50 Real Time Kubernetes Interview Questions and Answers

Kubernetes Interview Questions and Answers

1. What is Kubernetes

Kubernetes is one of the Leading open-source Container Orchestration Engine. It is used to automatic cluster deployment, scaling and manage containerized applications.

2. What is difference between Docker Swarm and Kubernetes

Docker Swarm	**Kubernetes**
It is Clustering for Docker Container	It is Container Orchestration
Setup is Easy	Setup is Hard
Application deployment using Pods, Deployments and Service	Application deployment using only Service
Auto scaling is Not Possible	Auto scaling is possible
It has no GUI Dashboard	It has GUI Dashboard
It supports logging and monitoring using ELK Stack,Grafana,Influx,etc.	It does not support

3. What is Kubeadm

Kubeadm helps for installing and configuring Kubernetes cluster using command line.

4. What are Kubeadm commands?

Command Name	**Purpose**
kubeadm init	Used on Master node and It is used to initialize and con any node as a master node.
kubeadm join	Used on worker node and It is used to initialize and con any node as worker node.
kubeadm token	It is used to genrate token.
kubeadm version	It used to check kubeadm version.

5. What are Kubernetes Cluster components on Master node

API Server, Scheduler, Controller Manager, ETCD

6. What are Kubernetes Cluster components on Worker node

Kubelet, Kubeproxy, Pods, Container

7. What is API Server

It is used to exposing various API's. It is used to create,delete and update any object inside the cluster using kubectl command. API objects can be pods,containers,deployments,services..etc.

8. What is Scheduler ?

Scheduler is responsible for physically scheduling pods across multiple nodes, depending upon when we submit requirement to API server, scheduler schedules pod accordingly.

9. What is Controller Manager?

It is responsible overall health of entire cluster such as no of nodes insides the cluster, up and running status as per specification.

10. What is ETCD

etcd is light weight key-value database, it stores information like about current state of cluster,..etc.

11. What is Worker node in Kubernetes?

Worker node can be any Physical Server or Virtual Machine where containers are deployed , containers can be docker,rocket,.etc.

12. What is Kubelet ?

Kubelet is primary agent which runs on each worker node.It ensures containers are running in pod.

13. What is Kubeproxy?

It is core Networking component of Kubernetes cluster, it is responsible for entire network configuration, it maintains distributed network across all containers, pods and nodes.

14. What is Pod?

Pod is scheduling unit in Kubernetes, it consists of one or more container. With the help of pod we can deploy one or more container.

15. What are the different types of services in Kubernetes ?

Cluster IP –   It is used to expose the service on internal IP within cluster.

Node Port – It is used to expose the service from outside.

Load Balancer – It creates external load balancer and assigns external IP to service.

External Name Creating – It is used expose the service using name.

16. What is the difference between deployment and service in Kubernetes ?

Deployment is an object in Kubernetes, using Deployment we can create and manage pods using replica set from template.

Deployment manages creating Pods using of Replica Sets

Service is responsible to allow network access to a set of pods.

17. What is the difference between pod and deployment in Kubernetes?

Pod is scheduling unit in Kubernetes, it consists of one or more container. With the help of pod we can deploy one or more container.

Deployment is an object in Kubernetes, using Deployment we can create and manage pods using replica set from template.

Both are objects in the Kubernetes API

18. What is the difference between config map and secrets in Kubernetes?

Config maps stores application configuration in a plain text format.

Secrets store sensitive data like password in an encrypted format

19. What is namespace in Kubernetes?

Using **namespace**, we can logically organize objects in the cluster like pod and deployments. When you create Kubernetes cluster , default, kube-system and kube-public namespace are available.

20. What is ingress in Kubernetes?

Ingress is a collection of routing rules for external services running in a Kubernetes cluster.

21. What is Namespace in Kubernetes/k8s ?

It is Kubernetes objects which is used to create multiple virtual clusters within same physical cluster.

We can deploy Pods, deployment, service within each Virtual Cluster called as Kubernetes Namespace.

22. What is use of Namespace in Kubernetes ?

Suppose you have Dev, QA and Prod Environment in your project and you want separate each environment in same cluster and deploy pods, deployments and services also.

In this scenario you can separate these resource in by creating Namespaces for Dev,QA,Prod and create pods, deployments, services.

23. What is ingress in Kubernetes ?

Ingress it is a Kubernetes objects which allows access to your Kubernetes services from outside/external.

Using Ingress we can expose pod's port like 80 ,443 from outside network of Kubernetes over internet.

24. What are the different types of Ingress Controller in Kubernetes

Below are some most used Ingress controllers on Kubernetes Cluster

1. Nginx Ingress Controller
2. AWS ALB
3. Traefik
4. Azure Application Gateway
5. HA Proxy
6. Contour
7. Istio

25. What is Replication Controller in Kubernetes ?

A Replication Controller ensures that a specified number of pod replicas are running at any given time. In other words, a Replication Controller makes sure that a pod or a homogeneous set of pods is always up and available.

26. What is ReplicaSet's 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. The ReplicaSets are also known as next generation replication controller.

ReplicaSets checks whether the target pod is already managed by another controller as well (like a Deployment or another ReplicaSet).

27. What is the Difference between Kubernetes Replication Controllers and ReplicaSets

Replication Controller and Replica Set do almost the same thing. Both of them ensure that a specified number of pod replicas are running at any given time.

The difference comes with the usage of selectors to replicate pods. Replica Set use Set-Based selectors which gives more flexibility while replication controllers use Equity-Based selectors.

28. Why we need replication in Kubernetes ?

A container or pod may crash because of multiple reasons. The main purpose of using Replication is Reliability, Load Balancing, and Scaling. It ensures that the pre-defined pods always exists.

To understand this in an easier manner lets take an example ->

Lets assume we are running our application on a single pod. What if for some reason our application crashes and the pod fails. Users will no longer be able to access our application.

To prevent users from losing access to our application we would like to have more than one instances of our application running at the same time. That way if one pod fails we still have our application running on the other one. The replication controller helps us run multiple instances of a single pod in the Kubernetes cluster. Thus providing **high availability**.

Kubernetes Networking and Security Interview Questions and Answers

How does Kubernetes handle networking for applications?

Kubernetes uses a service mesh to provide networking for applications. A service mesh is a dedicated infrastructure layer for handling communication between microservices.

What is a Kubernetes ingress controller?

An ingress controller is a component that handles incoming traffic to a Kubernetes cluster. It is responsible for routing traffic to the correct service based on the hostname or URL.

How does Kubernetes secure applications?

Kubernetes provides a number of security features, including network policies, pod security policies, and role-based access control (RBAC).

Advanced Kubernetes Interview Questions and Answers

How does Kubernetes handle horizontal pod autoscaling (HPA)?

HPA is a controller that automatically scales the number of pods in a deployment based on CPU or memory usage.

What are the different ways to manage persistent storage in Kubernetes?

Kubernetes supports a number of different ways to manage persistent storage, including using PersistentVolumes (PVs), PersistentVolumeClaims (PVCs), and CSI drivers.

How does Kubernetes handle log collection and monitoring?

Kubernetes provides a number of tools for log collection and monitoring, including the Fluentd logging agent and the Heapster metrics server.

What is difference between Kubectl Describe vs kubectl get Get vs kubectl Explain

1. The **kubectl describe** command is used to display detailed information about specific Kubernetes resources.
eg. kubectl describe pod my-pod -n my-namespace

2. The **kubectl get** command is used to retrieve a curated list of Kubernetes resources of a particular type of resource in the cluster. It provides a view of the current state of multiple resources.
eg. kubectl get pods -n my-namespace

3. The **kubectl explain** command is used to retrieve detailed information about the structure and properties of Kubernetes resources.
eg. kubectl explain pod

Difference between Cilium and Calico network plugin?

Cilium and Calico are both popular networking solutions used in Kubernetes environments,

but they have some different features and focuses which might make one more suitable than the other depending on the specific needs of a deployment.

Cilium:
1. BPF-based Networking:

Cilium utilizes eBPF (extended Berkeley Packet Filter), a powerful Linux kernel technology, to provide highly efficient network and security capabilities.

eBPF allows Cilium to perform networking, security, and load balancing functionalities directly in the Linux kernel without requiring traditional kernel modules or network proxies.

2. Security:

Cilium is highly focused on security. It offers network policies for container-based environments, API-aware network security, and support for transparent encryption.

3. Scalability and Performance:

Thanks to eBPF, Cilium is known for high performance and scalability, particularly in environments with high throughput and low latency requirements.

4. Service Mesh Integration:

Cilium integrates well with service mesh technologies like Istio, providing efficient load balancing and networking capabilities.

Calico:
1. Flexibility in Data Planes:

Calico provides options to use either standard Linux networking and routing capabilities or eBPF for more advanced scenarios.

This flexibility can be useful in different deployment environments.

2. Network Policy Enforcement:

Calico is well-known for its robust implementation of Kubernetes network policies, offering fine-grained control over network communication.

3. Cross-Platform Support:

Calico supports a wide range of platforms and environments, including Kubernetes, OpenShift, Docker EE, OpenStack, and bare-metal services.

4. Performance:

While Calico can use eBPF for high performance, its standard mode using IP routing and iptables is also very efficient and scalable.

💡 Choosing Between Cilium and Calico:

✅ If your primary focus is on advanced networking capabilities, leveraging the latest kernel technologies for performance, and tight integration with service meshes, Cilium is a strong choice.

✅ If you need a flexible, platform-agnostic solution that offers robust network policy enforcement and can operate in a wide variety of environments, Calico might be more suitable.

Ultimately, the choice between Cilium and Calico will depend on the specific requirements of your infrastructure, such as performance needs, security requirements, existing technology stack, and your team's familiarity with these tools.

What are different storage options are available in Kubernetes

Answer:

• **EmptyDir**
-> created when the Pod is assigned to a node
-> RAM & Disk based mounting options
-> Volume is initially empty

• **Local**
-> represents a mounted local storage device
-> only be used as a statically created PV
-> Dynamic provisioning not supported
-> must set a PV nodeAffinity

• **Hostpath**
-> mounts a file or dir from the host node's FS to Pod
-> presents many security risks- Avoid it
-> Mostly useful for Static Pod! **Why**?
↳ static Pods cannot access CM

• **PVC**
-> expanding PVC is enabled by default
-> used to mount a PersistentVolume
-> we can pre-bind PV & PVC

• **Secret**
-> secret volumes are backed by tmpfs (a RAM-backed fs) so they are never written to non-volatile storage
-> A Secret is always mounted as readOnly

• **ConfigMap**
-> Provides a way to inject config data into pods
-> You must create a CM before you can use it
-> CM is always mounted as readOnly.

• **CSI**
-> defines standard interface for container orchestration
-> CSI compatible volume driver need to deployed
-> Most widely used Option

Kubernetes Pod Troubleshooting Interview Questions and Answers

1. **POD OOM (Out of Memory) Errors-Pod exceeds memory limits**
Resolution: Analyze resource usage: **`kubectl top pod<pod-name>`**. Adjust memory requests/limits in pod spec.

2. **Kubernetes Pod High CPU Usage – Pod consumes excessive CPU.**
Resolution: Monitor CPU utilization: `kubectl top pod <pod-name>`. Optimize application performance or scale horizontally.

3. **Kubernetes Pods Stuck in Pending State – Insufficient resources or scheduling issues.**
– Resolution: Increase cluster capacity or adjust pod requests/limits. Review node conditions: `kubectl describe node`.

4. **Kubernetes Pod Network Connectivity Issues – Pod unable to communicate with external resources.**
– Resolution: Diagnose network configurations: `kubectl describe pod <pod-name>`. Check network policies and firewall rules.

5. **Kubernetes Pod Storage Volume Errors – Failure in accessing or mounting volumes.**
– Resolution: Verify volume configurations: `kubectl describe pod <pod-name>`. Check storage class availability and permissions.

6. **Kubernetes pod Crashes and Restarting Pods- Application errors or resource constraints.**
– Resolution: Review pod logs: `kubectl logs <pod-name>`. Address application bugs or adjust resource allocations.

7. **Kubernetes pod Failed Liveness or Readiness Probes – Pod fails health checks, affecting availability.**
– Resolution: Inspect probe configurations: `kubectl describe pod <pod-name>`. Adjust probe settings or application endpoints.

8. **Kubernetes Pod Eviction due to Resource Pressure – Cluster resource scarcity triggers pod eviction.**
– Resolution: Monitor cluster resource usage: `kubectl top nodes`. Scale resources or optimize pod configurations.

9. **Docker Image Pull Failures – Issues fetching container images from the registry.**
– Resolution: Verify image availability and credentials. Troubleshoot network connectivity with the registry.

10. **Kubernetes Pod Security Policy Violations – Pods violate cluster security policies.**
– Resolution: Review pod security policies: `kubectl describe pod <pod-name>`. Adjust pod configurations to comply with policies.

Scenario Based Kubernetes Interview Questions and Answers

Scenario 1: Troubleshooting a deployment

You have deployed a new application to your Kubernetes cluster, but it is not working as expected. How would you troubleshoot the issue?

Answer:

1. **Check the deployment logs:** The first step is to check the logs of the deployment to see if there are any errors or warnings. You can use the `kubectl logs` command to view the logs of a pod.

2. **Check the pod status:** You can also use the `kubectl get pods` command to check the status of the pods in the deployment. Make sure that all of the pods are running and that they are in a healthy state.

3. **Check the service status:** If the pods are healthy, but the application is still not working, you can check the status of the service that exposes the application. Make sure that the service is running and that it is configured correctly.

4. **Check the ingress controller:** If you are using an ingress controller to route traffic to your application, you can check the logs of the ingress controller to see if there are any errors.

5. **Use kubectl exec:** If you need to troubleshoot the application further, you can use the `kubectl exec` command to run a command inside of a container.

Scenario 2: Scaling an application

Your application is experiencing a surge of traffic and you need to scale it up quickly. How would you do this?

Answer:

1. **Horizontal Pod Autoscaling (HPA):** If you have configured HPA for your deployment, it will automatically scale the number of pods up based on the CPU or memory usage of the application.

2. **Manual scaling:** If you do not have HPA configured, you can manually scale your deployment by editing the `.yaml` file and increasing the number of replicas.

3. **Blue-green deployment:** If you want to avoid downtime during a scaling operation, you can use a blue-green deployment strategy. This involves deploying a new version of your application to a separate set of pods and then switching traffic to the new pods once they are ready.

Scenario 3: Handling a node failure

One of the nodes in your Kubernetes cluster has failed. How would you recover from this?

Answer:

1. **Kubelet will restart pods:** The Kubelet on the failed node will detect that the node is down and will restart the pods that were running on that node on other healthy nodes in the cluster.

2. **Services will continue to function:** The services that expose the application will continue to function, even if some of the pods are down. This is because Kubernetes uses a service mesh to handle communication between pods, and the service mesh will automatically route traffic to the healthy pods.

3. **New node can be added:** Once the failed node has been replaced, the Kubelet on the new node will report to the master node and join the cluster. The master node will then reschedule the pods that were running on the failed node to the new node.

. Scenario 4: Scaling Applications

Question: How would you scale a Kubernetes deployment when you observe an increase in traffic to your application?

Answer: You can scale a deployment using the `kubectl scale` command. For example, to scale a deployment named “app-deployment” to three replicas, you would use:

bash
```
kubectl scale --replicas=3 deployment/app-deployment
```

This will ensure that three pods are running to handle increased traffic.

Scenario 5: Rolling Updates

Question: Describe the process of performing a rolling update for a Kubernetes deployment.

Answer: To perform a rolling update, you can use the `kubectl set image` command. For instance, to update the image of a deployment named “app-deployment” to a new version, you would use:

bash
```
kubectl set image deployment/app-deployment container-name=new-image:tag
```
Kubernetes will gradually replace the old pods with new ones, ensuring zero downtime during the update.

Scenario 6: Troubleshooting Pods

Question: A pod is not running as expected. How would you troubleshoot and identify the issue?

Answer: First, use `kubectl get pods` to check the status of the pod. Then, use `kubectl describe pod <pod-name>` to get detailed information, including events and container statuses. Inspecting the pod’s logs using `kubectl logs <pod-name>` for each container can provide insights into issues. Additionally, using `kubectl exec -it <pod-name> -- /bin/sh` allows you to access the pod’s shell for further debugging.

Scenario 7: Persistent Volumes

Question: Explain how you would manage persistent storage in Kubernetes.

Answer: Persistent Volumes (PVs) and Persistent Volume Claims (PVCs) are used for storage. A PV represents a physical storage resource, and a PVC is a request for storage by a pod. Admins create PVs, and users claim storage by creating PVCs. Pods reference PVCs. Storage classes define the type and characteristics of the storage. The YAML files for PVs, PVCs, and the deployment or pod need to be configured accordingly.

Scenario 8: Service Discovery

Question: How does service discovery work in Kubernetes, and how can services communicate with each other?

Answer: Kubernetes uses DNS for service discovery. Each service gets a DNS entry formatted as `<service-name>.<namespace>.svc.cluster.local`. Pods within the same namespace can communicate using this DNS. To enable communication between services in different namespaces, use the full DNS name, including the namespace. Kubernetes Services abstract the underlying pods, providing a stable endpoint for communication.

Scenario 9: Deploying StatefulSets

Question: Explain when you would use a StatefulSet instead of a Deployment, and how does it handle pod identity?

Answer: Use StatefulSets for stateful applications like databases, where each pod needs a unique identity and stable network identity. StatefulSets provide guarantees about the ordering and uniqueness of pods. Pods in a StatefulSet get a unique and stable hostname (e.g., `<pod-name>-0`, `<pod-name>-1`). This is crucial for applications requiring persistent storage and where the order of deployment and scaling matters.

Scenario 10: ConfigMaps and Secrets

Question: How do you manage configuration data and sensitive information in Kubernetes?

Answer: ConfigMaps are used to manage configuration data, while Secrets are used for sensitive information. ConfigMaps can be created from literal values or configuration files and mounted into pods as volumes or environment variables. Secrets store sensitive information and are mounted similarly. Ensure that access to Secrets is properly restricted and consider using tools like Helm for managing and templating configuration.

Conclusion:

We have covered, Kubernetes Interview Questions and Answers for Freshers and Experienced Candidate. If you need any support, please comment.