KUBERNETES INTERVIEW QUESTIONS

Specify the uses of Pod, Replica-Set, and Deployment.

1. **Pod:** A Pod is the smallest deployable unit in Kubernetes and represents a single instance of a running process in the cluster. A pod can have multiple containers in it.
2. **ReplicaSet:** A ReplicaSet is a higher-level abstraction that ensures a specified number of replicas (identical copies) of a Pod are running at all times.
3. **Deployment:** A Deployment is a higher-level abstraction that manages ReplicaSets and provides declarative updates for Pods and ReplicaSets.

- **Rolling updates:** Deployments support rolling updates, allowing you to update the application without downtime by gradually replacing old Pods with new ones.
- **Rollback:** If an update causes issues, you can easily roll back to a previously known-good version.
- **Self-healing:** In case a Pod or node fails, the Deployment controller ensures that the desired number of replicas is maintained.
- **Scaling:** Deployments allow you to scale the application up or down by adjusting the number of replicas.

Q2: I have thousands of pods and lots of applications running on the pods which deployment strategy you will follow rolling or re-create? and why?

1. **Rolling Updates:**

- A rolling update strategy gradually replaces instances of the old application with instances of the new version. This means that your application remains available throughout the update process, as the new pods are slowly introduced and the old ones are terminated.

2. Re-create Strategy:

- In the re-create strategy, all existing pods with the old version of the application are terminated before the new version is deployed. This means there will be a brief downtime during the update process.

Q3: What are the components of k8s?

- **Master Node:** The master node is responsible for managing the cluster and orchestrating its components. It consists of several components
- **Kube API Server:** The central control point and the front end for the Kubernetes API.
- **etcd:** A distributed key-value store that stores the cluster's configuration data and the current state.
- **Scheduler:** The component that assigns newly created pods to nodes based on resource requirements and constraints.
- **Controller Manager:** A set of controllers that handle various aspects of the cluster, such as node and replica management, endpoints, services, etc.
- **Worker Nodes:** These are the machines where containerized applications (pods) are deployed and run. Each worker node runs the following components
- **Kubelet:** The primary agent that runs on each node and communicates with the master. It is responsible for starting, stopping, and monitoring containers as instructed by the master.
- **Container Runtime:** The software responsible for running containers, such as Docker, containerd, or CRI-O.
- **kube-proxy:** Manages network connectivity for pods and services on the node.
- **Pods:** The basic building blocks in Kubernetes. A pod is the smallest and simplest unit in the Kubernetes object model, representing one or more containers that are deployed together on the same host and share the same network namespace.
- **ReplicaSets:** A higher-level abstraction that ensures a specified number of replicas (identical copies) of a pod are running at all times.
- **Services:** An abstraction that defines a stable endpoint (IP address and port) to connect to a group of pods, allowing them to be accessed via a single IP address and load-balanced if multiple replicas exist.
- **Volumes:** A way to provide persistent storage to containers within pods.
- **Namespaces:** A virtual cluster within a Kubernetes cluster, used to divide resources and create isolation between different applications or teams.
- **ConfigMaps and Secrets:** ConfigMaps store configuration data, while Secrets store sensitive information like passwords or API keys. Secrets are always in the base64 encoding format. Both can be used to decouple configuration from container images.
- **Deployment:** A higher-level abstraction that allows declarative updates to pods and replica sets.

- **StatefulSets:** An extension of ReplicaSets to handle stateful applications with unique identities and stable network identities.
- **DaemonSets:** Ensures that all or some nodes run a copy of a pod, typically used for system daemons.

Q4: What is PV & PVC?

1. **PV: Persistent Volume**

- A Persistent Volume (PV) in Kubernetes is a piece of storage in the cluster that has been provisioned by an administrator. It is a way to abstract the underlying storage technology and provide a unified interface for applications to request and use storage resources. PVs can be thought of as a “claimable” resource by pods.

2. PVC: Persistent Volume Claim

- A Persistent Volume Claim (PVC) is a request for storage by a user or application. It allows users to request specific storage resources (size, access mode, etc.) without needing to know the details of the underlying storage technology. PVCs provide a way for pods to dynamically request and use storage volumes from the available PVs.

Q5: What is taint & toleration in k8s?

- **Taint:** A taint is a label applied to a node in the Kubernetes cluster, indicating that the node has a specific attribute or constraint that affects the types of pods that can be scheduled on it. Taints are typically used to prevent pods from being scheduled on specific nodes unless those pods have corresponding tolerations.
- **Toleration:** A toleration is a property defined in a pod’s configuration that allows the pod to be scheduled on nodes with specific taints. When you specify a toleration for a pod, you are essentially telling Kubernetes that the pod is willing to tolerate nodes with the specified taints. Pods without appropriate tolerations will not be scheduled on nodes with matching taints.

Q6: What is the label & selector in k8s?

- **Labels:** Labels are key-value pairs that you can attach to Kubernetes objects, such as pods, nodes, services, and more. They are used to mark resources with metadata that provides additional information about the resource.

- **Selectors:** Selectors are used to efficiently and flexibly identify a group of resources based on their labels. Selectors allow you to filter and target specific resources that match certain criteria.

Q7: What is Node Affinity in k8s?

- **Node affinity** refers to a feature that allows you to specify rules or preferences for the scheduling of pods onto nodes (worker machines) within a cluster. It lets you control where pods are placed based on node properties, such as labels or other node attributes.
- **Hardware Requirements:** If certain nodes in your cluster have specialized hardware (e.g., GPUs) and your pods require that hardware, you can use node affinity to ensure those pods are scheduled only on nodes with the required hardware.
- **Topology Constraints:** You might want to ensure that pods are distributed across different availability zones, regions, or data centers for high availability. Node affinity can be used to spread pods across such topologies.
- **Co-locating Pods:** You might have pods that should run on the same node for performance or communication reasons. Node affinity can help ensure these pods are scheduled together.

Q8: What are the different services within Kubernetes?

1. **Cluster IP service:**

- The ClusterIP is the default Kubernetes service that provides service inside a cluster (with no external access) that other apps inside your cluster can access.

2. Node Port service:

- The NodePort service is the most fundamental way to get external traffic directly to your service. It opens a specific port on all Nodes and forwards any traffic sent to this port to the service.

3. Load Balancer service:

- The LoadBalancer service is used to expose services to the internet. A Network load balancer, for example, creates a single IP address that forwards all traffic to your service.

Q9: What is a headless service?

- Headless services are particularly useful in scenarios where you need to discover and interact with individual pods within a Kubernetes service without going through a load balancer.
- This is where Headless service allows us to get all the pod's IP addresses through DNS lookup.

Q10: What is Minikube?

- The Minikube makes it easy for the local running of Kubernetes. Within a virtual machine, the Minikube runs a single-node Kubernetes cluster.

Q11: What is load balancing on Kubernetes?

- The process of load balancing will let us expose services. There are two types of load balancing when it comes to Kubernetes:
- **Internal load balancing:** This is used for balancing the loads automatically and allocating the pods with the required configuration.
- **External load balancing:** This directs the traffic from the external loads to the backend pods.

Q12: What is crashloofbackoff state in a pod?

- CrashLoopBackOff is a Kubernetes state representing a restart loop happening in a Pod: a container in the Pod is started, but crashes and is then repeatedly restarted. CrashLoopBackOff is not an error in itself but indicates that there's an error happening that prevents a Pod from starting correctly.

Q13: What is the Pending state in the Pod?

- If a Pod is stuck in Pending it means that it can not be scheduled onto a node.

Q14: What is Readiness & Liveness Probe?

1. **Liveness Probe**

- Suppose that a Pod is running our application inside a container, but due to some reason let’s say memory leak, CPU usage, application deadlock, etc the application is not responding to our requests, and is stuck in an error state.
- The liveness probe checks the container's health as we tell it to, and if for some reason the liveness probe fails, it restarts the container. We can define liveness probe in 3 ways

2. Readiness Probe

- In some cases, we would like our application to be alive, but not serve traffic unless some conditions are met e.g, populating a dataset, waiting for some other service to be alive, etc.
- In such cases, we use a readiness probe. If the condition inside the readiness probe passes, only then our application can serve traffic.

Q15: Deployment vs Statefulsets

1. **Deployment:**

- Deployments are best suited for stateless applications, where each instance of the application is identical and can be easily replaced without concerns about its identity or data.
- When a pod that is managed by k8s gets deleted the new pod will come up with the new ID
- Deployment provides rolling updates & roll back feature and pods can easily be rolled out

2. Satetfulsets:

- StatefulSets are designed for managing stateful applications, where each pod instance has a unique identity and possibly persistent data.
- When a pod that is managed by k8s gets deleted the new pod will come up with the same ID
- StatefulSets are more complex to update compared to Deployments, as you have to consider data migration, ordering, and potential disruption.

Q16: What is cordon in k8s?

- In Kubernetes (k8s), a "cordon" refers to a feature that is used to mark a node as "unschedulable." When you cordon a node, it means that no new pods will be scheduled onto that node by the Kubernetes scheduler.

Q17: What is draining of node in k8s?

- In Kubernetes (often abbreviated as K8s), a “node” refers to a worker machine in the cluster that runs containerized applications. These nodes are responsible for executing tasks and managing the containers. The process of “draining” a node in Kubernetes refers to safely evicting or relocating the workloads (containers) running on a node to other nodes in the cluster. This is

typically done when you need to perform maintenance on the node, upgrade it, or decommission it.

Q18: What is the ingress controller in k8s?

- **Path and Host-Based Routing:** You can configure the Ingress Controller to route traffic to different Services based on the URL path or the host header of the incoming request.
- **Load Balancing:** The Ingress Controller can distribute incoming traffic across multiple instances of a Service, helping to ensure high availability and efficient resource utilization.

Q19: Can we put multiple containers inside a pod?

- Yes, a pod can contain multiple containers of a similar kind

Q20: Can 2 pods on the same node communicate with each other?

- If two pods are scheduled on the same node, Pods use virtual bridges to communicate with each other.

Q21: How to check the deprecated API version in K8s while upgrading the cluster?

- You can use kubectl commands to list the deprecated API versions in your cluster
- **kubectl api-versions | grep -i deprecated**
- From the K8s docs also you can check
- Starting from Kubernetes 1.19, you might receive deprecation warnings when using deprecated APIs.

Q22: What is the difference between NodePort & Loadbalancing service?

- A NodePort service exposes a specific port on all the nodes in the cluster.
- It doesn't provide automatic load balancing across nodes.
- Typically used for small-scale clusters or testing purposes.
- A LoadBalancer service is designed for production environments and provides built-in load balancing.
- It requests a cloud provider to allocate an external load balancer, distributing traffic across multiple nodes.
- External clients can access the service using the IP address of the load balancer.
- The load balancer distributes traffic to the nodes where the service is running.

Q23: What is the pod disruption budget?

- If you have a deployment with a replica count of 5 pods and you set a Pod Disruption Budget of 2, it means that during maintenance or other disruptions, only a maximum of 2 pods can be simultaneously unavailable. This guarantees that the application remains available and responsive to user requests to the extent specified by the budget.

Q24: How many maximum pods you can schedule on a node?

- You can schedule up to 110 pods on the single node

Q25: What are Kubernetes network policies?

- Kubernetes network policies are rules that control the flow of network traffic between pods and services within a Kubernetes cluster.

Q26: What is Kubernetes logging?

- Kubernetes logging is the process of collecting and analyzing the logs generated by the applications and services running in a Kubernetes cluster.

Q27: What is Kubernetes scheduling policy?

- Kubernetes scheduling policy is a set of rules and criteria used to determine which node in the cluster should run a specific pod.

Q28: What is a container runtime in Kubernetes?

- A container runtime is responsible for starting and stopping containers on a node. Examples include Docker, containerd, and CRI-O.

Q29: If the scheduler is down then will my pods get scheduled?

- No, if the scheduler in a Kubernetes cluster is down, new pods will not be scheduled automatically.
- However, if pods were already scheduled and running before the scheduler went down, they will continue to run as long as the underlying nodes are operational and healthy.

Q30: What is a service account in K8s?

- A Service Account in Kubernetes (often abbreviated as "SA") is a mechanism used to provide an identity and set of permissions for applications (pods) running within a Kubernetes cluster. Kubernetes uses Service Accounts to

control the level of access and authorization that pods have when interacting with the Kubernetes API server or other cluster resources.

- By default, every namespace in a Kubernetes cluster has a default Service Account named "default." If no specific Service Account is specified for a pod, it will use this default Service Account.
- In addition to the default Service Account, you can create custom Service Accounts within a namespace. This allows you to define specific permissions for different groups of pods.

Q31: What is requests & limit in K8s?

- The "requests" setting defines the minimum amount of resources (such as CPU and memory) that a pod requires to run.
- The "limits" setting specifies the maximum amount of resources a container is allowed to consume.

Q32: How can I give different IPs to different namespaces in K8s?

- In Kubernetes, each pod runs in its own IP address, and namespaces provide a way to isolate resources within a cluster. By default, pods within a namespace share the same IP address range as the cluster.
- If you want to allocate different IP address ranges to different namespaces, you can achieve this through the use of a Container Network Interface (CNI) plugin that supports IP allocation customization.
- Calico is the most common CNI plugin used for that
- We can install the calico and create the IP pool resource using that and specify the IP address range in that and using the namespaceselector we can attach that pool to the namespace.

Q33: How to scale the AWS EKS worker nodes?

- Modify the desired, minimum, or maximum number of nodes in the node group as per your requirements. (Manual way)
- Create an auto-scaling group and link it to your EKS cluster's node group. Set up scaling policies based on metrics such as CPU utilization or custom CloudWatch metrics. As workload demand changes, the Auto Scaling group will automatically adjust the number of instances in the node group based on the defined scaling policies.

Q34: Difference between create and apply in K8s?

- The main difference between create and apply is that create will always create a new resource, even if a resource with the same name already exists in the cluster. This means that if you use Create to apply a YAML file multiple times, it will create multiple resources with the same name, causing conflicts and errors.
- On the other hand, apply will create a new resource if it does not exist, but will modify an existing resource if it does. This means that if you use apply to apply a YAML file multiple times, it will modify the existing resource rather than create a new one.
- In summary, create should be used for initial resource creation, while apply should be used for updates and modifications to existing resources.

Q35: What is a multinode cluster and single-node cluster in Kubernetes?

- A single-node cluster is a Kubernetes cluster that runs on a single machine. It is mostly used for testing and development purposes where a single machine is used to run and manage Kubernetes.
- For example, you can use Minikube to run a single-node Kubernetes cluster on your local machine for testing and development purposes.
- On the other hand, a multinode cluster is a Kubernetes cluster that runs on multiple machines or nodes. It is used in production environments where a large number of containerized applications need to be deployed and managed.
- For example, if you want to deploy a web application that receives a large number of requests, you can use a multi-node cluster to distribute the load across multiple nodes, ensuring high availability and scalability.

Q36: Can you explain the concept of self-healing in Kubernetes and give examples of how it works?

- Self-healing is an important concept in Kubernetes, which allows the system to automatically detect and recover from failures in the cluster. This ensures that the applications running in the cluster are always available and responsive to user requests.
- One of the ways in which Kubernetes achieves self-healing is through replication. By creating multiple replicas of a pod, Kubernetes can ensure that if one replica fails, the other replicas can continue to serve traffic without interruption.
- Another way in which Kubernetes achieves self-healing is through probes. Probes are used to periodically check the health of a pod by sending

requests to it and verifying that it responds correctly. If a pod fails a probe, Kubernetes can automatically terminate the pod and create a new one to take its place.

Q37: How to enable auto scaling of pods in the K8s?

- Enabling auto scaling of pods in Kubernetes (K8s) involves using the Horizontal Pod Autoscaler (HPA) resource. The HPA dynamically adjusts the number of replica pods in a deployment or replica set based on CPU utilization, memory utilization, or custom metrics.

Q38: How does Kubernetes handle storage management for containers?

- **Volumes:** Volumes in Kubernetes provide a way to persist data beyond the lifetime of a container. Kubernetes supports various types of volumes such as hostPath, emptyDir, configMap, secret, persistentVolumeClaim, etc.
- **Persistent Volume (PV)** and **Persistent Volume Claim (PVC):** PV is a piece of storage that is provisioned by an administrator, while a PVC is a request for storage by a user. PVCs can request a certain amount of storage from a PV and are bound to the PV that matches the capacity and access mode.
- **Storage Classes:** A storage class provides a way for administrators to describe different types of storage that are available in the cluster. It defines the provisioner that will be used to dynamically provision a new volume when a PVC is created.
- **StatefulSets:** StatefulSets in Kubernetes provide a way to manage stateful applications that require persistent storage. They ensure that each pod in the StatefulSet gets a unique and persistent hostname and volume.

Q39: How would you design a Kubernetes architecture for a large-scale, multi-tenant application, and what factors would you need to consider?

- We need to ensure that there is an ingress controller or a load balancer that will route the traffic to underline pods so that each and every pod gets the traffic and not the pod gets the overload.
- We also need to set up the HPA so that auto-scaling can work properly.
- You need to ensure that you can monitor and troubleshoot your architecture to detect and address issues quickly.
- We need to use the concept of Kubernetes namespaces so that the resources will get isolated from each other and the same kind of resources will remain in the same namespace.

Q40: Why is Sticky Identity Important?

- When databases scale, they add Slave Pods to only read data. This is because if two Pods write on the same data, it will create data inconsistency. Hence, one Pod is always the master node and the other slaves. The slave Pods get replicated data of the master Pod and are called Read Replicas.

Q41: Let's say my pod has 3 containers and I want to check the logs of all the containers how can I do that?

- Command: kubectl logs <pod-name> -n <namespace> –all-containers

Q42: How to restrict the access of pods from various IPs in the K8s cluster?*

- NetworkPolicy is a Kubernetes object that enables the creation of policies to restrict the communication between pods and external entities in a namespace, using various factors like IP addresses, ports, protocols, and labels.
- The ingress section defines incoming traffic rules while the egress section defines outgoing traffic rules. NetworkPolicy uses podSelector to select pods based on their labels, namespaceSelector to select pods in particular namespaces, and ipBlock to specify IP address blocks that allow or deny access to pods.

Q43: What is ClusterAutoScaler in AWS EKS?

- The Cluster Autoscaler is responsible for adjusting the size of the worker node Auto Scaling Group (ASG) in your EKS cluster based on the pending pods in the cluster. When there are more pending pods than available resources in the cluster, it scales up the number of worker nodes in the ASG, and when there are idle nodes, it scales them down to save costs.

Q44: From where KubeAPI server get all the info about pods?

- The primary source of truth for all Kubernetes cluster data, including pod information, is the etcd datastore. Etcd is a distributed key-value store that stores the configuration data of the entire cluster. When a user or a controller creates, updates, or deletes a pod, the relevant information is persisted in etcd.

- The kubelet is an agent that runs on each node in the Kubernetes cluster and is responsible for managing containers and pods on that node. Kubelet communicates with the kube-apiserver to report the status of pods on its node and to perform actions on pods as instructed by the API server. Kubelet regularly polls the kube-apiserver for updates and syncs the state of pods on its node with the desired state defined in the Kubernetes API.

Q45: I want a daemonsets to be schedule on a particular node how can I do that?

- Using nodeSelector, nodeAffinity, Taints, and Tolerations, you can restrict the daemonset to run on specific nodes.

Q46: What is the role of the kube proxy?

- One of the essential functions of kube-proxy is to enable load balancing for services in a Kubernetes cluster. When you create a Kubernetes service, kube-proxy ensures that traffic to the service is distributed among the relevant pods that back that service.
- Kube-proxy performs health checks on endpoints to ensure that traffic is directed only to healthy pods.
- Kube-proxy logs relevant events and information for debugging purposes. This can be helpful when troubleshooting networking issues within the cluster.

Q47: What will happen in thebackground when you type kubectl get pods?

- Kubectl reads your Kubernetes configuration file (usually located at ~/.kube/config) to determine which cluster to connect to and what credentials to use. It may also involve additional authentication mechanisms like tokens or certificates.
- Kubectl establishes a secure connection to the Kubernetes API server.
- The API server authenticates and authorizes the user based on the provided credentials.
- Kubectl sends an HTTP request to the Kubernetes API server, specifically a GET request to the /api/v1/pods
- The Kubernetes API server receives the request and processes it and send the data.
- kubectl receives the response from the API server and parses it.

Q48: How can you manage to roll back to a previous version of a Kubernetes Deployment?

- Retrieve the revision history of the Deployment using the kubectl rollout history command.
- Roll back to the desired revision using the kubectl rollout undo command with the appropriate revision number.
- Monitor the status of the rollback using kubectl rollout status.
- If required, pause the automatic rollout using kubectl rollout pause and resume it using kubectl rollout resume after verification.

Q49: How to check deprecated API version in k8s while upgrading cluster?

- You can use kubectl commands to list the deprecated API versions in your cluster
- **kubectl api-versions | grep -i deprecated**
- From the k8s docs
- Starting from Kubernetes 1.19, you might receive deprecation warnings when using deprecated APIs.

Q50: How can you perform a canary deployment in Kubernetes?

- Canary deployment is a deployment strategy where a new version of an application is rolled out to a small subset of users or Pods to validate its performance before rolling it out to the entire user base. In Kubernetes, you can achieve canary deployments using the following steps:
- **Create Canary Deployment:** Create a new Deployment with the new version of your application. This Deployment should have a smaller number of replicas, representing the canary group.
- **Traffic Routing:** Use Kubernetes Service with Service Type LoadBalancer or NodePort, or an Ingress resource, to route a portion of traffic to the canary Deployment while the majority of the traffic goes to the stable version.
- **Monitoring and Validation:** Monitor the performance and behavior of the canary Deployment. Use metrics, logs, and user feedback to validate the new version’s stability and correctness.
- **Gradual Promotion:** If the canary Deployment performs well, gradually increase the number of replicas in the canary group and decrease the replicas in the stable version until all traffic is routed to the new version.
- **Rollback:** In case of issues, quickly roll back by redirecting all traffic back to the stable version.

Q51: What is a container runtime in Kubernetes?

- A container runtime is responsible for starting and stopping containers on a node. Examples include Docker, containerd, and CRI-O.

Q52: What is the init container?

- An init container is a type of container in Kubernetes that runs before the main application containers in a pod. The purpose of an init container is to perform initialization tasks or setup procedures that are not present in the application container images. Examples of tasks that an init container might perform include downloading configuration files, setting up a network connection, or initializing a database schema.

Q53: Differentiate between a replica set and a replication controller.

- They are the same but differ only in using selectors to reproduce pods.The replication controller allows us to create multiple pods easily, but if a pod crashes, it ensures it is replaced with a new pod. It can scale the number of pods and update or delete multiple pods with a single command.
- The replica set is the same as the replication controller except that they have more options for the selectors. They use set-based selectors to manage the pods. Here the rolling-update command won't work.

Q54: K8s deployment best practices - how our application is more reliable, high available, and disaster recovery

- Use Kubernetes Deployments or StatefulSets to manage replicas of your application. This ensures high availability by automatically replacing failed pods and maintaining a desired number of healthy replicas.
- Use HPA to auto scale the stateless application so that based on the CPU or Memory or any other custom matrices it can auto scale the pods
- Implement readiness and liveness probes for your application. Readiness probes indicate when the application is ready to receive traffic, while liveness probes determine if the application is functioning properly. Kubernetes can use these probes to decide when to route traffic to a specific pod or when to restart a failing one.
- Define resource limits (maximum resources a container can use) and resource requests (amount of resources a container needs). This prevents resource contention and ensures fair allocation among pods.
- Use namespaces to isolate different environments, teams, or applications. This helps prevent interference between resources, simplifies management, and enhances security.

Q55: Can you explain the concept of self-healing in Kubernetes and give examples of how it works?

- Self-healing is an important concept in Kubernetes, which allows the system to automatically detect and recover from failures in the cluster. This ensures that the applications running in the cluster are always available and responsive to user requests.
- One of the ways in which Kubernetes achieves self-healing is through replication. By creating multiple replicas of a pod, Kubernetes can ensure that if one replica fails, the other replicas can continue to serve traffic without interruption.
- Another way in which Kubernetes achieves self-healing is through probes. Probes are used to periodically check the health of a pod by sending requests to it and verifying that it responds correctly. If a pod fails a probe, Kubernetes can automatically terminate the pod and create a new one to take its place.