**1. What is kops?**

it is way to set up k8 cluster instead of using kubeadmin. It is an automation tool. It will not help to set up managed k8 cluster. We can spin up a test cluster or small dev cluster quickly, test your application, deploy your service or do any kind of load balancing. It not only create the cluster but also helps in destroy, maintain and upgrade the cluster

Kubdeadm will not help in integrating will all the external services like Ec2, ASG, route 53, EBS. Kops will help in integrating all these external services.

Kops is an automation tool using which I can set up k8 cluster in a cloud account. It helps to automate the whole setup process, destroy the cluster or upgrade it or maintain it. If I will be given change to use k8ops then I will use it to set up a quick development or testing cluster in my company.

If we want to do certain kind of testing like load testing, application testing we can quickly spin a cluster, test it and destroy it.

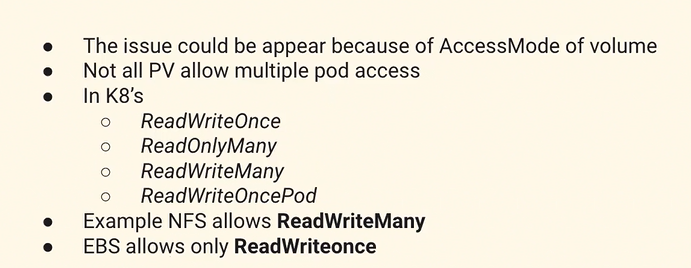
**2. Explain replication controller in k8?**

It is responsible for maintain the number of pods across the cluster**.** It is responsible for maintain x number of replicas in the cluster at any given point of time.

**3. What is pv and pvc? what role do they play?**

PVs and PVCs work together to provide persistent storage to applications running in Kubernetes Pods. The PV is the actual storage resource, while the PVC is a request made by a user or application for a certain amount and type of storage. When a PVC is created, Kubernetes finds an appropriate PV (if available) and binds them together. If no matching PV is available, Kubernetes can dynamically provision one based on the StorageClass specified in the PVC. So, PVCs help users to claim the storage they need, and PVs provide the actual storage resources to fulfill those claims.

****



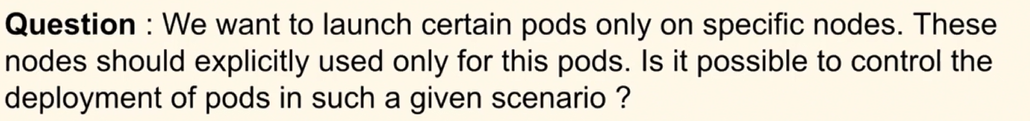
The error could be appearing because we could be trying to attach a particular volume which has only ReadWriteOnce access mode. The solution for this is to make changes in PV and PVC so that a separate volume is created for this or we make sure that the volume which we are trying to attach supports ReadWriteMany permissions.

5. What is a sidecar container?

These are the containers that helps the application running in the pod to perform the some common task. The most common example would be logging. we can have a side car container which has a purpose to send the logs from the appl container to centralized logging system. This mechanism will help us to make sure that a common use cases across ur application can be separated and a side car container will help us to make sure that our deployment is smooth and it will also remove certain complexities when we build the application.

In Kubernetes, a sidecar container is an additional container that runs alongside your main application container in the same pod. It provides extra functionality to support or enhance the main application. A sidecar container can handle tasks like logging, monitoring, or networking for your main application.

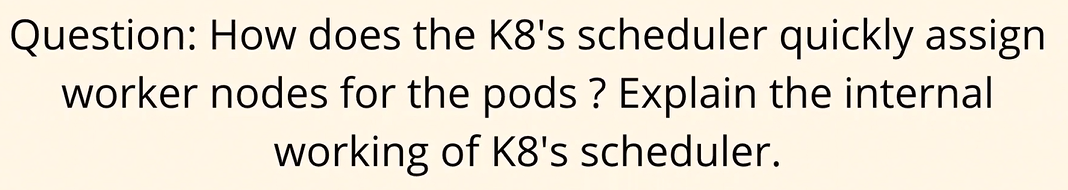
**Logging and Monitoring**: Sidecar containers can capture logs and metrics generated by the main application container and forward them to centralized logging and monitoring systems. This allows for centralized management and analysis of application logs and performance metrics.



We can use taints and toleration method in k8. Taint is a method in which we can label our node acc to our use case because of which it will repel all different kind of pods which doesn’t have the label. For the application which we want we can make sure it ahs the proper label which matches the taint label.

use taints and tolerations when you need to restrict pod scheduling based on node properties, node selector when you want to direct pods to nodes with specific attributes, and node affinity when you need more advanced rules for pod placement based on node characteristics.

in simple terms, use Deployments when you want things to be managed automatically, ReplicaSets when you want an exact number of workers, HPA when you want things to adjust automatically based on how busy it gets, and Manual Scaling when you want to decide things yourself.



We are aware that scheduler is responsible for scheduling the pods on the worker nodes. Scheduler already has the information or what we call the metadata of the worker node i.e how healthy a particular node is. Then this information is filtered and scored. There is a rank that is available for each worker node. When a pod comes for scheduling through its pod scheduling lifecycle, the information is then core related to the data that the scheduler already has. What is the requiremeng of the pod and which worler node is

The Kubernetes scheduler is a component responsible for assigning pods to worker nodes in the cluster. It continuously monitors the API server for newly created pods without a specific node assignment and selects a suitable node for them

1. \*\*Pod Scheduling Process\*\*:

- When a pod is created without a node assignment, it gets added to the scheduling queue.

- The scheduler watches this queue and selects pods one by one for scheduling.

- For each pod, the scheduler evaluates the list of candidate nodes in the cluster.

2. \*\*Node Selection\*\*:

- It filters out nodes that are not eligible for the pod based on based on several factors, including resource requirements and constraints, node affinity, taints and tolerations or other scheduling policies.

3. \*\*Scoring and Prioritization\*\*:

- After filtering the candidate nodes, the scheduler assigns a score to each remaining node based on how well it matches the pod's requirements and constraints.

- The scheduler prioritizes nodes based on this score and selects the node with the highest score as the best fit for the pod.

4. \*\*Binding the Pod to a Node\*\*:

- Once a node is selected, the scheduler updates the pod's scheduling information to reflect the assignment.

- It then sends an API request to the API server to update the pod's status with the assigned node.



Eviction. It is a node setting not a pod setting and it can be done use kubelet. Soft eviction and Hard eviction. based on memory available, inode size, docer space.

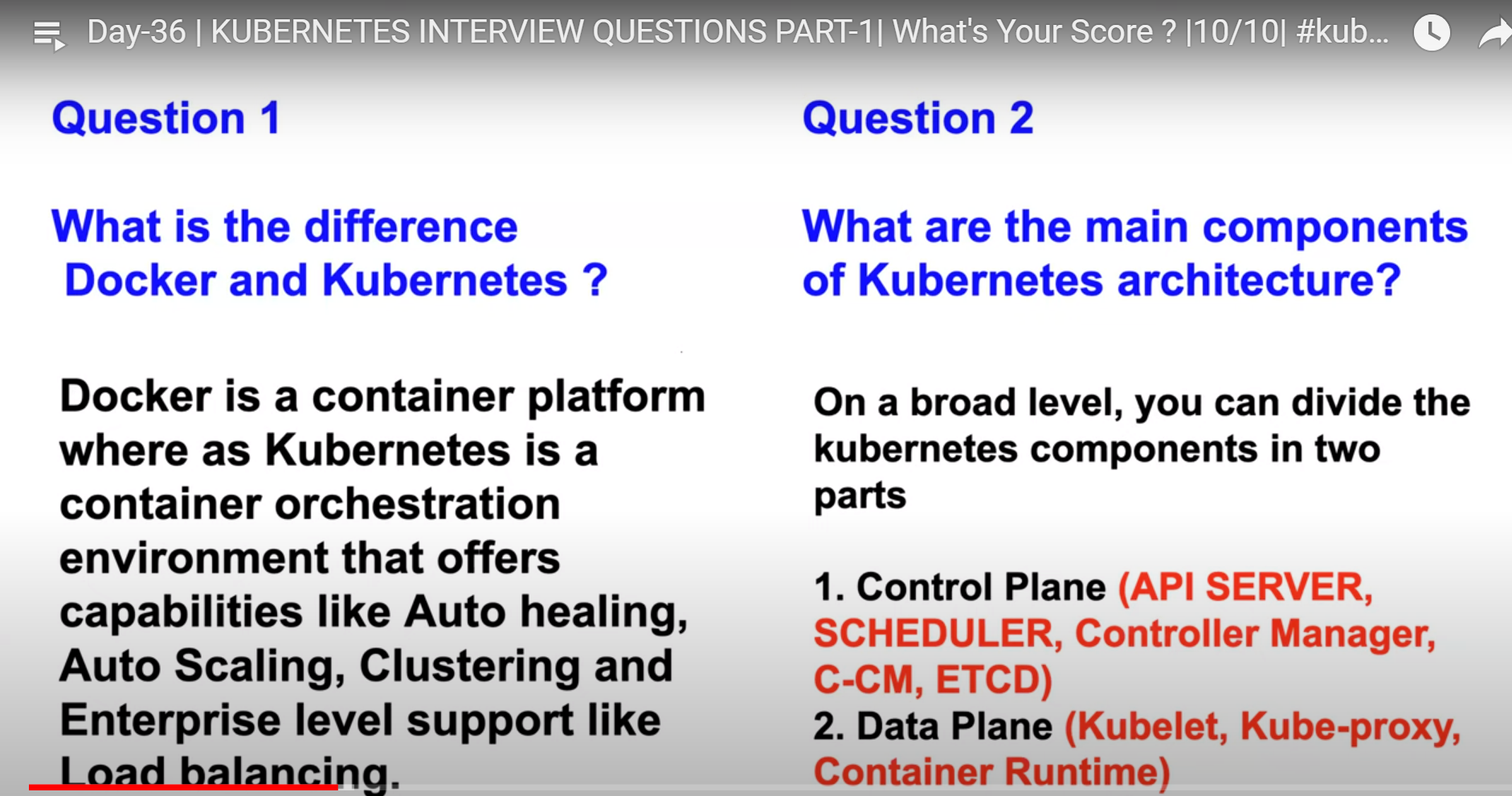
When a node has high disk utilization based on QOS it will select particular pods that it will evict. Once it has decided the pods it will evict then it will terminate these pods and try to shut down gracefully. If it is not possible then it will try to shut them forcefully. What will happen is that all the resouces utilized by those pods will be free

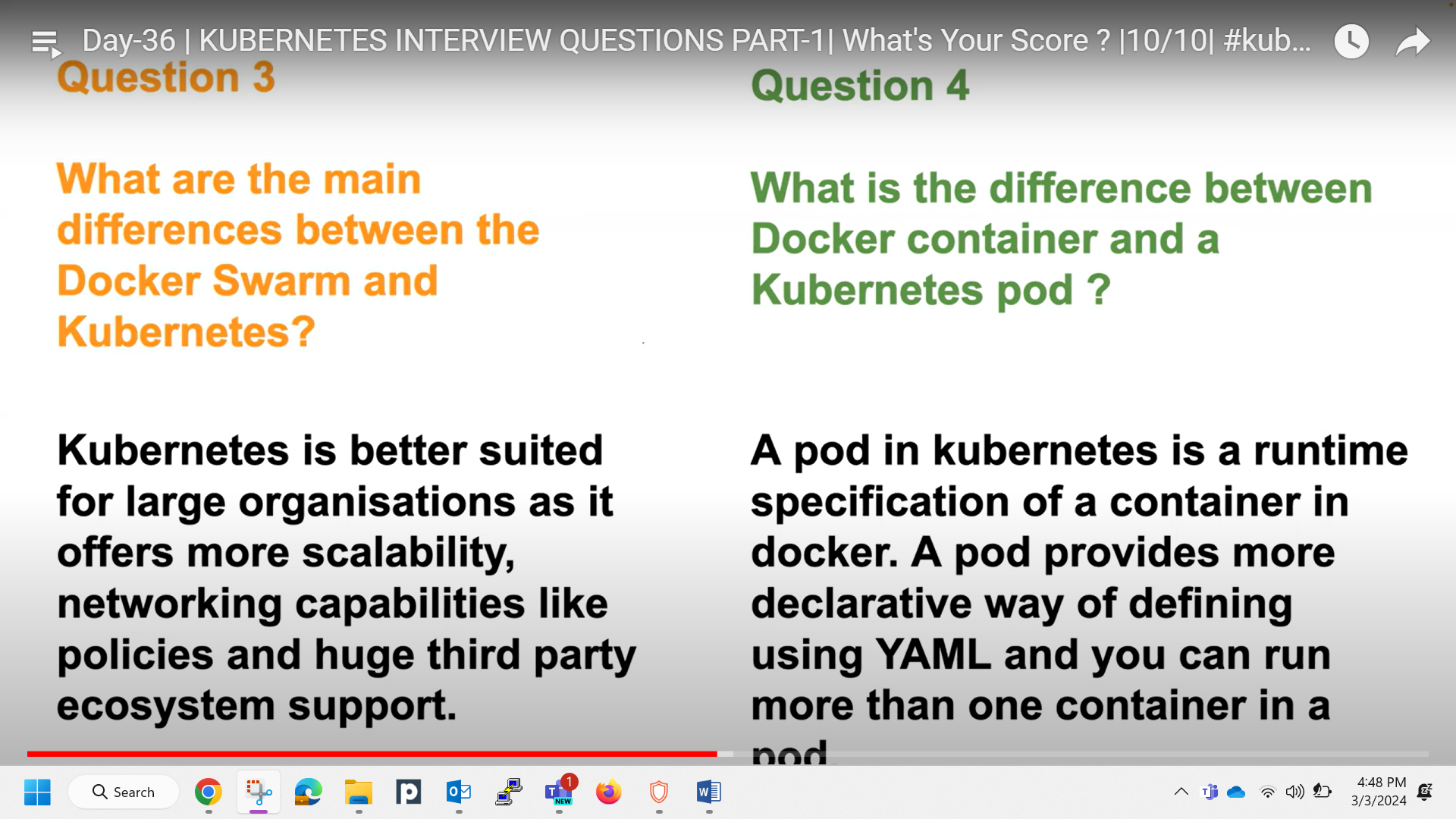
In summary, Quality of Service (QoS) classes in Kubernetes classify pods based on their resource requirements and guarantees. Pods with Guaranteed QoS have dedicated resources reserved for them and are not evicted under normal circumstances. Pods with Burstable QoS can burst beyond their requests but may be subject to eviction if resources become constrained. Pods with BestEffort QoS have no guaranteed resource allocation and are the first to be evicted when resources are scarce.

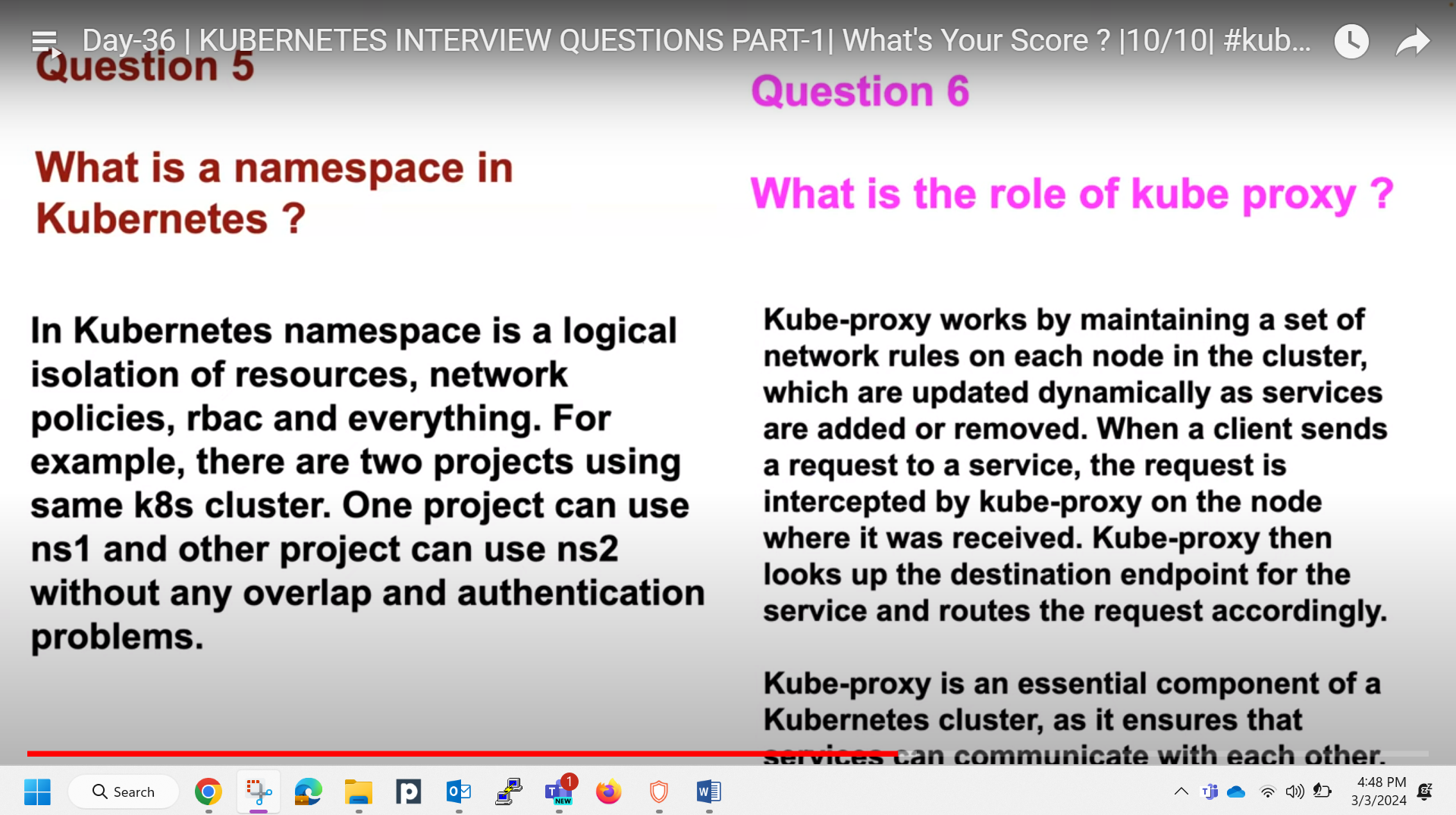
1. Day to day responsibility:  
As part of the DevOps engineer role, we manage Kubernetes clusters for our organization and we also ensure that applications are deployed onto the Kubernetes cluster and there are no issues with the application. So, we have set up monitoring on our Kubernetes cluster. We ensure that whenever there are bugs on the Kubernetes cluster, for example, if the developers are not able to troubleshoot some issue with respect to pods or services, or they are not able to route the traffic inside the Kubernetes cluster, in such cases, as subject matter experts on the Kubernetes clusters, we come into the picture and we solve their problems.

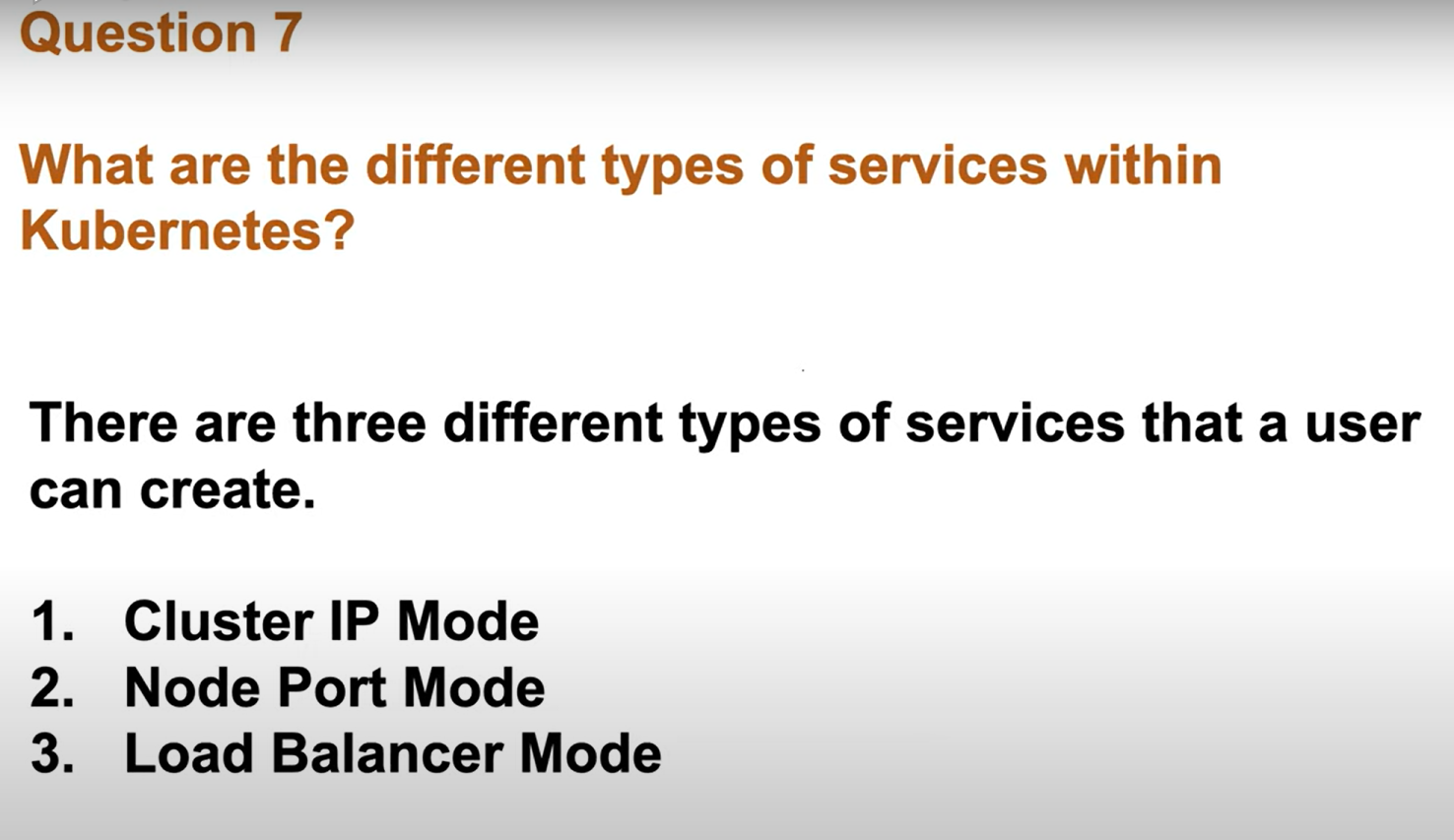
Apart from that, we also do a lot of maintenance activities. For example, we have Kubernetes clusters with three master nodes and ten worker nodes, so we have to do some continuous maintenance activities on these worker nodes. Probably upgrading the versions of these worker nodes or installing some default mandatory packages, ensuring that these worker nodes are not exposed to security vulnerabilities. So, all of these things are our day-to-day activities on Kubernetes.

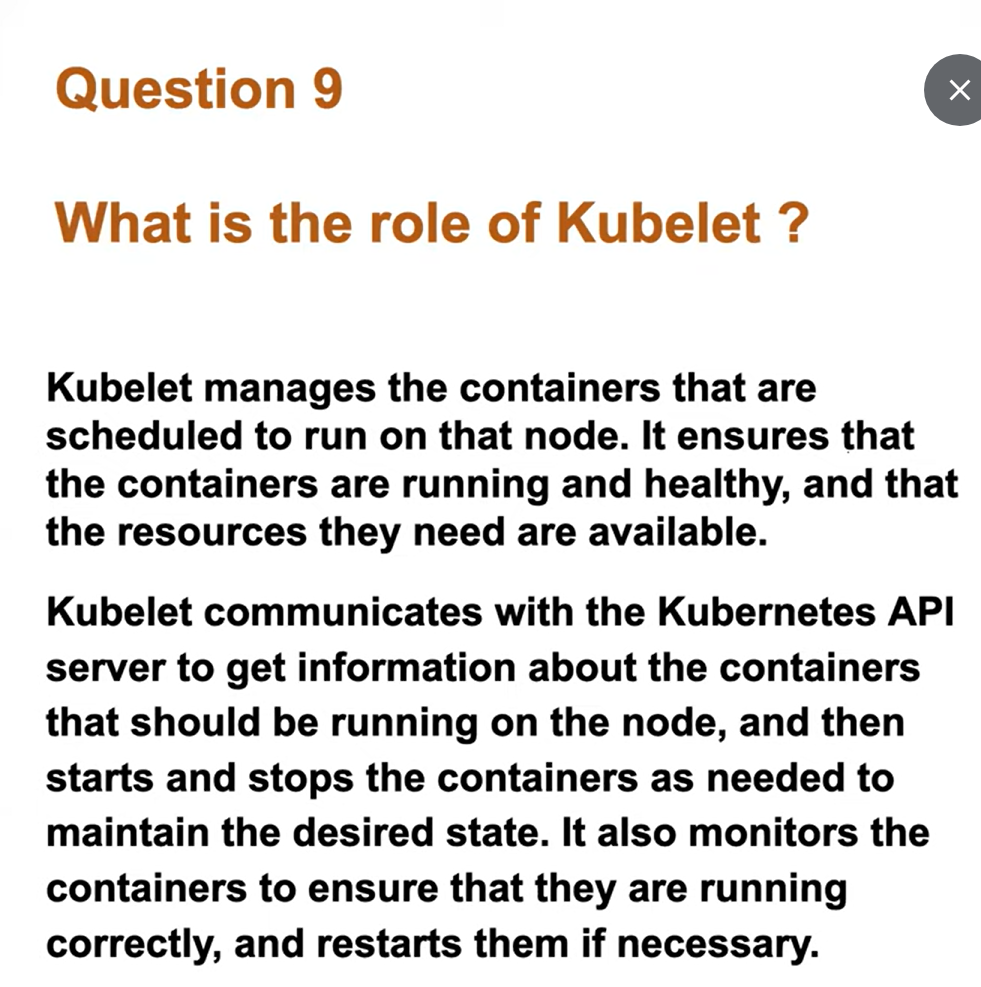
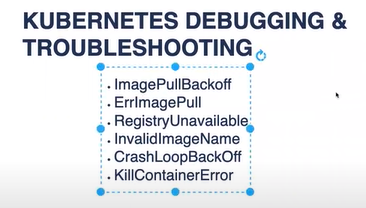
Apart from that, we also serve as subject matter experts on Kubernetes. So, if anyone in the organization has any issues with Kubernetes, they create Jira items for us or they create tickets for us, and we will help them in solving or making them understand the concept of Kubernetes.









  
  
   
diff b/w deployment, daemon set, statefulset

Docker question:

**1. diff b/w add and copy**

copy allows to copy file from host to docker image

Add: copies files from a Url and also extract the tar from source to destination

**2. diff b/w CMD and entrypoint**

**CMD** sets default command and can be overridden, while **ENTRYPOINT** defines the main executable that cannot be overridden directly.

Alright, let's simplify it further:

1. \*\*CMD\*\*:

- It's like a default instruction for what the container should do when it starts.

- For example, if you're running a web server, you might set `CMD` to start that web server by default.

2. \*\*ENTRYPOINT\*\*:

- This is like the main thing your container does. It defines the main executable.

- For instance, if your container is a Python app, `ENTRYPOINT` could be set to run your Python script.

3. \*\*Comparison\*\*:

- \*\*CMD\*\* Gets overridden if a command is provided at runtime.

- \*\*ENTRYPOINT\*\* is fixed, and Does not get overridden.

3.what id docker

4.Docker lifecycle.

5.Docker components.-docker client, docker Daemon, docker host, image, container, docker registery

6. what are n/w types in docker and what is bydefault?

**Bridge network**: This is the default network type in Docker. It allows containers to communicate with each other using the Docker daemon's built-in network bridge. Containers in the same bridge network can communicate with each other via IP addresses, and Docker automatically assigns IP addresses to containers on this network.

Bridge n/w

Bridge network is a default Network and what happens with the bridge network is there is a virtual Ethernet or V8 or Docker 0 Network that is created using which a container can access your host Network for example if your user tries to access the application through your host to the application that is inside the container so if you are creating the

bridge Network what he will do is he will try to firstly access your host from there using this virtual Ethernet or Docker 0 Network he will try to access the application that is inside your container when you do the port mapping so this is a default methodology whereas you can also do the host Network where where using the host Network you will bind your host network with the container Network so container will be part of your host Network itself so users can directly access the applications inside the container using the host Network there is no virtual Ethernet or there is no Docker 0 here in

this case so if you see this example here so this is the virtual ethernet

okay so container virtual ethernet Docker 0 and this is your

host Network this is about the bridge Network and if the host Network you do you don't have all of this mechanism

coming to Overland Network overlay is slightly complicated when you have multiple holes or when you are dealing

with Docker swam or kubernetes these kind of things overlay network will come into picture where you can connect

overlay is basically a networking type where you can connect multiple hosts or

you know you can create any overlay type of networking so it's like a tunnel type of networking

so Mac Miller Network again it's again you know for somebody who is trying to understand the containers you when you

want to explain to the interviewers you can say that this mode allows I mean the Mac Miller Network allows a container to

appear on the network as a physical host rather than a container so what happens is whenever

you are configuring this Mac Villa Network your container will actually appear as a physical host rather than a

19:14

container right so this kind of networkings should only be used in very special cases like you know you have to

19:21

sit down and understand do you require these kind of networkings like the host Network or macvillian network or even

19:27

the overlay networks because when you are doing these kind of Overland networks or Mac Miller Networks you're only complicating because with Docker you shouldn't by default go with this kind of networkings unless uh you know

8. can you explain how to isolate n/w b/w the containers?

how to isolate networking between the containers so this question can be a follow-up to

19:51

your previous question where interviewer will try to understand let's say there

19:57

is a container called payments and this container payments should be completely

20:02

isolated like there should not be any common path if you are talking about the bridge Network like I showed you what

20:08

happens is all the containers use the docker 0 Network by default if you try to create containers on your

20:15

laptop or on your ec2 instance what happens is the default network is its

20:20

virtual ethernet that is Docker 0. so every container use this Docker 0 Network to connect to your host so

20:27

whether it's a login container logout container or a payments container so if you see the example here every container

20:32

uses this V8 network using this weird Docker 0 Network they will try to connect to the host but what I want is I

20:40

want to separate these things like let's say there is a let's assume that this is a login container and this is a finance

20:47

container or payments related container I don't want both of them to use Docker 0 because if both of them are using

20:53

Docker 0 there is no networking isolation so for some reason if the login application or login container is

20:59

compromised there is a security attack on this login container so my finance or

21:05

the payments container should not be compromised so in such cases what I can do is I can do a networking isolation so

21:12

how do you do networking isolation is if you see the picture here what I have done this is my web container or this is

21:18

my login container which is using the default Docker 0 network but if you look at the right side this is not using

21:24

Docker 0 network but it is using my bridge so the answer to this question is you can create your own Bridge

21:30

networkings so using your own Bridge networkings like you can assign your own

21:35

network to a secure container you can say that uh there is create a new

21:41

networking in Docker using the docker Network create secure network okay so now what you can do is whenever you are

21:47

running the container using Docker run command you can say hyphen hyphen network is equals to secure network and

21:53

you can like when you do this the container will come up with a secure network that you have created by default

22:00

Docker will create a cidr block or subnet for you and now whenever the container is trying to access the host

22:06

or whenever it is trying to access ec2 instance it will access the or it will connect to the ec2 instance through that

22:12

secure Network so there is a network proper isolation this kind of isolation can be seen in Virtual machines because

22:19

when you create virtual machines each virtual machine has their own operating system they have their own kernel and

22:25

they have their own ethernet right so every every virtual machine talks to

22:30

your host with a separate Network so to achieve this kind of things in

22:36

Docker containers what you can do is you can create your own Bridge networks okay

22:43

I also showed you this with example in the last class so you can watch day 28 I'll share the video links as well so in

22:51

day 28 I created a live example and I showed you how to do this how to achieve

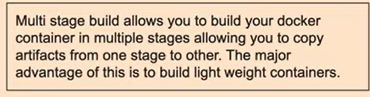
22:56

this network segregation I use the Ping command to also show you that one container is not able to talk to the

23:01

another container got it then what is a multi-stage built-in Docker so these are some of the

8. what is a multistage build in docker?

  
9. what is distro less images in Docker?

