# Kubernetes Interview Questions

**WebMagic Informatica**
[webmagicinformatica.com](https://webmagicinformatica.com)
+91 7021 4791 26

Q1. What is orchestration in software and DevOps?

Q2: How does Kubernetes help with orchestration?

Q3: What is a node in Kubernetes?

Q4: What is Kubernetes?

Q5: How are Kubernetes and Docker related?

Q6: What are the main differences between the Docker Swarm and Kubernetes?

Q7: What is the role of clusters in Kubernetes?

Q8: What is Kubectl used for?

Q9: How can you run Kubernetes locally?

Q10: What is Heapster?

Q11: What are the different components of Kubernetes Architecture?

Q12: What are federated clusters?

Q13: What is a Headless Service?

Q14: What do you understand by load balancer in Kubernetes?

Q15: Name the initial namespaces from which Kubernetes starts?

Q16: What is the Kubernetes controller manager?

Q17: What is a pod in Kubernetes?

Q18: What is the job of the kube-scheduler?

Q19: What is orchestration when it comes to software and DevOps?

Q20: What is ETCD in Kubernetes?

Q21: What do you understand by container resource monitoring?

Q22: What benefits does Kubernetes offer?

Q23: What are namespaces in Kubernetes?

Q24: What is a node in Kubernetes?

Q25: What are the two main components of the Kubernetes architecture?

Q26: What is the role of a pod?

Q27: Where is the cluster data stored in Kubernetes?

Q28: What is the difference between a pod and a job in Kubernetes?

Q29: What is a kubelet?

Q30: What is kube-proxy?

Q31: What is a Heapster in Kubernetes?

Q32: What is a scheduler in the Kubernetes architecture?

Q33: What are Daemon sets?

Q34: How do you determine that a pod is always running?

Q35: What are a few different ways to provide API security on Kubernetes?

Q36: What is load balancing in Kubernetes?

Q37: What is a controller manager, and what are its types?

Q38: What are the uses of the Google Kubernetes Engine?

Q39: How is Kubernetes different from Docker Swarm?

Q40: How to do maintenance on the K8 node?

Q41: What is the role of a pause container?

Q42: Why do we need service mesh?

Q43: How to control the resource usage of a POD?

Q44: What are the units of CPU and memory in POD definition?

Q45: Where else we can set a resource limit?

Q46: How will you update the version of K8?

Q47: Difference between helm and K8 operator?

Q48: Explain the role of CRD (Custom Resource Definition) in K8?

Q49: What are the various K8 related services running on nodes and the role of each service?

Q50: Recommended way of managing the access to multiple clusters?

Q51: What is PDB (Pod Disruption Budget)?

Q52: In what situations daemonsets are normally used?

Q53: When stateful sets are preferred?

Q54: What's init container and when it can be used?

Q55: What are the application deployment strategies?

Q56: How to troubleshoot if the POD is not getting scheduled?

Q57: How to run a POD on a particular node?

Q58: How to ensure PODs are collocated to get performance benefits?

Q59: What are the taints and toleration?

Q60: How to provide persistent storage for POD?

Q61: How do two containers running in a single POD have a single IP address?

Q621: What are the various ways to provide external world connectivity to K8?

Q63: What's the difference between nodeport and load balancer?

Q64: When do we need ingress instead of a LB?

Q65: How does POD to POD communication works?

Q66: How does POD to service communication work?

Q67: How does the service know about healthy endpoints?

Q68: What are the various things that can be done to increase the K8 security?

Q69: How to monitor K8 cluster?

Q70: How to make prometheus HA?

Q71: What are other challenges with prometheus?

Q72: What's the prometheus operator?

Q73: How to get the central logs from POD?

Q75. What is a Kubernetes deployment?

Q76. Explain the use case of Kubernetes deployment?

Q77. What is the difference between a pod and a deployment?

Q78. What is replicaset in kubernetes?

Q79. Can we put multiple containers inside a pod?

Q80. Name some container patterns you come across or use?

Q81. What is Sidecar Container Design?

Q82. When do you use Sidecar Container Pattern?

Q83. What is an Adapter Container Pattern?

Q84. When do you use Adapter Container Patterns?

Q85. What is an Ambassador Container Pattern?

Q86. Point out the tools which are utilized for container monitoring?

Q87. Disadvantages of Kubernetes

Q88. Characteristics of Kubernetes

Q89. What is a Kubernetes Operator?

Q90. Why do we need Operators?

Q91. What difference do you find between Docker Swarm and Kubernetes?

Q92. What difference do you find between deploying applications on the host and containers?

Q93. What is Minikube?

Q94. What are the best security measures that you can take while using Kubernetes?

Q95. What are the types of secrets available in Kubernetes?

Q96. What is a Kubernetes StatefulSet?

Q97. What is the difference between Docker Compose and Kubernetes?

Q98. How can RBAC be used to grant permission to Kubernetes resources?

Q99. How to create a storage class in kubernetes?

Q100. What is KOPS

## Q1. What is orchestration in software and DevOps?

In software development, orchestration refers to the integration of multiple applications or services, allowing them to automate a process or synchronise their data in real-time. If an application runs on multiple microservices that are placed in separate containers, orchestration helps the services communicate seamlessly to achieve a goal.

## Q2: How does Kubernetes help with orchestration?

Kubernetes is an open-source orchestration tool that helps in simplifying DevOps tasks like deployment, scaling, configuration, and others. Most distributed applications are hosted and run in containers. A container is an isolated environment in which the microservices of the app can run together. These containers are managed externally, and they must be scheduled, distributed, and load-balanced to support the infrastructure. But managing these containers becomes challenging in a clustered environment due to data persistence and network configurations.

Kubernetes (blog to get started with Kubernetes) can help overcome these challenges and easily manage a cluster of containers. It can link and orchestrate several containers running on multiple hosts. Kubernetes can manage applications that run on a cluster of hundreds of individual servers. It eliminates most of the manual processes in deploying and scaling containerized applications. Kubernetes allows you to fully implement and fully harness a container-based infrastructure.

- Provisioning and deployment of containers
- Upscaling or removing containers to divide application load evenly all across host infrastructure
- Redundancy and availability of containers
- Moving of containers from one host to another in case there is a shortage of resources in a host (or when the host dies)
- Allocation resources across containers
- Health monitoring of containers and hosts
- Externally exposing services running in a container to the outside world
- Load balancing of service discovery across containers
- Configuring an application relative to the containers running it

## Q3: What is a node in Kubernetes?

A node is the smallest fundamental unit of computing hardware. It represents a single machine in a cluster, which could be a physical machine in a data center or a virtual machine from a cloud provider. Each machine can substitute any other machine in a Kubernetes cluster. The master in Kubernetes controls the nodes that have containers.

### Q4: What is Kubernetes?

Kubernetes is an open-source container orchestration tool or system that is used to automate tasks such as the management, monitoring, scaling, and deployment of containerized applications. It is used to easily manage several containers (since it can handle grouping of containers), which provides for logical units that can be discovered and managed. This is an important Kubernetes Interview Questions

### Q5: How are Kubernetes and Docker related?

Docker is an open-source platform used to handle software development. Its main benefit is that it packages the settings and dependencies that the software/application needs to run into a container, which allows for portability and several other advantages. Kubernetes allows for the manual linking and orchestration of several containers, running on multiple hosts that have been created using Docker.

### Q6: What are the main differences between the Docker Swarm and Kubernetes?

Docker Swarm is Docker's native, open-source container orchestration platform that is used to cluster and schedule Docker containers. Swarm differs from Kubernetes in the following ways:

Docker Swarm is more convenient to set up but doesn't have a robust cluster, while Kubernetes is more complicated to set up but the benefit of having the assurance of a robust cluster

Docker Swarm can't do auto-scaling (as can Kubernetes); however, Docker scaling is five times faster than Kubernetes

Docker Swarm doesn't have a GUI; Kubernetes has a GUI in the form of a dashboard

Docker Swarm does automatic load balancing of traffic between containers in a cluster, while Kubernetes requires manual intervention for load balancing such traffic

Docker requires third-party tools like ELK stack for logging and monitoring, while Kubernetes has integrated tools for the same

Docker Swarm can share storage volumes with any container easily, while Kubernetes can only share storage volumes with containers in the same pod

Docker can deploy rolling updates but can't deploy automatic rollbacks; Kubernetes can deploy rolling updates as well as automatic rollbacks

**Q7: What is the role of clusters in Kubernetes?**

Kubernetes allows you to enforce the required state management by feeding cluster services of a specific configuration. Then, these cluster services run that configuration in the infrastructure. The following steps are involved in the process:

The deployment file contains all the configurations to be fed into the cluster services. The deployment file is fed into the API. Now, the cluster services schedule the pods in the environment Cluster services also ensure that the right number of pods are running So, the Kubernetes cluster is essentially made up of the API, the worker nodes, and the Kubelet process of the nodes.

**Q8: What is Kubectl used for?**

Kubectl is a tool for controlling Kubernetes clusters. In fact, "ctl" stands for control. It is a command-line interface that allows you to pass commands to the cluster and manage the Kubernetes component. This is an important Kubernetes Interview Questions

**Q9: How can you run Kubernetes locally?**

You can run Kubernetes locally using the Minikube tool. It runs a single-node cluster inside a virtual machine on your laptop. So, it provides an efficient way for users who are just getting started and want to try out Kubernetes.

**Q10: What is Heapster?**

Heapster is a performance monitoring and metrics collection tool supported natively on the Kubernetes cluster. It runs like any other pod in the cluster, discovering all nodes and querying information from Kubernetes nodes. This container management tool works via an on-machine agent.

**Q11: What are the different components of Kubernetes Architecture?**

The Kubernetes Architecture has mainly 2 components – the master node and the worker node. As you can see in the below diagram, the master and the worker nodes have many inbuilt components within them. The master node has Kube-controller-manager, Kube-apiserver, kube-scheduler, etc. Whereas the worker node has kubelet and Kube-proxy running on each node.

### Q12: What are federated clusters?

Multiple Kubernetes clusters can be managed as a single cluster with the help of federated clusters. So, you can create multiple Kubernetes clusters within a data centre/cloud and use the federation to control/manage them all in one place.

The federated clusters can achieve this by doing the following two things. Refer to the below diagram.

### Q13: What is a Headless Service?

Headless Service is similar to that of a 'Normal' service but does not have a Cluster IP. This service enables you to directly reach the pods without the need of accessing it through a proxy.

### Q14: What do you understand by load balancer in Kubernetes?

A load balancer is one of the most common and standard ways of exposing service. There are two types of load balancers used based on the working environment i.e. either the Internal Load Balancer or the External Load Balancer. The Internal Load Balancer automatically balances load and allocates the pods with the required configuration whereas the External Load Balancer directs the traffic from the external load to the backend pods.

### Q15: Name the initial namespaces from which Kubernetes starts?

Default Kube – system Kube – public

### Q16: What is the Kubernetes controller manager?

The controller manager is a daemon that is used for embedding core control loops, garbage collection, and Namespace creation. It enables the running of multiple processes on the master node even though they are compiled to run as a single process.

### Q17: What is a pod in Kubernetes?

Pods are high-level structures that wrap one or more containers. This is because containers are not run directly in Kubernetes. Containers in the same pod share a local network and the same resources, allowing them to easily communicate with other containers in the same pod as if they were on the same machine while at the same time maintaining a degree of isolation.

## Q18: What is the job of the kube-scheduler?

The kube-scheduler assigns nodes to newly created pods.

## Q19: What is orchestration when it comes to software and DevOps?

Orchestration refers to the integration of multiple services that allows them to automate processes or synchronise information in a timely fashion. Say, for example, you have six or seven microservices for an application to run. If you place them in separate containers, this would inevitably create obstacles for communication. Orchestration would help in such a situation by enabling all services in individual containers to work seamlessly to accomplish a single goal.

## Q20: What is ETCD in Kubernetes?

Etcd is a store for the configuration, state, and metadata of Kubernetes clusters. It is written in the Go programming language and represents the cluster state at a given point in time. This datastore serves as the backbone of distributed systems.

## Q21: What do you understand by container resource monitoring?

From the user perspective, it is vital to understand resource utilisation at different abstraction layers and levels, like container pods, services, and the entire cluster. Each level can be monitored using various tools, namely:

Grafana Heapster InfluxDB CAdvisor Prometheus

## Q22: What benefits does Kubernetes offer?

Kubernetes makes the handling of containers very easy. It helps you deploy applications faster, leading to an improved response to customer demands. Kubernetes helps automate scheduling and rollbacks. Kubernetes is also ideal for cloud-native apps requiring rapid scaling since it can cluster together groups of hosts across private, public, or hybrid clouds.

## Q23: What are namespaces in Kubernetes?

A namespace in Kubernetes is used to divide the resources in the cluster among multiple users. This is particularly helpful in environments where there are multiple users scattered over a vast area geographically.

### Q24: What is a node in Kubernetes?

A node is the smallest unit in the computing system. It is a single machine in a cluster formerly referred to as a minion since it is a worker unit that runs the pods. It can be either a physical machine or a virtual machine and is controlled by the master components in the Kubernetes system.

### Q25: What are the two main components of the Kubernetes architecture?

The Kubernetes architecture has two primary components – the master node and worker nodes. The master node consists of the API server, the scheduler, the controller manager, and the etcd components. The worker nodes have services such as container run time and Kube proxy that run on each node.

### Q26: What is the role of a pod?

A pod in Kubernetes is responsible for holding individual containers. Each pod can hold various containers depending on the configurations and requirements. The containers held within a single pod share the same resources and the same local network, which makes it easier for them to communicate.

### Q27: Where is the cluster data stored in Kubernetes?

The main data storage component in the Kubernetes system is the etcd. This is where the cluster data is stored.

### Q28: What is the difference between a pod and a job in Kubernetes?

The difference lies in their individual functions. The function of a pod is to ensure that a container is running. The function of a job is to ensure that the pods run to the completion of the job.

### Q29: What is a kubelet?

A kubelet can be considered the lowest level component in the Kubernetes system. Its function is to control a set of pods and ensure that all the containers in a given pod are running throughout its lifecycle.

### Q30: What is kube-proxy?

Kube-proxy manages the host sub-netting and makes services available to other components of the system. Its function is to direct and correct containers based on the IP and port number of incoming requests, thus helping with load balancing.

## Q31: What is a Heapster in Kubernetes?

Heapster is a performance monitoring tool in Kubernetes. It works like just another pod in the cluster and collects metrics on all nodes. It has native support for Kubernetes.

## Q32: What is a scheduler in the Kubernetes architecture?

A scheduler, or kube-scheduler, is responsible for distributing the workload among the worker nodes. It schedules the tasks and monitors the resource use for each node to ensure the distribution is done correctly.

## Q33: What are Daemon sets?

Daemon sets are a set of pods that are made to run on a host, only once. They are used for host layer attributes, like network monitoring, which are usually run only once on a host.

## Q34: How do you determine that a pod is always running?

This can be determined by introducing probes, such as a liveness probe. It can check if the application inside a pod is running. If it shows failure, the container will get restarted.

## Q35: What are a few different ways to provide API security on Kubernetes?

To provide API security on Kubernetes, some methods we can use are:

Use the correct authorization mode with the API server Use API authentication Ensure that all incoming traffic is protected by TLS Use authorization-mode=Webhook to make kubeless protect the API Use restrictive RBAC role policy on the kube-dashboard Remove any default service account permissions

## Q36: What is load balancing in Kubernetes?

The load balancing process allows you to expose services to the internet. Internal load balancing refers to automatically balancing the loads on pods and allocating them with the required configuration. External load balancing refers to directing traffic from external loads to the backend pods.

## Q37: What is a controller manager, and what are its types?

The Kubernetes controller manager is a daemon used for embedding core control loops that regulate the system. It helps to run multiple processes on the Master node.

## Q38: What are the uses of the Google Kubernetes Engine?

The GKE, also known as Google Container Engine, is a management platform for Docker containers and clusters. GKE lets you orchestrate container clusters that run within Google's public cloud services.

## Q39: How is Kubernetes different from Docker Swarm?

Docker Swarm is a default orchestration tool that comes integrated with Docker. It can orchestrate simple Docker containers. Docker Swarm does not do auto-scaling or deploy automatic rollbacks. Kubernetes is also an orchestration tool, but it can manage much more complex applications as it has a more robust cluster. Kubernetes performs auto-scaling and automatic rollbacks. Docker Swarm is easier to install, though, as compared to Kubernetes. Docker Swarm also does automatic load balancing between containers, which is not possible in Kubernetes. So, both tools have their set of pros and cons.

## Q40: How to do maintenance on the K8 node?

Maintenance activities are an inevitable part of administration, you may need to do the patching or apply some security fixes on K8. Mark the node unschedulable and then drain the PODs which are present on the K8 node.

```
kubectl cordon
```

```
kubectl drain --ignore-daemonsets
```

It's important to include the --ignore-daemonsets for any daemonset running on this node. Just in case if any statefulset is running on this node, and if no more node is available to maintain the count of stateful sets then the statefulset POD will be in pending status.

## Q41: What is the role of a pause container?

Pause container servers as the parent container for all the containers in your POD.

It serves as the basis of Linux namespace sharing in the POD.

PID 1 for each POD to reap the zombie processes.

### Q42: Why do we need service mesh?

A service mesh ensures that communication among containerized and often ephemeral application infrastructure services is fast, reliable, and secure. The mesh provides critical capabilities including service discovery, load balancing, encryption, observability, traceability, authentication and authorization, and support for the circuit breaker pattern.

### Q43: How to control the resource usage of a POD?

With requests and limits, resource usage of a POD can be controlled.

request: the amount of resources being requested for a container. If a container exceeds its request for resources, it may be throttled back down to its request.

limit: an upper cap on the resources a container is able to use. If it tries to exceed this limit it may be terminated if Kubernetes decides that another container needs the resources. If you're sensitive to pod restarts, it makes sense to have the sum of all container resource limits equal or less than the total resource capacity for your cluster.

### Q44: What are the units of CPU and memory in POD definition?

CPU is in milicores and memory in bytes. CPU can be easily throttled but not memory.

### Q45: Where else we can set a resource limit?

You may also set resource limits on a namespace. This is helpful in scenarios where people have a habit of not defining the resource limits in POD definition.

### Q46: How will you update the version of K8?

Before doing the update of K8, it's important to read the release notes to understand the changes introduced in newer versions and whether version update will also update the etcd.

https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade-1-12/

## Q47: Difference between helm and K8 operator?

An Operator is an application-specific controller that extends the Kubernetes API to create, configure and manage instances of complex stateful applications on behalf of a Kubernetes user. It builds upon the basic Kubernetes resource and controller concepts, but also includes domain or application-specific knowledge to automate common tasks better managed by computers. On the other hand, helm is a package manager like yum or apt-get.

## Q48: Explain the role of CRD (Custom Resource Definition) in K8?

A custom resource is an extension of the Kubernetes API that is not necessarily available in a default Kubernetes installation. It represents a customization of a particular Kubernetes installation. However, many core Kubernetes functions are now built using custom resources, making Kubernetes more modular.

## Q49: What are the various K8 related services running on nodes and the role of each service?

Mainly K8 cluster consists of two types of nodes: master and executor

master services:

kube-apiserver: Master API service which acts like a door to the K8 cluster.

kube-scheduler: Schedule PODs according to available resources on executor nodes.

kube-controller-manager: controller is a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the current state towards the desired state

executor node: (This also runs on the master node)

kube-proxy: The Kubernetes network proxy runs on each node. This reflects services as defined in the Kubernetes API on each node and can do simple TCP, UDP, and SCTP stream forwarding or round robin TCP, UDP, and SCTP forwarding across a set of backends.

kubelet: kubelet takes a set of PodSpecs that are provided through various mechanisms (primarily through the apiserver) and ensures that the containers described in those PodSpecs are running and healthy

## Q50: Recommended way of managing the access to multiple clusters?

kubectl looks for the config file; multiple cluster access information can be specified in this config file. kubectl config commands can be used to manage the access to these clusters.

## Q51: What is PDB (Pod Disruption Budget)?

A PDB specifies the number of replicas that an application can tolerate having, relative to how many it is intended to have. For example, a Deployment which has a .spec.replicas: 5 is supposed to have 5 pods at any given time. If its PDB allows for there to be 4 at a time, then the Eviction API will allow voluntary disruption of one, but not two pods, at a time. This is applicable for voluntary disruptions.

## Q52: In what situations daemonsets are normally used?

Daemonset is used to start the PODs on every node in the cluster. It's used generally to run the monitoring or logging agents which are supposed to run on every executor node in the cluster.

## Q53: When stateful sets are preferred?

When you are running the applications which require quorum, basically the applications which are not truly stateless for those applications, stateful sets are required.

## Q54: What's init container and when it can be used?

init containers will set a stage for you before running the actual POD.

Wait for some time before starting the app Container with a command like sleep 60.

Clone a git repository into a volume.

## Q55: What are the application deployment strategies?

In this agile world there is continuous demand of upgrading the applications, we have multiple options for deploying the new version of app:

Recreate: Old style, existing application version is destroyed and new version is deployed. Significant amount of downtime.

Rolling update: Gradually bringing down the existing deployment and introducing the new versions. You decide how many instances can be upgraded at a single point of time.

Shadow: Traffic going to the existing version of the application is replicated to the new version to see it's working. Istio provides this pattern.

A/B Testing using Istio: Running multiple variants of application together and determining the best one based on user traffic. It's more for management decisions.

Blue/Green : Blue is mainly about switching the traffic from one version of the app to another version.

Canary deployment : In which a certain percentage of traffic is shifted from one version to another. If things work well we will keep on increasing the traffic shift. It's different from the rolling update in which the existing version count is reduced gradually.

## Q56: How to troubleshoot if the POD is not getting scheduled?

There are many factors which can lead to unstartable POD. Most common one is running out of resources, use the commands like kubectl describe <POD> -n <Namespace> to see the reason why POD is not started. Also, keep an eye on kubectl get events to see all events coming from the cluster.

## Q57: How to run a POD on a particular node?

Various methods are available to achieve it.

nodeName: specify the node name in the POD spec, it will try to run the POD on a specific node.

nodeSelector : you may assign a specific label to nodes which have special resources and use the same label in POD spec so that POD will run only on that node.

nodeaffinities: requiredDuringSchedulingIgnoredDuringExecution, preferredDuringSchedulingIgnoredDuringExecution are hard, soft requirements for running the POD on specific nodes. This will be replacing nodeSelector in future. It depends on the node labels.

## Q58: How to ensure PODs are collocated to get performance benefits?

podAntiAffinity and podAffinity are the affinity concepts to not keep and keep the PODs on the same node. Key point to note is that it depends on the POD labels.

## Q59: What are the taints and toleration?

Taints allow a node to repel a set of pods. You can set taints on the node and only the POD which have tolerations matching the taints condition will be able to run on those nodes. This is useful in the case when you allocated a node for one user and don't want to run the PODs from other users on that node.

## Q60: How to provide persistent storage for POD?

Persistent volumes are used for persistent POD storage. They can be provisioned statically or dynamically.

Static : A cluster administrator creates a number of PVs. They carry the details of the real storage which is available for use by cluster users.

Dynamically : Administrator creates a PVC (Persistent volume claim) specifying the existing storage class and volume created dynamically based on PVC.

## Q61: How do two containers running in a single POD have a single IP address?

Kubernetes implements this by creating a special container for each pod whose only purpose is to provide a network interface for the other containers. There is one pause container which is responsible for namespace sharing in the POD. Generally, people ignore the existence of this pause container but actually this container is the heart of the network and other functionalities of POD. It provides a single virtual interface which is used by all containers running in a POD.

## Q621: What are the various ways to provide external world connectivity to K8?

By default POD should be able to reach the external world but for vice-versa we need to do some work. Following options are available to connect with POD from the outer world.

Nodeport (it will expose one port on each node to communicate with it)

Load balancers (L4 layer of TCP/IP protocol)

Ingress (L7 layer of TCP/IP Protocol)

One another method is kube-proxy which can be used to expose a service with only cluster IP on the local system port.

```
$ kubectl proxy --port=8080 $
http://localhost:8080/api/v1/proxy/namespaces//services/:
/
```

### Q63: What's the difference between nodeport and load balancer?

nodport relies on the IP address of your node. Also, you can use the node ports only from the range 30000–32767, on the other hand the load balancer will have its own IP address. All the major cloud providers support creating the LB for you if you specify the LB type while creating the service. On bare metal based clusters, metallb is promising.

### Q64: When do we need ingress instead of a LB?

For each service you have one LB. You can have single ingress for multiple services. This will allow you to do both path based and subdomain based routing to backend services. You can do the SSL termination at ingress.

### Q65: How does POD to POD communication works?

For POD to POD communication, it's always recommended to use the K8 service DNS instead of POD IP because PODs are ephemeral and their IPs can change after the redeployment.

If the two PODs are running on the same host then the physical interface will not come into the picture.

Packet will leave the POD1 virtual network interface and go to docker bridge (cbr0).

The Docker bridge will forward the packet to the right POD2 which is running on the same host.

If two PODs are running on a different host then the physical interface of both host machines will come into the picture. Let's consider a scenario in which CNI is not used.

POD1 = 192.168.2.10/24 (node1, cbr0 192.168.2.1) POD2 = 192.168.3.10/24 (node2, cbr1 192.168.3.1)

POD1 will send the traffic destined for POD2 to its GW (cbr0) because both are in different subnets.

GW doesn't know about the 192.168.3.0/24 network hence it will forward the traffic to the physical interface of node1.

node1 will forward the traffic to its own physical router/gateway.

That physical router/GW should have the route for 192.168.3.0/24 network to route the traffic to node2.

Once traffic reaches node2, it passes that traffic to POD2 through cbr1

If the Calico CNI it's responsible for adding the routes for cbr (docker bridge IP address) in all nodes.

**Q66: How does POD to service communication work?**

PODs are ephemeral, their IP addresses can change hence to communicate with POD in a reliable way service is used as a proxy or load balancer. A service is a type of Kubernetes resource that causes a proxy to be configured to forward requests to a set of pods. The set of pods that will receive traffic is determined by the selector, which matches labels assigned to the pods when they were created. K8 provides an internal cluster DNS that resolves the service name.

Service is using a different internal network than the POD network. Netfilter rules which are injected by kube-proxy are used to redirect the request actually destined for service IP to the right POD.

**Q67: How does the service know about healthy endpoints?**

The kubelet running on the worker node is responsible for detecting the unhealthy endpoints, it passes that information to API server then eventually this information is passed to kube-proxy which will adjust the netfilter rules accordingly.

I highly recommend reading the following series to get a solid understanding about the K8 networking.

**Q68: What are the various things that can be done to increase the K8 security?**

This is a huge topic, I am sharing some thoughts on it.

By default, POD can communicate with any other POD. We can set up network policies to limit this communication between the PODs.

RBAC (Role-based access control) to narrow down the permissions.

Use namespaces to establish security boundaries.

Set the admission control policies to avoid running the privileged containers.

Turn on audit logging.

Monitoring

**Q69: How to monitor K8 cluster?**

Prometheus is used for K8 monitoring. The Prometheus ecosystem consists of multiple components. main Prometheus server which scrapes and stores time series data client libraries for instrumenting application code a push gateway for supporting short-lived jobs special-purpose exporters for services like HAProxy, StatsD, Graphite, etc. an alertmanager to handle alerts various support tools

**Q70: How to make prometheus HA?**

You may run multiple instances of prometheus HA but grafana can use only one of them as a datasource. You may put a load balancer in front of multiple prometheus instances, use sticky sessions and failover if one of the prometheus instances dies. This makes things complicated. Thanos is another project which solves these challenges.

**Q71: What are other challenges with prometheus?**

Despite being very good at K8 monitoring, Prometheus still has some issues:

Prometheus HA support.

No downsampling is available for collected metrics over the period of time.

No support for object storage for long term metric retention.

All of these challenges are again overcome by Thanos.

**Q72: What's the prometheus operator?**

The mission of the Prometheus Operator is to make running Prometheus on top of Kubernetes as easy as possible, while preserving configurability as well as making the configuration Kubernetes native.

### Q73: How to get the central logs from POD?

This architecture depends upon application and many other factors. Following are the common logging pattern

Node level logging agent

Streaming sidecar container

Sidecar container with logging agent

Export logs directly from the application

In our setup, filebeat and journalbeat are running as a daemonset. Logs collected by these are dumped to kafka topics which are eventually dumped to the ELK stack.

Same can be achieved using EFK stack and fluentd-bit.

### Q75. What is a Kubernetes deployment?

A Kubernetes deployment is a resource object in Kubernetes that provides declarative updates to applications. A deployment allows you to describe an application's life cycle, such as which images to use for the app, the number of pods there should be, and the way in which they should be updated.

Example: The following is an example of a Deployment. It creates a ReplicaSet to bring up three nginx Pods:

```
# controllers/nginx-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

  name: nginx-deployment

  labels:

    app: nginx

spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

**Q76. Explain the use case of Kubernetes deployment?**

- Create a Deployment to rollout a ReplicaSet. The ReplicaSet creates Pods in the background. Check the status of the rollout to see if it succeeds or not.
- Declare the new state of the Pods by updating the PodTemplateSpec of the Deployment. A new ReplicaSet is created and the Deployment manages moving the Pods from the old ReplicaSet to the new one at a controlled rate. Each new ReplicaSet updates the revision of the Deployment.
- Rollback to an earlier Deployment revision if the current state of the Deployment is not stable. Each rollback updates the revision of the Deployment.
- Scale up the Deployment to facilitate more load.

- Pause the Deployment to apply multiple fixes to its PodTemplateSpec and then resume it to start a new rollout.
- Use the status of the Deployment as an indicator that a rollout has stuck.
- Clean up older ReplicaSets that you don't need anymore

## Q77. What is the difference between a pod and a deployment?

A pod is the core building block for running applications in a Kubernetes cluster; a deployment is a management tool used to control the way pods behave.

Both Pod and Deployment are full-fledged objects in the Kubernetes API. Deployment manages creating Pods by means of ReplicaSets. What it boils down to is that Deployment will create Pods with specs taken from the template. It is rather unlikely that you will ever need to create Pods directly for a production use-case.

## Q78. What is replicaset in kubernetes?

A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical Pods.

```
# controllers/frontend.yaml

apiVersion: apps/v1

kind: ReplicaSet

metadata:

  name: frontend

  labels:

    app: guestbook

    tier: frontend

spec:

  # modify replicas according to your case

  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```

### Q79. Can we put multiple containers inside a pod?

Yes. A pod that contains one container refers to a single container pod and it is the most common kubernetes use case. A pod that contains Multiple co-related containers refers to a multi-container pod.

### Q80. Name some container patterns you come across or use?

1. Init Container Pattern
2. Sidecar Container Pattern
3. Adapter Container Pattern
4. Ambassador Container Pattern

### Q81. What is Sidecar Container Design?

Sidecar containers are the containers that should run along with the main container in the pod. This sidecar pattern extends and enhances the functionality of current containers without changing it.

Imagine that you have the pod with a single container working very well and you want to add some functionality to the current container without touching or changing. How can you add the additional functionality or extend the current functionality? This sidecar container pattern really helps exactly in that situation.

All the Containers will be executed parallelly and the whole functionality works only if both types of containers are running successfully. Most of the time these sidecar containers are simple and small and consume fewer resources than the main container.

### Q82. When do you use Sidecar Container Pattern?

- Whenever you want to extend the functionality of the existing single container pod without touching the existing one.
- Whenever you want to enhance the functionality of the existing single container pod without touching the existing one.
- You can use this pattern to synchronise the main container code with the git server pull.
- You can use this pattern for sending log events to the external server.
- You can use this pattern for network-related tasks.

### Q83. What is an Adapter Container Pattern?

There are so many applications that are heterogeneous in nature which means they don't contain the same interface or are not consistent with other systems. This pattern extends and enhances the functionality of current containers without changing it as the sidecar container pattern.

Imagine that you have the pod with a single container working very well, but it doesn't have the same interface with other systems to integrate or work with it. How can you make this container to have a unified interface with a standardised format so that other systems can access your container? This adapter container pattern really helps exactly in that situation.

All the Containers will be executed parallelly and the whole functionality works only if both types of containers are running successfully. Most of the time these adapter containers are simple and small and consume fewer resources than the main container.

### Q84. When do you use Adapter Container Patterns?

Whenever you want to extend the functionality of the existing single container pod without touching the existing one.

Whenever you want to enhance the functionality of the existing single container pod without touching the existing one.

Whenever there is a need to convert or standardise the format for the rest of the systems.

**Q85. What is an Ambassador Container Pattern?**

The Ambassador container is a special type of sidecar container which simplifies accessing services outside the Pod. When you are running applications on kubernetes, there's a high chance that you should access the data from the external services. The Ambassador container hides the complexity and provides a uniform interface to access these external services.

Imagine that you have the pod with one container running successfully, but you need to access external services. But, these external services are dynamic in nature or difficult to access. Sometimes there is a different format that external service returns. There are some other reasons as well and you don't want to handle this complexity in the main container. So, we use the Ambassador containers to handle these kinds of scenarios.

All the Containers will be executed parallelly and the whole functionality works only if both types of containers are running successfully. Most of the time these ambassador containers are simple and small and consume fewer resources than the main container.

**Q86. Point out the tools which are utilized for container monitoring?**

- Grafana
- cAdvisor
- Heapster
- InfluxDB
- Prometheus

**Q87. Disadvantages of Kubernetes**

- Kubernetes dashboard isn't as useful as it ought to be
- Security isn't viable.
- It is intricate and can diminish profitability
- Kubernetes is more costly than its other options.

## Q88. Characteristics of Kubernetes

- Self-Healing Capabilities
- Automated Scheduling
- Application-centric management
- You could make predictable infrastructure
- Automated rollouts & rollback
- Offers a higher density of resource utilisation
- Horizontal Scaling & Load Balancing
- Provides enterprise-ready features
- Auto-scalable infrastructure
- Provides environment consistency for testing, development, and production.
- Infrastructure is lightly coupled to each segment and can act as a separate unit.

## Q89. What is a Kubernetes Operator?

Operators are software extensions to K8s which make use of custom resources to manage applications and their components. Operators follow Kubernetes principles, notably the control loop.

## Q90. Why do we need Operators?

The process of managing applications in Kubernetes isn't as straightforward as managing stateless applications, where reaching the desired status and upgrades are both handled the same way for every replica. In stateful applications, upgrading each replica might require different handling due to the stateful nature of the app; each replica might be in a different state. As a result, we often need a human operator to manage stateful applications. The Kubernetes Operator is supposed to assist with this.

This will also help with automating a standard process on multiple Kubernetes clusters

## Q91. What difference do you find between Docker Swarm and Kubernetes?

| Parameters | Kubernetes | Docker Swarm |
|---|---|---|
| GUI | Kubernetes Dashboard is the GUI | Has no GUI |

| | | |
|---|---|---|
| Installation & cluster configuration | Setups are quite complicated but the cluster is robust. | Setup is easy but the cluster is not robust. |
| Auto-scaling | Can do auto-scaling. | Cannot do auto-scaling. |
| Scalability | Scales fast. | Scales 5 times faster than Kubernetes. |
| Load Balancing | Manual support needed for load balancing traffic between containers & pods. | Does auto load balancing of traffic between containers in clusters. |
| Data volumes | Can only share storage volumes with containers in the same pod. | Can share storage volumes with other containers. |
| Rolling updates and rollbacks | Does rolling updates and automatic rollbacks. | Can do rolling updates but no automatic rollbacks. |
| Logging and monitoring | Has in-built tools to perform logging and monitoring. | Requires 3rd party tools like ELK stack to do logging and monitoring. |

### Q92. What difference do you find between deploying applications on the host and containers?

When you deploy the application on hosts:

- There will be an operating system and that operating system will have a kernel which again will have diverse libraries (installed on the operating system) that are required for the application.
- In this kind of framework, you can have several applications and you will see all the applications sharing the libraries present in the operating system.

When you deploy an application on the container:

- In this architecture, you will have a kernel which will be the only common thing between all the applications.
- Here you will see every application has their necessary libraries and binaries isolated from the rest of the system, which cannot be approached by any other application.
- Like if one app needs access to Python, that particular app will get it, if the particular application needs access to Java, then only that particular app will have access to Java.

### Q93. What is Minikube?

Minikube is a tool used for easy running Kubernetes locally, it runs a single-code Kubernetes cluster within a virtual machine.

## Q94. What are the best security measures that you can take while using Kubernetes?

- By restricting access to ETCD
- By applying security updates to the environment regularly
- By implementing network segmentation
- By logging everything on the producing environment
- By having continuous security vulnerability scanning
- By having a strict policy or protocol for resources
- By enabling auditing
- By defining resource quota
- By limiting direct access to Kubernetes nodes
- By using images from the authorised repository only

## Q95. What are the types of secrets available in Kubernetes?

| Builtin Type | Usage |
|---|---|
| Opaque | arbitrary user-defined data |
| kubernetes.io/service-account-token | service account token |
| kubernetes.io/dockercfg | serialized ~/.dockercfg file |
| kubernetes.io/dockerconfigjson | serialized ~/.docker/config.json file |
| kubernetes.io/basic-auth | credentials for basic authentication |
| kubernetes.io/ssh-auth | credentials for SSH authentication |
| kubernetes.io/tls | data for a TLS client or server |
| bootstrap.kubernetes.io/token | bootstrap token data |

## Q96. What is a Kubernetes StatefulSet?

StatefulSet is the workload API object used to manage stateful applications. Manages the deployment and scaling of a set of Pods, and provides guarantees about the ordering and uniqueness of these Pods.

Like a Deployment, a StatefulSet manages Pods that are based on an identical container spec. Unlike a Deployment, a StatefulSet maintains a sticky identity for each of their Pods. These pods are created from the same spec, but are not interchangeable: each has a persistent identifier that it maintains across any rescheduling.

Using StatefulSets:

StatefulSets are valuable for applications that require one or more of the following.

- Stable, unique network identifiers.
- Stable, persistent storage.
- Ordered, graceful deployment and scaling.
- Ordered, automated rolling updates.

### Q97. What is the difference between Docker Compose and Kubernetes?

Docker (or specifically, the docker command) is used to manage individual containers, docker-compose is used to manage multi-container applications and Kubernetes is a container orchestration tool.

Docker Compose:

- Docker Compose is the declarative version of the docker cli
- It can start one or more containers
- It can create one or more networks and attach containers to them
- It can create one or more volumes and configure containers to mount them
- All of this is for use on a single host

Kubernetes:

- Kubernetes is a platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.
- are fault-tolerant,
- can scale, and do this on-demand
- use resources optimally
- can discover other applications automatically, and communicate with each other
- can update/rollback without any downtime.

## Q98. How can RBAC be used to grant permission to Kubernetes resources?

Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within your organisation.

RBAC authorization uses the rbac.authorization.k8s.io API group to drive authorization decisions, allowing you to dynamically configure policies through the Kubernetes API.

## Q99. How to create a storage class in kubernetes?

A StorageClass provides a way for administrators to describe the "classes" of storage they offer. Each StorageClass contains the fields provisioner, parameters, and reclaimPolicy, which are used when a PersistentVolume belonging to the class needs to be dynamically provisioned.

Administrators set the name and other parameters of a class when first creating StorageClass objects, and the objects cannot be updated once they are created. Administrators can specify a default StorageClass only for PVCs that don't request any particular class to bind to.

## Q100. What is KOPS

Kops, short for Kubernetes Operations, is a set of tools for installing, operating, and deleting Kubernetes clusters in the cloud. A rolling upgrade of an older version of Kubernetes to a new version can also be performed.

Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management. Google originally designed Kubernetes, but the Cloud Native Computing Foundation now maintains the project.

To build a good DockerFile: https://docs.docker.com/engine/reference/builder/

More Interview Question: https://webmagicinformatica.com/interview-questions/