**Kubernetes Questions**

1. What is orcherstration in Kubernetes?
2. Explain about Kubernetes architecture?
3. Explain control manager in Kubernetes?
4. Why we use Kubernetes?
5. What is pod,deployment,service,ingress,pvc and pv,stateful set, demon set, crawn job, main spaces,secrets and config maps?
6. What is manual and dynamic storage provising in Kubernetes?
7. How do you troubleshoot if the pod is not running?
8. What is replicas?
9. How do you scale the applications in the Kubernetes?

**10.**What are different services in Kubernetes?

**11.** What is difference between Ingress Controller,Ingress resources?

**12.**How can you check the memory usuage,and cpu in podsand nodes?

**13.** what is manual scheduling,taint and tolerations and what is node selector node affinity?

**14.** What is difference between retained and delete in PVC?

**15.** What are the different deployment strategies in Kubernetes?

**16.** What is blue-green deployment?

**17.** What is recreating, rolling update, kenry ?

**18.** What is difference between liveness probe,readiness probe?

**19.**What is difference between EKS,ECS in aws?

**20.** Why do we use help chats?

**21.** How did you implement logging mechanism in Kubernetes?

**22.** What is the difference between statefulset and deployment?

**23.** Have you created Kubernetes cluster?

Here are detailed answers to your Kubernetes interview questions:

**1. What is orchestration in Kubernetes?**

Orchestration in Kubernetes refers to the automated management of containerized applications, including deployment, scaling, and networking. It ensures:

Automated scaling of applications.

* Self-healing (restarting failed containers).
* Load balancing.
* Rolling updates and rollbacks.

# Kubernetes automates these tasks, reducing manual effort and improving reliability….

**2. Explain Kubernetes architecture.**

Kubernetes architecture consists of the following components:

1. **Control Plane (Master Node)**:
   * **API Server**: The gateway for all Kubernetes operations.
   * **Controller Manager**: Manages controllers (e.g., replication controller).
   * **Scheduler**: Assigns pods to worker nodes based on resources.
   * **etcd**: A distributed key-value store for cluster state.
2. **Worker Nodes**:
   * **Kubelet**: Ensures containers run properly.
   * **Kube Proxy**: Manages networking between services.
   * **Container Runtime**: Runs the containers (e.g., Docker, containerd).

**3. Explain Control Manager in Kubernetes.**

The **Kubernetes Controller Manager** is a component that runs controller processes, responsible for:

* Managing node health.
* Scaling applications.
* Handling endpoints for services.
* Managing PersistentVolumeClaims (PVCs) and PersistentVolumes (PVs).

It ensures the desired state of the cluster matches the actual state.

**4. Why do we use Kubernetes?**

Kubernetes provides:

* **Scalability**: Automatically scales applications.
* **Self-Healing**: Restarts failed containers.
* **Load Balancing**: Distributes traffic across multiple instances.
* **Rolling Updates**: Updates applications without downtime.
* **Storage Management**: Supports persistent storage..

**5. What are Pod, Deployment, Service, Ingress, PVC and PV, StatefulSet, DaemonSet, CronJob, Namespaces, Secrets, and ConfigMaps?**

| **Component** | **Description** |
| --- | --- |
| **Pod** | The smallest deployable unit, containing one or more containers. |
| **Deployment** | Manages the lifecycle of pods, including scaling and rolling updates. |
| **Service** | Exposes a set of pods as a network service. |
| **Ingress** | Manages external access to services using HTTP/HTTPS. |
| **Persistent Volume (PV)** | A storage resource provisioned for Kubernetes. |
| **Persistent Volume Claim (PVC)** | A request for storage by a pod. |
| **StatefulSet** | Manages stateful applications (e.g., databases). |
| **DaemonSet** | Ensures a pod runs on all nodes (e.g., monitoring agents). |
| **CronJob** | Runs scheduled tasks at specified times. |
| **Namespaces** | Isolate different teams or projects within a cluster. |
| **Secrets** | Store sensitive information (e.g., passwords). |
| **ConfigMaps** | Store configuration data separately from application code. |

**6. What is manual and dynamic storage provisioning in Kubernetes?**

* **Manual provisioning**: Admins manually create PersistentVolumes (PVs) before applications request storage.
* **Dynamic provisioning**: Kubernetes automatically creates PVs when a PersistentVolumeClaim (PVC) is made.

Example of **dynamic provisioning**:

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: my-pvc

spec:

storageClassName: fast-storage

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 5Gi

**7. How do you troubleshoot if a pod is not running?**

1. **Check pod status**:
2. kubectl get pods
3. **Describe the pod**:
4. kubectl describe pod <pod\_name>
5. **Check logs**:
6. kubectl logs <pod\_name>
7. **Check events**:
8. kubectl get events
9. **Execute into the pod**:
10. kubectl exec -it <pod\_name> -- /bin/sh

**8. What is Replicas?**

Replicas define how many copies of a pod should run at a time. In a Deployment:

spec:

replicas: 3

This ensures availability and load balancing.

**9. How do you scale applications in Kubernetes?**

Scale up/down using:

kubectl scale deployment my-app --replicas=5

Or define in YAML:

spec:

replicas: 5

**10. What are different services in Kubernetes?**

| **Service Type** | **Description** |
| --- | --- |
| **ClusterIP** | Default; accessible only within the cluster. |
| **NodePort** | Exposes service on a node’s port. |
| **LoadBalancer** | Uses an external load balancer. |
| **ExternalName** | Maps service to an external name (DNS). |

**11. What is the difference between Ingress Controller and Ingress Resources?**

* **Ingress Resource**: A Kubernetes object that defines routing rules.
* **Ingress Controller**: A pod that implements these rules (e.g., Nginx Ingress).

**12. How can you check memory and CPU usage in pods and nodes?**

Check resource usage:

kubectl top pod

kubectl top node

**13. What is manual scheduling, taints/tolerations, and node affinity?**

* **Manual scheduling**: Assigns pods to nodes manually (nodeName).
* **Taints & Tolerations**: Prevents pods from running on certain nodes unless tolerated.
* **Node Affinity**: Assigns pods to preferred nodes.

Example of **tainting a node**:

kubectl taint nodes node1 key=value:NoSchedule

**14. What is the difference between "Retained" and "Delete" in PVC?**

* **Retained**: The storage remains even after the PVC is deleted.
* **Delete**: The storage is deleted when the PVC is deleted.

Defined in persistentVolumeReclaimPolicy.

**15. What are the different deployment strategies in Kubernetes?**

* **Rolling Update**: Replaces pods gradually.
* **Recreate**: Deletes all old pods before creating new ones.
* **Blue-Green**: Two versions run simultaneously; switch traffic when ready.
* **Canary**: Deploys new version to a subset of users before full rollout.

**16. What is Blue-Green Deployment?**

Two environments run in parallel:

* **Blue**: Existing version.
* **Green**: New version.

Traffic is switched to **Green** after testing.

**17. What is Recreating, Rolling Update, and Canary?**

* **Recreate**: Deletes old pods before starting new ones.
* **Rolling Update**: Gradually replaces pods with the new version.
* **Canary**: Deploys new versions to a small percentage of users first.

**18. What is the difference between Liveness Probe and Readiness Probe?**

| **Probe** | **Purpose** |
| --- | --- |
| **Liveness** | Checks if a pod is alive; restarts if needed. |
| **Readiness** | Checks if a pod is ready to receive traffic. |

Example of **Readiness Probe**:

readinessProbe:

httpGet:

path: /health

port: 8080

**19. What is the difference between EKS and ECS in AWS?**

| **Feature** | **EKS (Elastic Kubernetes Service)** | **ECS (Elastic Container Service)** |
| --- | --- | --- |
| Orchestrator | Kubernetes | AWS Native |
| Flexibility | High | Limited |
| Scaling | Automatic | Manual or Fargate |

EKS is better for Kubernetes workloads, while ECS is easier for AWS users.

**20. Why do we use Helm Charts?**

* Simplifies Kubernetes deployments.
* Manages applications with versioning.
* Reduces complexity.

Example:

helm install my-app ./chart

**21. How did you implement logging in Kubernetes?**

* **kubectl logs** for basic logs.
* **Fluentd, Elasticsearch, Kibana (EFK stack)** for advanced logging.
* **Prometheus and Grafana** for monitoring.

**22. What is the difference between StatefulSet and Deployment?**

| **Feature** | **StatefulSet** | **Deployment** |
| --- | --- | --- |
| Use case | Stateful apps (databases) | Stateless apps (web services) |
| Pod names | Fixed | Random |
| Storage | Persistent | Ephemeral |

**23. Have you created a Kubernetes cluster?**

Yes! (If applicable, explain tools like kubeadm, Terraform, or AWS EKS.)

Kubectl create deployment nginx –image nginx replicas=3 –dry-run-client-o wide yaml>dep.yaml

**Service File**

A Service file defines a logical set of Pods and a policy for accessing them. Here's an example of a Service file:

apiVersion: v1

kind: Service

metadata:

name: my-service

spec:

selector:

app: my-app

ports:

- name: http

port: 80

targetPort: 8080

type: LoadBalancer

This Service file defines a Service named my-service that selects Pods labeled with app: my-app. It exposes port 80 and forwards traffic to port 8080 on the selected Pods.

**Pod File**

A Pod file defines a single Pod. Here's an example of a Pod file:

apiVersion: v1

kind: Pod

metadata:

name: my-pod

spec:

containers:

- name: my-container

image: my-image

ports:

- containerPort: 8080

This Pod file defines a Pod named my-pod with a single container named my-container. The container runs the my-image image and exposes port 8080.

**Deployment File**

A Deployment file defines a desired state for a set of Pods. Here's an example of a Deployment file:

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-deployment

spec:

replicas: 3

selector:

matchLabels:

app: my-app

template:

metadata:

labels:

app: my-app

spec:

containers:

- name: my-container

image: my-image

ports:

- containerPort: 8080