently using service.

10:34  
So I just want to, I'm not going to pinpoint 1 to one really differences between Kubernetes and Sales Fabric.

10:43  
Although they are, they both are distributed systems platform but they are not fully comparable like 1 to one in terms of how they function.

10:51  
So I'll try to highlight a few reasons why most people at those organizations are transitioning from Service Fabric to Kubernetes and maybe few issues that we have in our good old in house Service Fabric that we have developed in Microsoft itself.

11:08  
So the main problem with targeting applications for Service Fabric is that they will lack portabilities.

11:15  
So Service Fabric SDK is incredibly opinionated uh and when I mean opinionated like if you come into Service Fabric you will be tied to a specific SDK and application server Uh I don't know how many of you are uh running on applications using Service Fabric but uh you have to stick with a particular SDK and like runtime uh when you are developing an applications and uh and even if you look you know Microsoft docs to find write run times and SDKS you'll you'll find like very few which has which has not evolved yet.

11:49  
Like there are only 1-2 versions of service public SDK and runtime where you have to stick with that in order to develop applications.

11:57  
So the second Service Fabric isn't directly comparable to a container of this data such as Kubernetes like as it is more of an application server with a specific style of distributed system.

12:08  
So in this in this particular point when I say this it's it's not an entirely contained orchestrator Service Fabric because when you create applications and Service fabric you use Service Fabric SDK within microservices that you developed.

12:23  
So you learn and and in fact if you already used like when you create a stateless as stateful applications in Service Fabric, you will be inheriting from the base classes of stateless service and stateful service which are again within SDK of Service Fabric.

12:38  
So you you tightly couple your applications with Service Fabric SDK and you pretty much view your ownership to service fabric to handle the runtime of your applications when you are deploying the the the services that you built.

12:52  
And I think that the last point which I mentioned is again related to the same.

12:57  
Like native Service fabric services are based on very specific style of implementation.

13:02  
So Service Fabric provides like different types of service patterns which you could use could be reliable actors, containers and native executables like.

13:13  
You have to stick with those patterns to create microservices and they are not really a contain container of a container but they are a microservice pattern which which which service Fabric handles on its own and it takes care of deploying and providing the runtime when it gets deployed.

13:33  
So there are a few more.

13:34  
Again I just want to highlight a few more things.

13:37  
Service public competitive.

13:38  
Now this public also provides container support.

13:45  
There's a there's a way in which you can create native containers which I just described earlier like what exactly is a container.

13:52  
But when you when you when you make the container based services like they are not really containers in service Public, you still like blend into the service public SDK and you often rely on service public to handle the execution of your services and and it's again dependent on Service Fabric.

14:13  
The other problem with Service Fabric is not a pure container orchestrator as I just mentioned, like you, you you blend the microservices that you developed with Service Fabric.

14:23  
So it's not an orchestrator by definition, but it does orchestrator capabilities where it handles the micro services that you build on top of Service Fabric SDP.

14:35  
The other problem is the resource governance.

14:39  
So resource handling is much more difficult when noisy neighbors.

14:45  
When I say noisy neighbors, it's another applications or services running on the Service Fabric cluster itself.

14:51  
When they starve other services of resources, when they eat up all the resources, you couldn't really handle too much or you couldn't really control too much when you deploy applications under Service Fabric.

15:03  
So you can assign CPU and memory to services, but this is an inflexible setting that reserves the capacity and this makes it difficult to plan for unpredictable load and resource utilization across the entire cluster.

15:19  
So, and as I said like these problems will be like handled in a much more efficient way in Kubernetes.

15:27  
Like Kubernetes provides a more flexible system of request and limits, so you can define a baseline of resource consumption while specifying what resource is allowed to do when it grows.

15:39  
So yeah, having said the some class in Kubernetes, sorry some class in Service Fabric and like how these are like effectively and efficiently handled in Kubernetes.

15:49  
I'm not saying again Service Fabric is really bad, we must move to right away.

15:54  
But if you look at developer E and so if you if you just look at in developer perspective who just builds an application, Service Fabric setup is 1.

16:05  
Let's say one service Fabric setup is done successfully.

16:07  
Like you don't really need to care about anything in service Fabric.

16:09  
Like you just create a microservice under Service Fabric application and done like Service Fabric you let service public handle everything for you like it's a it's like a boon for dev right.

16:19  
Like we don't really care about anything but as it evolve as an organization evolves and as as we develop more sophisticated and distributed systems so you really care about the infrastructure, resource consumption, effective utilization and how the rollouts and roll backs are being handled.

16:37  
And if you want to use the long term support and the the new technology changes that are coming.

16:44  
So Kubernetes is answerful, so I'll stop here again to see if there are any questions before I go into the architecture side of Kubernetes.

17:05  
Cool, I think don't have any.

17:08  
So let's have a look into Kubernetes architecture.

17:11  
So on a high level, Kubernetes architecture components include the Kubernetes control plane and the worker nodes in the cluster.

17:19  
So the main control plane machine components include Kubernetes API server, Kubernetes scheduler, Kubernetes Controller manager and EDCD which is a Kubernetes cluster store.

17:32  
And on the worker control plane or the worker sorry on the worker plane or worker mode we include container runtime, 2 blade 2 proxy and POC.

17:41  
We will talk about in a minute on like how these internal components are like handling the Kubernetes in overall and like why they are needed.

17:52  
So for a minute like just think about this part like I'll I'll cover this in the upcoming slides but so part is an entity under which a container runs container application runs.

18:06  
We'll just remember that for a minute and we will talk like what exactly is part and by Kubernetes needs that.

18:11  
So.

18:12  
So let me just jump into the control plane.

18:16  
As I mentioned like this is the left side part in Kubernetes and which is the brains of Kubernetes if you call it explicitly.

18:24  
So it it pretty much controls what a worker nodes needs to run like what how worker nodes gets managed.

18:34  
So every time you run a service under Kubernetes, pretty much all the applications runs on the worker nodes, but the control plane is the one which is handling it.

18:45  
And it is pretty much handling all the features that we have mentioned as part of Kubernetes that will be done in the control plane of Kubernetes.

18:53  
So it as I mentioned it typically contains these four components.

18:57  
It has controller, manager, API, Server, ATC, DS, again Kubernetes cluster, store, store and the scheduler is 1 which assigns the task.

19:06  
We'll talk about it in a minute, but these are the four main components we have in the Kubernetes control plane.

19:11  
What does it store in the Kubernetes cluster store so it maintains I'll, I'll talk in a minute but yeah basically it maintains the state of individual worker nodes, individual container applications that are running under Kubernetes.

19:25  
So it it's basically a database a stateful database which we'll need in in Kubernetes to maintain the state of individual applications.

19:33  
We'll talk about in a minute by the state is really needed.

19:37  
So let me just jump into individual components in the control plane.

19:42  
So as I mentioned, like Kubernetes is a control plane is the brains of Kubernetes which runs the worker nodes.

19:48  
And let's talk about the first component.

19:50  
Actually the API server is the front end of control plane and it it provides and supports updates, scaling and other kinds of life cycle orchestrator features that we have just mentioned in the functionalities of Kubernetes.

20:05  
So clients must be able to access the API server from outside the cluster because.

20:12  
It serves as a gateway supporting life cycle, orchestrator, orchestration at each stage.

20:18  
So clients use the APA server as a tunnel to talk to parts, services and nodes and authenticate via APA server.

20:27  
So in a way like you talk to Kubernetes using APA server, so it exposes a bunch of REST APIs and you use Kubernetes, you call or you hit Kubernetes APA server to pass in any commands from outside Kubernetes.

20:42  
And more importantly all the internal components within Kubernetes may be within control plane or may be the the worker nodes and the components within worker nodes, they all communicate using the API server REST endpoints itself.

20:56  
So that's that's the importance of API server.

21:02  
So then we have a cluster store as Akash just asked.

21:06  
Like cluster store or ETC Is a persistent component on the entire control plane and it is where actually the configurations and state of the cluster are kept and that includes the configurations and state of all the applications which are running under the Kubernetes cluster.

21:23  
So right now it is currently based on an open source keyword based configuration named etc.

21:29  
Which is a Nosql database and it automatically distributes all the, etc components across the control plane nodes.

21:40  
So etcd acts as a single source of truth for all the Kubernetes cluster components which are responding to the queries from the control plane and it retrieves various parameters and it it uses various parameters to retrieve the data which it stores in the, in the etc.

21:57  
So etcd's again like used to store configuration details in a in a very general way.

22:05  
So it it stores conflict maps, subnets, it stores secrets along with the cluster state data.

22:12  
So, so in a way like it's a it's a database within Kubernetes control plane.

22:17  
In fact like it's a database for the entire Kubernetes cluster.

22:22  
So we have controller manager.

22:24  
So before I talk about controller manager, like we have multiple controllers within Kubernetes ecosystem that drives the data of different endpoints, maybe parts or maybe services and it drives the states of token, service, accounts, notes and replication.

22:41  
And pretty much for every action that you perform on Kubernetes, you have a specific controller in UMM in in Kubernetes which handles that.

22:49  
Now controller manager is like uh is a or whatever.

22:54  
We call this simply a simple controller is a daemon service which runs on the Kubernetes control plane and it it provides all these controller functions which pretty much handle different operations within Kubernetes control plane.

23:09  
So it the way it performs is like it executes watch loops which continuously monitor the state of different operations which are performing like let's say the node controller.

23:22  
It maintains a watch loop to look at the the nodes health like it it it looks at the desired state of the node and the expected state of the node and depending upon the situation it tries to execute and communicate with the API server to bring the node again back to the desired state.

23:38  
So it it uses the watch loops and it it basically pulls for every given interval of time and eventually brings the state of different components they are looking in and they bring them to back to the existing personal state.

23:52  
So that's what controller manager does in the control plane of Kubernetes.

23:57  
And then we have a scheduler, so when API server receives a request from for scheduling the parts like for running the containers or for running the applications within containers.

24:07  
So it's a scheduler task which actually comes into action which assigns the tasks to the node workers to run the parts.

24:16  
So it it actually takes care of intelligently handling which node they want to run a particular application at which node where they want to replicate the application.

24:26  
So scheduler task is the one which assigns tasks the worker nodes.

24:30  
So those are like 4 different components within control plane of Kubernetes which which basically handles pretty much all the operations within Kubernetes.

24:41  
So they are like the brains of Kubernetes.

24:44  
So any questions here, How does controller monitors the health of containers?

24:59  
So sorry, come again, I mentioned like the controller, yeah, that that we'll talk about in the the worker nodes, how worker nodes communicate back to controller and how they maintain their health.

25:10  
So I'll, I'll talk about that in a minute.

25:13  
So I'll, I'll jump back to the worker plane.

25:15  
So again, the worker plane has like a bunch of components.

25:18  
You have a cool IT which is a Kubernetes service or an agent which actually takes commands and like reports help back to the Kubernetes cluster or the Kubernetes control plane.

25:31  
You have a container runtime which is really needed to run the containers which are there on the local nodes.

25:37  
You need a coup proxy which is used for routing and load distribution.

25:42  
And you have parts which are actually eventually the ones which are running the containers.

25:47  
So let's let's look into that.

25:49  
Again, individual components within within worker plane.

25:54  
So cooplet is nothing but an agent that communicates with the control plane.

25:58  
It ensures that the containers in the pod are running and it provides the status back to the control plane.

26:03  
So when a control plane requires a specific action to be happened on the worker node, it passes those commands to the cooplet itself.

26:12  
So it's it's more of like a worker agent for the control plane.

26:15  
So anything that Kubernetes control plane wants to perform on the Kubernetes worker node, it communicates via coblet.

26:23  
And even the coblet service is the one which reports back the part and everything back to the control plane.

26:29  
That's how control plane knows the health status and everything.

26:32  
It does not go through API server, it it goes through API server.

26:35  
But Coblet is the one which calls to the API server, right?

26:38  
This is the service.

26:40  
Makes sense, yeah.

26:45  
And then we have containers runtime, yeah.

26:47  
So when we talked about Kubernetes, Kubernetes is an orchestrator which which orchestrates the applications, but it's not the one which actually is running the containers which which doesn't really know how a container could run.

27:01  
So we we need to have a runtime which runs the containers, and every node must contain a container runtime which which runs the containers which are running under the parts within worker nodes.

27:13  
So this is one of the main component again in in in Kubernetes worker node and Kubernetes supports all the.

27:22  
We have a different file, sorry we have an open container initiative so that's nothing.

27:29  
But like they have some complaints on like what kind of container runtimes that are supported in Kubernetes.

27:34  
So anything any container it might not be docker like you can use some other container take as well which supports and provides that complaints on like how container runtime should be.

27:45  
So you can you can run the different container runtimes as well in Kubernetes worker nodes.

27:51  
So they're basically the the runtimes which we need to run the containers which eventually runs the applications and we container as such like running the coding the application or something else.

28:03  
Yeah, so you blend an image or or an application within container, right.

28:10  
But you need to run the container eventually to run the application within.

28:13  
Yes.

28:13  
So you need a runtime, right.

28:16  
So that runtime is the container runtime.

28:18  
So Kubernetes uses container runtime to run the container.

28:23  
So within what will mean runtime like for example memory allocation and all those things or no, no.

28:28  
So runtime is basically the run which runs the application, right.

28:32  
Like you need dark net runtime to run a dark net, yes, application build by SDK.

28:36  
So dark net runtime is a runtime which you need forrunning.net applications built with that NET SDK, right?

28:43  
So container runtime is the one which you need to run containers which are created using containers within containers.

28:50  
Like when you create a darker image or a darker container, you need darker runtime to run that container.

28:56  
Yeah it's more of it's not it's not an environment but something which executes your binaries like you need runtime to execute the binaries and and this runtime is different from like.net runtime because like yeah it's it's a container front times yeah which which will be again developed by the the the container provides providers like if you are using docker images, docker provides a docker runtime to run the images.

29:23  
It's it's again like a scope to the one which which the type of e-mail that you are creating.

29:29  
Then we have coup proxy.

29:32  
Each compute node contains a network proxy called coup proxy that facilitates Kubernetes networking services.

29:38  
So KU Proxy runs on each node to ensure that services are available to external parties and deal with individual host subnetting.

29:47  
It's also the network proxy and service load balancer on its node and it it manages the network routing for different parts of different applications which are running on the parts.

30:00  
So that's what Kuuproxy does.

30:02  
So it's basically a an agent for routing and load balancing.

30:09  
So before that that wraps the previous slide wraps up different components that we have under Kubernetes which are under green control plane and worker plane.

30:21  
And we just talked about the subcomponents that we have under the control plane and worker plane.

30:27  
So let's take a look at BOD.

30:29  
I just mentioned something back that BOD is the one which under which you need to run a container in Kubernetes.

30:36  
So if you if you look at the unit of deployment for different technologies like like for a VMware its unit of deployment is virtual machine for a VMware tech, for docker tech or for a container tech you the unit of deployment is containers.

30:54  
You you you communicate or you execute things at container level.

30:57  
At Docker, tech cannot control things with internal to the container, so it it isolates things.

31:03  
It bundles things within controller.

31:05  
But a tech which runs the containers controls only at control level.

31:10  
So similarly Kubernetes the unit of deployment is part.

31:13  
So Kubernetes controls entities in in in terms of parts.

31:17  
And within parts you run your containers and everything.

31:19  
But Kubernetes is the one which controls the parts.

31:22  
So S Kubernetes orchestrates and manages the containers, but every container must be within part.

31:30  
You need to encapsulate every container within part when you are like deploying the Kubernetes.

31:35  
So with that.

31:37  
So the part is again you just code unit of management in Kubernetes so it acts as a logical boundary for containers and which shares the shared resources and context.

31:48  
As I mentioned again, your part represents a single instance of an application and the simplest unit with which the Kubernetes communicates.

31:55  
So it Kubernetes level it's only parts.

31:57  
Kubernetes really doesn't go inside pod and talk to containers, but Kubernetes talks to parts.

32:02  
While orchestrating.

32:04  
Each part is again composed of container.

32:07  
If you look at in the diagram so tightly coupled containers, you can blend again multiple containers within pod along with the rules that control how the containers run.

32:16  
So in a very part is more of encapsulation of what you are running, what containers you are running within Kubernetes.

32:23  
So with one part have fall back mechanism on another part, No, sorry.

32:30  
Could you in case of any fall back requirement, if something fails, definitely yeah that's that's what Kubernetes does.

32:35  
So at Kubernetes level the unit of execution is part, so it takes care of like checking the health of part, monitoring the parts.

32:43  
If something is not working as expected, it kills the parts.

32:47  
It spins up the new parts.

32:48  
So every time a part comes up, spins up new part or it falls back to another part.

32:52  
No, no it it again maintains the state.

32:54  
Let's say you desired like 5 replicas of your application and it spins up five parts for you.

33:00  
And let's say something goes down, it again spins up a new part, so it again maintains the state of five parts by application instances running in the cluster.

33:11  
So yeah, that's covers the basic definition of part.

33:16  
And yeah, you must encapsulate a container within a part.

33:19  
Uh, you could definitely run multiple containers within a part, but that's not recommended because obviously you want to have more control of your application, uh, and you don't want to have multiple applications again within a part.

33:31  
So that again messes up your main like, umm, aim of having a sophisticated distributed system.

33:39  
So the recommendation is to have a create a container within a part per container.

33:45  
So that's how you do and I just want to again in this last 1-2 slides talk about deployments.

33:54  
So Kubernetes deployment tells Kubernetes how to create a modified instances that hold the containerized applications.

34:05  
So it's a format the way it's a script which is very specified how you want your part to be, how many replicas you want it to be.

34:13  
It's more of an input yaml file script that you send to Kubernetes input describing how you want your replicas, containerized application and resource utilization.

34:24  
Pretty much the declarative configuration that I just mentioned.

34:27  
You provide it through a deployment so Deployment holds the structure and how you want the parts and containers to be within Kubernetes.

34:35  
So typical deployment file looks something like this where so this is again the basic Yammer file which you need in order to run or orchestrate orchestrate your services or containerized application within Kubernetes.

34:50  
So you start with the deployment.

34:52  
It has a specific format but pretty much like if you look at it like it has spec it has replicas.

34:57  
Replicas again talks about the number of parts that you want and if you come little bit down you have a spec where you could define container the containers that you want to run under the part.

35:08  
And you can also provide the port which you want to open up on the particular container.

35:13  
So it has again bunch of other features that you can provide like you can also specify for each container and for each part how much resources like how much memory you want, how much CPU utilization you want, how many cores you want.

35:26  
So at that level you could define things in Kubernetes deployment DML file.

35:32  
So that's about deployment DML file.

35:35  
Yeah, this is the last slide for this session.

35:38  
So I talked a lot about Kubernetes, right?

35:40  
Like I talked about how Kubernetes efficiently handles the infrastructure and how it brings the middle plane which which provides the capabilities and how we can control the resources.

35:54  
But the cloud providers like when when we have Azure like Google and Amazon which are providing like different offerings like when you want to run your own Kubernetes, you only consider like pulling the machines like the nodes and you build your Kubernetes on top of that and you take care of like assigning multiple nodes to your control plane and assigning multiple nodes to your worker plane and then you manage that.

36:19  
So that's one way of dealing it and what the cloud providers supported is so they came up with a wrapper within within the Kubernetes.

36:29  
So what it is like they said, hey we we control the we take care of the control plane which manages and efficiently handles the Kubernetes brains.

36:38  
We make sure that it's highly available and everything just take care about only the worker side and provide us what how many replicas and how many senses and what utilization and what memory you need.

36:49  
So that's where like these Akas, Google Kubernetes Engine or Amazon Elastic Kubernetes Search came into picture where they provided an offering where they call this is again a platform as service offering where cloud providers came up with a platform offering telling that you we are already giving the case to you, you pay us for this and you just depend on deployment.

37:12  
Yamal file will take care of like the control plane management and effectively handling the parts and everything on that.

37:20  
So that's how like ASK and GK and EKS were built where they handle the control plane of Kubernetes and let's the devs removes the headache of devs taking care of maintaining the control plane.

37:33  
Because control plane is like a grain of Kubernetes which handles everything.

37:36  
So you have to make sure control plane has more scalability, high availability and reliability and the cloud providers provided that as an offering and they they build the a service platform as a service offering on top of that.

37:50  
So that's what we got a case and right now like when we want to use Kubernetes, we we don't directly use the Kubernetes and build on infrastructure on Prem, on the on Prem notes.

38:01  
But we typically go to one of these cloud provider offerings for Kubernetes and we let them like run our parts or contain those applications within them.

38:11  
So that's about Kubernetes as tasks offering from multiple cloud providers.

38:17  
So with that I'll conclude.

38:19  
I think I pretty much covered like what Kubernetes is, how it differentiates from Service Fabric internal architecture of Kubernetes, how control plane node and worker nodes work together to ensure effective orchestration and how cloud providers are providing a platform as a service offering for Kubernetes which you could use.

38:41  
I am open to any questions.

38:43  
If you have one question.

38:46  
So you talked about like of self healing and automated scaling and all that.

38:51  
So where is that logic like deployed or written?

38:54  
So the controller manager and the coop let which we talked about right.

38:58  
Controller manager uses the watch loops on each worker nodes and coop.

39:02  
Let's try sending the part health back to controller managers.

39:06  
So the controller manager in control plane takes care of again spinning up the new parts depending upon the help and everything.

39:13  
So how do we feed that logic?

39:15  
Using AIR?

39:16  
No, you don't need to feed that logic, Kubernetes already has that English logic, you just need to provide the desired state.

39:21  
Like let's say you want 3 replicas of your application running and let's say due to some issue like something went wrong and some part right, Kubernetes internally takes care.

39:30  
Hey they've asked me 3 replicas, right now there are only two.

39:35  
So it intelligently handles like spinning up a new part on the depending upon the health, current health and like current a current state of replicas and the expected state of replicas.

39:46  
So using that it's been such a new part.

39:48  
So this is about healing, about scaling and I'll suppose I'm getting more request on my service so that you could define on the the yaml file that you yaml file.

39:57  
Yeah, the logic correct.

39:59  
It's it's highly declarative and highly extensible where you could like explicitly mention everything in the yaml file and you could command Kubernetes to perform the a particular operation.

40:13  
Any other questions?

40:13  
It was very informative.

40:19  
This was good.

40:19  
Thanks.

40:24  
Folks online, do we have any questions and Ask is a fully managed service within Azure.

40:35  
So use ask for any Kubernetes clusters, especially in Microsoft.

40:47  
I'll start.

40:49  
Thanks.

40:50  
Thanks, Arvind.

40:51  
Thanks, Arvind.

40:51  
Very helpful information.

40:59  
We'll start the.

41:00  
Yeah.

41:00  
OK, We'll close the session then.

41:02  
Thank you everyone for joining me.

41:04  
Thank you.

41:04  
Thanks Arvind.

41:06  
Thank you.

41:06  
Thank you.

41:06  
Bye, bye, bye.

41:07  
Yeah.

41:08  
Thank you.