**100 Kubernetes Interview Questions That Will Land You Any Kubernetes Job**

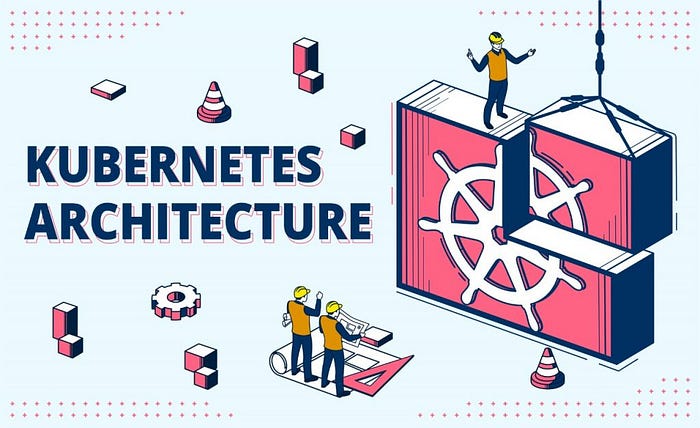
This blog post is a collection of Kubernetes interview questions that will help you get the job of your dream. Make sure you memorize a short answer and can provide a detailed explanation in your own words.

Our goal was to make this guide as technical and detailed as possible. We assume you already know what Kubernetes is, are familiar with general ideas of containerization, and understand the benefits of these technologies. Therefore, we omit generic questions and dive right into detailed, practical, and low-level questions, which are more likely to be asked during a Kubernetes interview.

You can use this as a roadmap for your interview preparation or as a last-minute knowledge check.

Expand to reveal the answer and further related questions 👇👇👇

**Part 1 — Kubernetes Architecture**



**What is the latest version of Kubernetes that is generally available?**

As of September 2022, the latest version of Kubernetes is 1.25, also called “Combiner”.  
However, it is crucial to understand that Kubernetes is still in active development, and new versions are released every several months.

**What is the Kubernetes control plane?**

The control plane components work together to coordinate the activities of a Kubernetes cluster. The control plane is responsible for managing the cluster’s state, ensuring that its desired configuration is always maintained.

**What is the Kubernetes data plane?**

The Kubernetes data plane is the part of a cluster responsible for running the workloads. In other words, all nodes in a cluster belong to a data plane.

**What are the main Kuberenets components?**

For a complete and working Kubernetes cluster, you will need the following components:

* API Server;
* etcd;
* kubelet;
* kube-proxy;
* Scheduler;
* Controller Manager;
* Cloud Controller Manager;
* Container Runtime.

**What is the API Server?**

The API Server is the heart and soul of the Kubernetes control plane. It exposes the Kubernetes API and acts as a gateway, providing the API for all other components and external clients.

**What is etcd?**

Etcd is a key-value store that is being used for holding configuration data, state data, and metadata. Kubernetes uses etcd to save information such as the list of nodes in the cluster, the desired state of all workloads that are running in the cluster, and cluster events.

**What is kubelet?**

Kubelet is a Kubernetes agent that runs on each node in a Kubernetes cluster. Its role is to ensure that all pods scheduled on the node are running and healthy.

**What is kube-proxy?**

Kube-proxy is a networking service that runs on each node in a Kubernetes cluster. It is responsible for maintaining network rules on the node and provides services such as port forwarding and load balancing for Kubernetes services.

**What is scheduler?**

The scheduler is a Kubernetes control plane component that assigns newly created pods to nodes in a cluster. It tracks various parameters such as resource requirements, hardware constraints, and affinity rules, taking them into account during scheduling decisions.

**What is Controller Manager?**

The Controller Manager is a Kubernetes component that manages controllers.

**What is Controller?**

The Kubernetes controller is an infinite loop that watches the state of the API object and makes required changes ensuring that its observed state matches the desired state.

**What is Cloud Controller Manager?**

Cloud Controller Manager is a dedicated Kubernetes component that manages cloud-specific controllers.

**What is Container Runtime?**

A container runtime provides the low-level components needed to launch and run containers. It sets up the underlying kernel primitives, such as namespaces and control groups, and provides a high-level API for container management.

**What container runtimes are supported by Kubernetes?**

The following Container runtimes are supported by Kubernetes:

* containerd;
* CRI-O;
* Docker Engine;
* Mirantis Container Runtime;

**Which Kubernetes components are considered Control Plane components?**

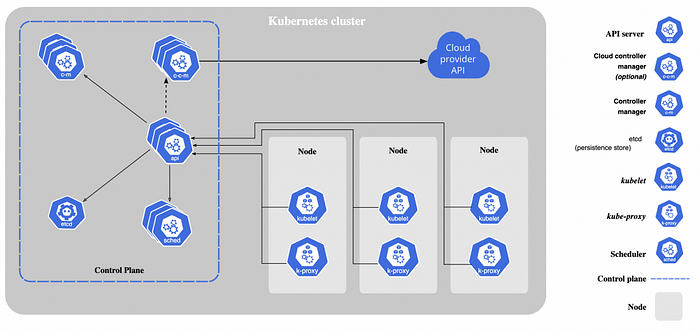
The Kubernetes Control plane components are:

* API Server;
* etcd;
* Scheduler;
* Controller Manager;
* Cloud Controller Manager;

**Which Kubernetes components are considered Data Plane components?**

The Kubernetes Data Plane components are:

* kubelet;
* kube-proxy;
* Container Runtime;Draw a diagram of Kubernetes components and interactions between them.



**Part 2 — Kubernetes API**

**Which Kubernetes component is responsible for providing access to Kubernetes API?**

The component responsible for providing access to Kubernetes API is called API Server.

**What are the different ways we can interact with the API server?**

We can interact with Kubernetes API through the following tools:

* kubectl;
* rest API using kubectl proxy;
* rest API by passing an authentication token directly;
* official client libraries.

**What will happen with a cluster if the API server is down?**

The current workloads will continue to run. However, neither clients nor internal Kubernetes components would be able to interact with the Kubernetes API and record any changes in the state of a cluster.

**What are the main Kubernetes API objects?**

The main Kubernetes API objects are:

* Pod
* ReplicaSet
* Deployment
* StatefulSet
* DaemonSet
* Service
* Ingress
* ConfigMap
* Secret
* PersistentVolume
* PersistentVolumeClaim
* StorageClass
* Namespace
* ServiceAccount
* Role
* RoleBinding
* ClusterRole
* ClusterRoleBinding

**What is Kubernetes Pod?**

A Pod is the thinnest Kubernetes abstraction. It consists of one or more containers that share storage and network settings with each other. Containers within a pod always run together on the same node with a shared context.

**What is Kubernetes ReplicaSet?**

A ReplicaSet is a group of identical pods with the same label selector. A ReplicaSet ensures that a specified number of pod replicas are running at any given time. If a pod dies, the ReplicaSet automatically replaces it with a new one.

**What is Kubernetes Deployment?**

A deployment is a Kubernetes object that encapsulates a ReplicaSet and provides declarative updates (i.e. desired state management) for Pods.

**What is the difference between a ReplicaSet and Deployment?**

The main difference between a Deployment and a ReplicaSet is that a Deployment manages a ReplicaSet, and a ReplicaSet only manages Pods.

**What is Kubernetes StatefulSet?**

A StatefulSet is a special type of Deployment that manages a set of identical Pods that are scaled together. Pods in a StatefulSet have a fixed identity(pod name) that is maintained across restarts.

**What is Kubernetes DaemonSet?**

A DaemonSet runs a copy of a Pod on each node of a cluster. When a node is added to the cluster, the DaemonSet schedules a Pod there. When a node is removed, those Pods are deleted.

**What is Kubernetes Namespace?**

A namespace is a logical entity used to segregate resources within a Kubernetes cluster. Each namespace is isolated from other namespaces and provides a unique scope for resources within the cluster.

**Provide examples of Kubernetes resources that are Namespaced and Kubernetes resources that are not Namespaced**

Kubernetes resources that are namespaced:

* Pods
* ReplicationControllers
* Deployments
* Services
* Secrets
* ConfigMaps
* Roles
* RoleBindings
* PersistentVolumeClaims

Kubernetes resources that are not namespaced:

* Nodes
* Events
* PersistentVolumes
* ClusterRoles
* ClusterRoleBindings

**What are the top-level properties of a Kubernetes API object?**

The top-level properties of a Kubernetes API object are:

* apiVersion;
* kind;
* metadata;
* spec;
* status;

**What is the apiVersion field?**

Kubernetes apiVersion is a string that specifies the API version to use

**Provide examples of apiVersions that you are aware of.**

The examples of apiVersions are:

* apps/v1
* batch/v1
* v1
* alpha

**What is the kind field?**

Kubernetes API kind represents an object in the Kubernetes system; for example, Deployment is a kind. Each object kind has a different set of attributes and behaviors.

**What is the metadata field?**

Kubernetes metadata is data that describes Kubernetes objects. Metadata often store information that is not part of the object’s core properties.

**What metadata fields do you know?**

The examples of the metadata fields are:

* name;
* namespace;
* UID;
* resourceVersion;
* generation;
* creationTimestamp and deletionTimestamp;
* labels;
* annotations.

**What is the UID field?**

The UID is an identifier that is used to distinguish between objects with the same name that have been deleted and recreated.

**What is the resourceVersion field?**

The resourceVersion field is used by clients to find out when objects have changed.

**What are the creatonTimestamp and the deletionTimestamp fields?**

The creationTimestamp and deletionTimestamp are fields representing the date when the object was created or deleted.

**What is the labels field?**

Labels are key-value pairs that are used to organize and categorize objects.

**What is the annotations field?**

Annotations are the key-value fields that are used by external tooling to store and retrieve arbitrary metadata about the object.

**What is the spec field?**

The spec is an API field containing information about a Kubernetes object’s desired state.

**What is the status field?**

The status field indicates the current status of the resource. For example, a pod’s status might include information about its current phase, what containers are running in the pod, and the pod’s IP address.

**What is the Kubernetes API verb?**

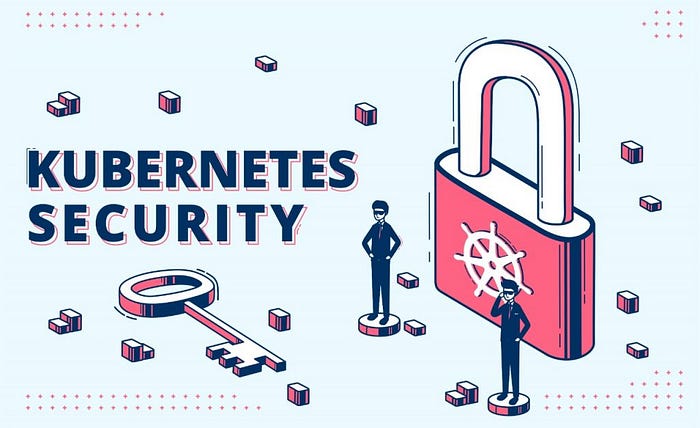
The Kubernetes API verb is a set of operations that can be performed on a Kubernetes resource.

**What Kubernetes API verbs exist?**

Kubernetes supports the following API verbs:

* get
* create
* apply
* update
* patch
* delete
* proxy
* list
* watch

**Part 3 — RBAC and Security**



**What does Kubernetes RBAC mean?**

The RBAC stands for Role-Based Access Control. This mechanism allows you to manage permissions for users and groups within a Kubernetes cluster. With RBAC, you can define who has access to what and under which conditions.

**What RBAC components exist in Kubernetes?**

There are the following components of Kubernetes RBAC:

* Role;
* RoleBidning;
* ClusterRole;
* ClusterRoleBinding.

**What is the Kubernetes Role?**

A role is a definition of a set of permissions that can be applied to one or more users or service accounts.

**What is the Kubernetes RoleBinding?**

A RoleBinding is an object that allows you to bind a role or a cluster role to a user or a service account.

**What is the Kubernetes ClusterRole?**

A ClusterRole is the same as a role but cluster-wide. Permissions assigned via cluster role are not limited by any namespace.

**What is the Kubernetes ClusterRoleBinding?**

A ClusterRoleBinding is the same as a role, but cluster-wide and used with cluster roles.

**What is AAA in Kubernetes?**

The AAA in Kubernetes stands for:

* Authentication;
* Authorization;
* Admission.

**What is Authentication?**

Authentication is the procedure of verifying that someone is who they say they are.

**What is Authorization?**

Authorization is verifying that someone has the right to perform an action or access a resource.

**What is Admission?**

In Kubernetes, Admission is a process of validating and admitting a resource into the cluster. Admission controllers are Kubernetes components that intercept requests to the Kubernetes API before the persistence of the object in etcd.

**Which kubectl command can determine if the current user can perform a given action without invoking the API action itself?**

“kubectl auth can-i” is a command that will check if the user has permission to perform an action on a resource.

**What is ServiceAccount?**

A ServiceAccount is an account used by a workload to interact with the Kubernetes API. Service accounts give processes the ability to perform a certain set of actions within a cluster.

**What is SecurityContext?**

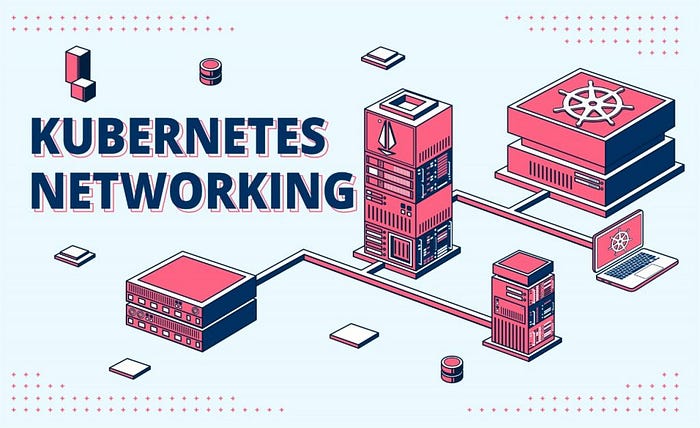
A SecurityContext is a setting that defines security settings, privilege, and access control settings for a Pod.

**What settings can you specify via SecurityContext?**

Security context settings include:

* User Id and Group Id;
* SELinux settings;
* Container privileges;
* Linux capabilities.

**Part 4 — Kubernetes Networking**



**What is Kubernetes Service?**

Kubernetes Services are a way to expose an application running on a set of Pods via a network. A service defines a collection of pods and a collection of networking rules for accessing them.

**What types of Services exist in Kubernetes?**

The following types of Services exist in Kubernetes:

1. ClusterIP;
2. NodePort;
3. LoadBalancer;
4. ExternalName.

**What is the ClusterIP service?**

The ClusterIP service is a service that exposes an application running on a set of Pods to other applications within the same cluster.

**What is the NodePort service?**

The NodePort service is a ClusterIP service with a manually specified port (the NodePort) that is opened on every node in the cluster. When a client requests a NodePort service, the request is automatically routed to one of the nodes in the cluster, and that node proxies the request to the Service.

**What is the LoadBalacner service?**

The LoadBalancer service makes a service accessible to external clients. When created, Kubernetes creates an external load balancer (if the cloud provider supports it) and maps the service to the load balancer.

**What is the ExternalName Service?**

The ExternalName service maps a service to an external DNS name. This can be used to access services such as an external database or an API.

**What is Kubernetes Ingress?**

An Ingress is an API object responsible for routing external traffic to services within a Kubernetes cluster.

**What is an Ingress controller?**

An ingress controller is a set of resources running inside a Kubernetes cluster and implementing a backend for ingress functionality.

**What popular Kubernetes Ingress controllers do you know?**

Some examples of popular ingress controllers are:

* Contour
* Nginx Ingress controller
* Ambassador
* AWS Load Balancer controller

**What is CNI?**

The CNI, or a Container Network Interface, is a set of plugins used for configuring network interfaces in Kubernetes pods. CNI is the operating system component that routes the actual network request to a container.

**What CNI do you know?**

The most popular CNI are:

* Flannel
* Calico
* Cillium
* Weave Net
* Canal

**What is NetworkPolicy?**

The network policy is an API object that defines how Pods and Services are allowed to communicate with each other and other the network.

**What CNI drivers support NetworkPolicy?**

The following CNI drivers support network policy:

* Calico
* Cillium
* Weave Net
* Canal

The following CNI driver does not support network policy:

* Flannel

**What is the “pause” container and what’s its purpose?**

The “pause” container is a special container that is not visible via Kubernetes API. It holds the network namespace for the pod. Kubernetes creates pause containers to acquire the respective pod’s IP address and set up the network namespace for all other containers that join that pod. Due to the pause container, crashed and restored containers within a Pod maintain the same IP address.

**What is a sidecar proxy?**

A sidecar proxy is a container that is deployed alongside a primary container providing it with additional networking functionality. It intercepts all the network requests sent by the primary container and provides features such as encryption, request routing, and observability. A sidecar proxy is one of the main building blocks for a service mesh.

**What is a service mesh?**

A service mesh is an infrastructure layer that allows you to manage communication between microservices. It handles things like routing, load balancing, and service discovery, and can provide features like observability, security, and resiliency.

**What popular service meshes do you know?**

The most popular service meshes are:

* Istio;
* Linkerd;
* Consul.

**Part 5 — Configuration, Deployment, and Resource Management**

**What are the main options for passing configuration information into a Kubernetes Pod?**

The main options for passing configuration information into a Kubernetes Pod are:

* ConfigMap
* Secret

**What is ConfigMap?**

A ConfigMap is an API object that acts as a dictionary of key-value pairs that store configuration data for your application.

**What is Secret?**

A Secret is an API object similar to ConfigMap but designed for storing sensitive data.

**What encryption method is used in Secret by default?**

By default, Secrets are not encrypted. Instead, they are just base64-encoded.

**How can a container inside a Kubernetes Pod find out in which namespace it is running without interacting with Kubernetes API?**

You can use downward API to inject any information about a Pod into a container within this pod. This includes things like:

* name;
* namespace;
* labels;
* any other metadata or spec field.

**What are the main ways of managing resources assigned to a container?**

The main ways of managing resources assigned to a container are:

* Requests and limits
* Limit ranges

**What parameters can we control via requests and limits?**

We can use requests and limits to control:

* CPU;
* Memory;
* Hugepages;

**What are resource requests?**

Resources specified in the requests field must be reserved for a container. The kube-scheduler uses this information to decide which node to place the Pod on. The Pod will be scheduled only if a Node has enough resources to accommodate the request.

**What are resource limits?**

When limits are specified, a container is not allowed to use more resources than what is set in the limit.

**What is the QoS class in Kubernetes?**

Kubernetes evaluates requests and limits settings and assigns a so-called Quality of Service class or QoS.

**What QoS classes exist in Kubernetes?**

The following QoS classes exist in Kubernetes:

* Guaranteed
* Burstable
* BestEffort

**What is the Guaranteed QoS class?**

The Guaranteed QoS class will be assigned if:

* Every Container in the Pod has memory limits and memory requests.
* Every Container in the Pod has memory limits equal to memory requests.
* Every Container in the Pod has CPU limits and CPU requests.
* Every Container in the Pod has CPU limits equal to CPU requests.

**What is the Burstable QoS class?**

The Burstable QoS class will be assigned if:

* The Pod does not fall under the criteria for QoS class Guaranteed.
* At least one Container in the Pod has a memory or CPU request or limit.

**What is the BestEffort QoS class?**

The BestEffort QoS class is assigned if Containers in the Pod don’t have any memory, CPU limits, or requests.

**What are Limit ranges?**

A limit range is a setting that is applied on a namespace level and enforces additional resource usage constraints.

**What parameters can we control via limit ranges?**

The LimitRange allows to:

* Constrain on a minimum and maximum resource usage per Pod or Container in a namespace.
* Constrain on a minimum and maximum storage request per PersistentVolumeClaim in a namespace.
* Enforce a ratio between request and limit for a resource in a namespace.
* Set default request and limit for resources in a namespace and automatically inject them to Containers.

**What happens if a container is trying to request more CPU or memory than is available on a cluster?**

A pod will hang in a pending state unless the requested resources become available.

**What happens if a container is trying to use more CPU than the limit assigned to it?**

A container will be throttled if it tries to use more CPU than the limit assigned to it.

**What happens if a container is trying to use more memory than the limit assigned to it?**

Currently, there is no way to throttle memory usage. This means a container will be killed with the OOMKilled status and restarted.

**What are the most popular solutions for managing Kubernetes manifests?**

The most popular solutions for managing Kubernetes manifests are:

* helm
* kustomize
* jsonnet

**What is helm?**

Helm is a Kubernetes package manager that allows configuring and managing Kubernetes applications more easily.

**What is kustomize?**

kustomize is a tool for customizing Kubernetes resources. It can be used to create, edit, and delete resources and to apply patches.

**What is jsonnet?**

jsonnet is a tool for generating JSON files, including Kubernetes manifests.

**What is GitOps?**

GitOps is a practice for managing software deployments and infrastructure changes using git repositories. GitOps involves using a git repository as the source of truth for both your application code and your infrastructure code. This approach allows you to use the same tools and workflows for managing both codebases and makes it easy to track changes and rollbacks.

**What are the most popular Kubernetes GitOps tools?**

The most popular GitOps tools are:

* ArgoCD
* FluxCD

**Part 6 — Scheduling**

**What does the term scheduling mean in Kubernetes?**

Kubernetes scheduling refers to the process of allocating Pods to nodes in a Kubernetes cluster. This process is responsible for ensuring that all Pods in a are scheduled to run on available nodes.

**What component is responsible for scheduling workloads on a Kubernetes cluster?**

The Kubernetes component responsible for scheduling Pods is the Scheduler.

**Please explain how does Scheduler work**

After a Pod is created, the scheduler determines which Node has enough resources to run the Pod. After that, the scheduler modifies the nodeName field of a Pod specification. After that, kubelet on a selected Node picks up a manifest and creates all the containers specified in a Pod.

**What will happen if the Scheduler fails?**

Existing workloads will continue to run, while new Pods will remain unscheduled.

**How can you instruct a Pod to run on a particular Node?**

You can instruct a Pod to run on a particular Node by using any of these:

* node selector with node labels;
* node affinity;
* the nodeName field;
* Pod topology spread constraints;

**How does node selector work?**

You can add the nodeSelector field to the definition of your Pod and provide the labels you want the target to have. Kubernetes only schedules the Pod to nodes that have each of the labels you specify.

**How does node affinity work?**

Node affinity is very similar to nodeSelector. It also allows you to constrain nodes your Pod can be scheduled on based on node labels, but a little more flexible.

**How does nodeName work?**

The nodeName field is a more direct way of node selection. If the nodeName field is not empty, the scheduler ignores the Pod, and the kubelet on the named node tries to place the Pod on that node. Using nodeName overrides using nodeSelector or affinity and anti-affinity rules.

**How to prevent a Pod from running on a particular Node?**

You can use taints to prevent certain Pods from being scheduled on a Node

**How can you instruct a Pod to ignore taints?**

You can use tolerations to instruct a Pod to ignore taints.

**How can you instruct two certain Pods to run on the same Node or, opposite, to never land on the same Node?**

To prevent two pods from running on the same Node, you can use podAntiAffinity.  
To instruct two Pods to run on the same Node, you can use podAffinity.

**What types of affinity rules exist in Kubernetes?**

For both nodeAffinity and podAffinity, there are two types of affinity rules:

* requiredDuringSchedulingIgnoredDuringExecution
* preferredDuringSchedulingIgnoredDuringExecution

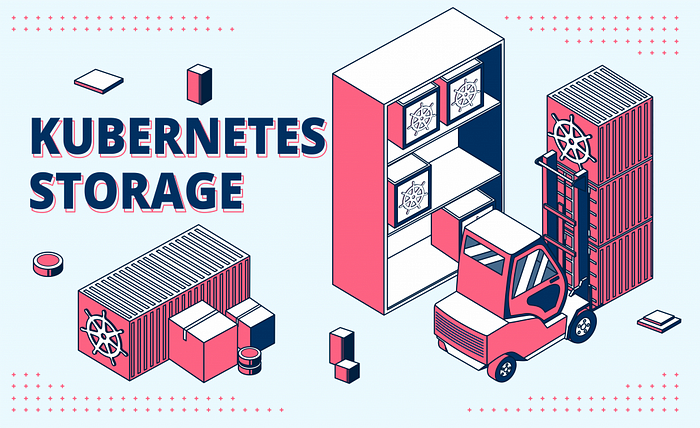
**What does requiredDuringSchedulingIgnoredDuringExecution mean?**

The hard rule. The scheduler can’t schedule the Pod unless it is satisfied.

**What does preferredDuringSchedulingIgnoredDuringExecution?**

The soft rule. The scheduler will attempt to find a node that meets the criteria. If no matching nodes are available, the scheduler still schedules the Pod.

**Kubernetes Interview Questions — Part 7 — Storage**



**What storage options exist in Kubernetes?**

The storage abstractions of Kubernetes are:

* Volume
* PersistentVolume
* PersistentVolumeClaim
* StorageClass

**What is Kubernetes Volume?**

A volume is a directory that stores data used by a pod. Volumes are attached to pods and exist as long as the pod does. A single Kubernetes volume can be shared by multiple containers within a single Pod.

**What is Kubernetes PersistentVolume?**

A PersistentVolume is an API object that represents a piece of storage in the cluster.

**What is Kubernetes PersistentVolumeClaim?**

A PersistentVolumeClaim is a request for storage created by a user. It is similar to a Pod — Pods consume node resources, and PVCs consume PV resources.

**What are the two ways for provisioning a Persistent Volume?**

There are two ways PVs may be provisioned: static provisioning and dynamic provisioning.

**What is static provisioning?**

A cluster administrator creates persistent volumes manually.

**What is dynamic provisioning?**

The cluster will dynamically provision a volume exclusively for the persistent volume claim using a storage class if none of the static persistent volumes match a PersistentVolumeClaim.

**What is Kubernetes StorageClass?**

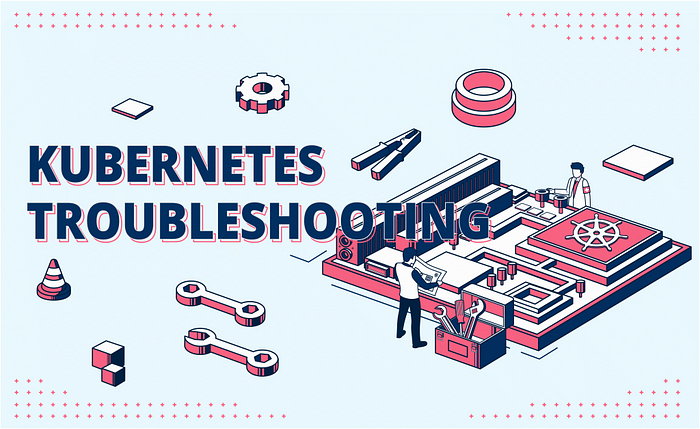
A StorageClass provides a way to define the “class” of storage they want for their applications. Which is later can be used for dynamic provisioning — creating a PersistentVolumeClaim without having to manually create a PersistentVolume.

**What types of Persistent Volumes do you know?**

PersistentVolume types are implemented as plugins. Kubernetes currently supports the following plugins:

* local — local storage devices mounted on nodes;
* nfs — Network File System (NFS) storage;
* cephfs — CephFS volume;
* fc — Fibre Channel (FC) storage.

**Part 8 — Troubleshooting Kubernetes**



**How can you debug a Pod which is failing?**

To debug a Pod, you can do the following:

* Use kubectl describe pod to fetch details of the pod;
* Use kubectl logs to obtain logs from the failed pod;
* Exact into a container using the “kubectl exec” command;
* Connect an ephemeral container using the “kubectl debug” command.

**How to obtain logs from a Kubernetes Pod?**

To obtain logs from a Kubernetes Pod, you can use the “kubectl logs” command.

**How to troubleshoot a Pod which hangs in a Pending state?**

There are no logs yet since the Pending container is not scheduled and hasn’t started yet. This means we can’t use “kubectl logs”. Instead, “kubectl describe” should be used.

**What does CrashLoopBackOff mean?**

The CrashLoopBackOff is an error state in Kubernetes where a container continuously restarts but never succeeds in staying up for more than a few seconds. This can be caused by a number of factors, such as a misconfigured container or a bug in the application code.

**You have successfully created a Deployment. However, no Pods were created. What could be the issue?**

That happens when a ReplicaSet is unable to create a Pod. You need to run “kubectl describe replicaset” to get more details about the reason for that issue.

**What is debug container?**

The debug container is an ephemeral container that can be attached alongside the failed container you wish to debug. This is useful when your primary container does not have utilities that are required for debugging installed in it.

**Part 9 — Kubernetes Operators**

**What is Kubernetes Operator?**

A Kubernetes Operator is a way of packaging, deploying, and managing a Kubernetes application. An Operator is essentially a human operator but embodied in a piece of software. It makes use of the Kubernetes API to automate the management of an application.

**What are the main building blocks of an operator?**

The main building blocks of an operator are:

* Custom Resource Definition and Custom Resource;
* Controller.

**What is Custom Resource?**

A custom resource is a primary option to extend the Kubernetes API. It allows adding new resource types that are not available by default in a Kubernetes cluster.

**What is Custom Resource Definition?**

A Custom Resource Definition (CRD) is a manifest that defines a custom resource.

**What is a Controller?**

A Kubernetes controller is a loop that watches the state of the API object or the Custom Resource and makes necessary changes ensuring that the observed state of the API object matches the desired one.

**What is the Kubernetes Operator Framework?**

The Kubernetes Operator Framework is a set of tools and practices for simplifying the development and distribution of Kubernetes Operators.

**What are the main components of the Kubernetes Operator Framework?**

The main components of the Kubernetes Operator Framework are:

* Operator SDK;
* Operator Lifecycle Manager;
* OperatorHUB.

**What is Operator SDK?**

A set of tools that helps to start developing an Operator project from scratch is called Operator SDK. It includes predefined APIs, some standard functions, and code generators.

**What is Operator Lifecycle Manager?**

Operator Lifecycle Manager is a tool that runs inside a Kubernetes cluster and provides functionality for interacting with Kubernetes Operators. It is mainly used for the installation and upgrading of Operators.

**What is OperatorHub?**

OperatorHub is an open catalog of publicly available operators.

**Before We Finish**

Memorizing questions without having a proper understanding isn’t usually effective. Make sure you can explain each and every concept mentioned in this article in your own words.

Also, you should never forget about practice. Focusing on learning theory and implementing it in a real environment will produce the best result.

If this list of Kubernetes interview questions helped you, please support our effort by sharing the article with your friends and colleagues. Our mission is to help as many people as possible to get a better life by landing high-paying jobs in Kubernetes, Cloud-Native, and DevOps.

You can contribute your own question and get featured by filling out the form 👇👇👇