KUBERNETES QUESTIONS

Did You Know?

- Kubernetes was open-sourced in 2014 by Google and has become popular due to its potential to automate the deployment and scaling of applications.
- Kubernetes is called K8s as a numeronym, derived from abbreviating the word based on its first and last letter along with the number of letters in between.
- Kubernetes has been successful in creating a new ecosystem and market, enabling the creation of a niche market worth billions of dollars.
- More than 1400 contributors, including companies like Red Hat, Google, Microsoft, Alibaba, and Amazon, contribute to Kubernetes.
- In Kubernetes, the control unit is the pod, which is a group of containers performing the same task within a single application.
- Kubernetes manages scales not by the number of containers but by the sequence of pods.

1. What is Kubernetes?

First, let us compare Kubernetes with Docker Swarm:

Comparison	Kubernetes	Docker Swarm
Controller	Master	Manager
Slave	Nodes	Workers
Deployment unit	Pod	Task
Load balancing	Service	Ingress

Kubernetes is a container orchestration tool that is used for automating the tasks of managing, monitoring, scaling, and deploying containerized applications. It creates groups of containers that can be logically discovered and managed for easy operations on containers.

2. What are the benefits of Kubernetes?

With the container orchestration tool Kubernetes, it becomes extremely easy to handle containers. We can respond to customer demands by deploying the applications faster and in a more predictable manner.

Here, we will list some of the benefits of Kubernetes:

- Automatic scheduling
- Automated rollback
- Horizontal scaling
- Auto-healing capabilities

3. What is a Kubernetes cluster?

A Kubernetes cluster is a group of nodes that run containerized applications across various environments and machines—cloud-based, physical, virtual, and on-premises. It enables the easy development of applications as well as their management and movement.

4. What is Kubernetes used for?

Kubernetes is used for the automation of the manual operations that are involved in the deployment, management, and scaling of containerized applications. It keeps track of the ones that are deployed into the cloud, restarts orphaned ones, shuts down the unused, and automatically provides resources such as storage, memory, and CPU when required.

5. How does Kubernetes work?

The best way to carry out the management of the life cycle of containerized applications over a large scale is through a container orchestration system like Kubernetes. It automates the deployment and scaling of several containers simultaneously. Containers that are running the same application are arranged together and act as replicas. They serve to load balance incoming requests. Kubernetes, then, supervises these groups of containers and ensures that they are functioning correctly.

6. What is the difference between Kubernetes and Docker Swarm?

Docker Swarm is a default container orchestration tool that comes with Docker. Docker Swarm can only orchestrate simple Docker containers. Kubernetes, on the other hand, helps manage much more complex software application containers. Kubernetes offers support for larger demand production environment.

7. What is orchestration in software?

Application orchestration in the software process means that we can integrate two or more applications. We will be able to automate arrangement, coordination, and management of computer software. The goal of any orchestration process is to streamline and optimize frequent repeatable processes.

8. What is a Kubernetes namespace?

The Kubernetes namespace is used in the environment wherein we have multiple users spread in the geographically vast areas and working on multiple projects. What the namespace does is dividing the cluster resources between multiple users.

9. What are federated clusters?

Multiple clusters that are managed as a single cluster is referred to as federated clusters.

10. What is a pod in Kubernetes?

We can think of a Kubernetes pod as a group of containers that are run on the same host. So, if we regularly deploy single containers, then our container and the pod will be one and the same.

11. What is a node in Kubernetes?

A node in Kubernetes is a worker machine which is also known as a minion. This node could be a physical machine or a virtual machine. For each node, there is a service to run pods, and it is managed by master components. The node services could include kubelet, kube-proxy, and so on.

12. What is a Heapster?

The Heapster lets us do the container cluster monitoring. It lets us do cluster-wide monitoring and event data aggregation. It has native support for Kubernetes.

13. What is a container cluster?

A container cluster lets us place and manage containers in a dynamic setup. It can be considered as a set of nodes or Compute Engine instances. The API server of Kubernetes does not run on cluster nodes, instead the Container Engine hosts the API server.

INTERMEDIATE QNS:

14. What is a kubelet?

We can think of a kubelet as the lowest level component in a Kubernetes. The kubelet is responsible for making the individual machines run. The sole purpose of a kubelet is that in a given set of containers, it has to ensure that they are all running.

15. How to write a Kubernetes scheduler?

The kube-scheduler is the default scheduler for Kubernetes. It is designed such that if you prefer, you can write your own one and use that instead.

Following is the syntax:

```
kube-scheduler [flags]
```

The scheduling life cycle:

1. A pod is created and the preferred state is mentioned, and without filling the node name, it is saved to etcd
2. The scheduler notices the new pod with no node bound
3. It finds a suitable node for that pod
4. It then informs the API server to bind the pod to the node, and next, the new desired state is saved to etcd
5. Kubelets watch the pods that are bound and start the containers on the particular node

16. What are the ways to provide API Security on Kubernetes?

Following are some of the ways that provide API Security:

- Using the correct auth mode with the API server authentication mode= Node, RBAC
- Ensuring that the traffic is protected by TLS
- Using API authentication
- Ensuring that kubeless protects its API via authorization-mode=Webhook
- Monitoring RBAC failures
- Removing default Service Account permissions

- Ensuring that the kube-dashboard applies a restrictive RBAC policy
- Implementing a pod security policy for container restrictions and the protection of the node
- Using the latest version of kube

17. If an organization is looking for ways to improve its deployment methods and desires a more scalable and responsive platform, what should be done?

The company should move to a cloud environment and implement a microservice architecture for implementing Docker containers. Once the base framework is set up, Kubernetes can be used for the autonomous development of applications and the quick delivery of the same by the team.

18. What is the difference between a replica set and a replication controller?

The difference is mainly in the selectors used for pod replication. A replica set uses set-based selectors, and replication controllers use equity-based selectors.

19. How does Kubernetes scale?

The kubectl scale command enables the ability to instantly change the number of replicas needed for running an application. While using this command, the new number of replicas need to be specified by setting the –replicas flag.

20. What is a Kubernetes context?

A context is a group of access parameters that has a cluster, a user, and a namespace. The current context is the cluster that is currently the default for kubectl, and all kubectl commands run against that cluster.

21. How can you list all services in the current namespace?

Use kubectl get services to list all services.

22. How do you create a new service to expose a deployment in Kubernetes?

Use kubectl expose deployment nginx –port=80 –target-port=80. This will create a new service called “nginx” that targets port 80 on the nginx deployment pods.

23. What command is used to delete a service in Kubernetes?

To delete a service in Kubernetes, you can use the kubectl delete service command.

24. What does a kube-scheduler do?

The kube-scheduler has the job of assigning the nodes to the newly created pods.

ADVANCED QNS:

25. Give examples of some recommended security measures for Kubernetes.

- Defining resource quotas
- Auditing support
- Providing restricted access to etcd
- Regular security updates
- Network segmentation
- Strict resource policies
- Regular scans for security vulnerabilities
- Using images from repositories that are authorized

26. What is a Headless Service?

The headless service is like normal services but without the Cluster IP. It enables direct access to pods without the need for a proxy.

27. 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.

28. What is Kubectl?

Kubectl is a Kubernetes command-line tool that is used for deploying and managing applications on Kubernetes. Kubectl is especially useful for inspecting the cluster resources, and for creating, updating, and deleting the components.

29. What is GKE?

GKE is Google Kubernetes Engine which is used for managing and orchestrating systems for Docker containers. GKE also lets us orchestrate container clusters within the Google Public Cloud.

30. What is kube-proxy?

The kube-proxy runs on each of the nodes. It can do simple tasks such as TCP, UDP, forwarding, and so on. It shows the services in the Kubernetes API on each node.

31. What are the components of a Kubernetes Master?

The components of the Kubernetes Master include the API server, the controller manager, the Scheduler, and the etcd components. The Kubernetes Master components are responsible for running and managing the Kubernetes cluster.

32. What is the use of kube-controller-manager?

It is the Kubernetes Controller Manager. The kube-controller-manager is a daemon that embeds the core control loops which regulate the system state, and it is a non-terminating loop.

33. 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.

34. Where is the Kubernetes cluster data stored?

The primary data store of Kubernetes is etcd, which is responsible for storing all Kubernetes cluster data.

35. How to set a static IP for Kubernetes load balancer?

Kubernetes Master assigns a new IP address.

We can set a static IP for Kubernetes load balancer by changing the DNS records whenever Kubernetes Master assigns a new IP address.

36. What is Ingress and how does it help route traffic in a Kubernetes cluster?

Ingress is an API object that manages external access to services in a Kubernetes cluster. It allows you to create rules for routing external traffic to specific services based on the URL path, hostname, and other criteria. Ingress is typically implemented using a load balancer, such as NGINX or HAProxy.

Here is a simplified explanation of how Ingress works:

- A user sends a request to the Kubernetes cluster and then this request is routed to the Ingress controller.
- The Ingress controller looks up the Ingress rules to determine which service to route the request to.
- The request is forwarded to the selected service.
- The service processes the request and sends a response back to the user

37. Can you explain what Kubernetes is and why it's widely used for container orchestration?

Kubernetes is an open-source container orchestration platform that simplifies the deployment, management, and scaling of containerized applications. It is widely used due to its scalability, flexibility, and ease of use, enabling efficient container management and improved application performance.

38. Can you explain the purpose and usage of the kubectl describe pod command?

The kubectl describe pod command provides in-depth information about a specific pod within a Kubernetes cluster. It displays the pod's status, events, configuration, and other relevant details. To utilize this command, you need to specify the name of the pod you want to examine. For instance, to describe a pod named my-pod, you would use the following command:

```
kubectl describe pod my-pod
```

39. How do I utilize the kubectl command to eliminate a pod within a Kubernetes cluster?

To eliminate a pod in a Kubernetes cluster using the kubectl command, simply execute the following command:

kubectl delete pod [pod-name]

40. What command would you use to view the logs of a specific pod in Kubernetes?

To view the logs of a specific pod in Kubernetes, you can use the kubectl logs command. Syntax:

kubectl logs <pod_name>

41. What is a Kubernetes deployment?

A Kubernetes deployment provides a declarative way to deploy and manage pods and replicas. Key characteristics of a deployment:

- Specifies the desired state – number of replica pods to deploy
- The deployment controller handles scaling up/down and rolling updates of pods
- Supports rollback to previous versions
- Maintains revision history of deployments
- Offers availability and scaling guarantees for pods
- Used in conjunction with pods, replicas, and replication controllers
- Allows defining update strategies like rolling updates or blue-green deployments

42. How can you list all deployments in the current namespace?

To list all deployments in the current namespace in Kubernetes, you can use the kubectl get deployments command.

43. What is the command to create a new deployment in Kubernetes?

kubectl create deployment my-deployment --image=nginx:1.16

44. How do you check the status of a deployment rollout in Kubernetes?

We can check the status of a deployment using the below command:

kubectl rollout status deployment/my-deployment

45. How can you update the image of a container in a Kubernetes deployment using kubectl?

It can be achieved in two ways:

kubectl edit deployment my-deployment

kubectl set image deployment/my-deployment nginx=nginx:1.17

46. What are ConfigMaps and Secrets in Kubernetes?

ConfigMaps and Secrets are two key ways to inject configuration data and sensitive data into Kubernetes pods and containers.

ConfigMaps is used to decouple configuration artifacts from image content to keep containers portable. A ConfigMap is an API object used to store non-confidential data in key-value pairs.

Secrets are similar to ConfigMaps but are specifically intended to hold confidential data like passwords, OAuth tokens, and ssh keys. Secrets are encoded but not encrypted so should not be used to store highly sensitive data.

47. How do you list ConfigMaps and Secrets in a Kubernetes cluster?

We can use kubectl get configmaps command.

48. Explain the Kubernetes architecture.

Pods

Pods are the smallest units that Kubernetes administers. It constitutes a set of containers. It shares a single IP address and all the resources, such as storage and memory, among every container within it. A pod can have a single container when the service or application is a single process.

Deployments

Kubernetes deployments determine the scale at which one wants to run an application, such as how the pods need to be replicated on the Kubernetes nodes, the desired number of pod replicas to be run, and the desired update strategy for the deployment.

Services

If a pod dies, Kubernetes replaces it to prevent any downtime. A service is the only interface that the application consumers deal with. When pods are changed, their internal names and IPs might change as well. A service exposes a single IP address or machine name linked to pods whose numbers and names are unreliable. It ensures that nothing appears changed to the outside network.

Nodes

A Kubernetes node collects, runs, and manages pods that function together.

The Kubernetes Control Plane

The Kubernetes control plane is the main entry point for users and administrators to handle the management of various nodes. HTTP calls or command-line scripts are used to assign operations to it. How Kubernetes interacts with applications is controlled by the control plane.

Cluster

The above components put together in a single unit is referred to as a cluster.

Kubernetes Components:

The control plane and the individual nodes consist of three main components each.

Control plane

- **API Server**

The Kubernetes API server validates and configures data for API objects, including pods, replication controllers, services, etc. It serves REST operations and provides the frontend to the cluster’s shared state through which all other components communicate.

- **Scheduler**

The scheduler assigns work to the nodes, keeps track of the capacity of resources, and ensures that a worker node's operation is within the right threshold.

- **Controller Manager**

The controller manager ensures that a cluster's shared state is operating in the desired manner. It monitors various controllers, which respond to events.

Worker Node Components:

- **Kubelet**

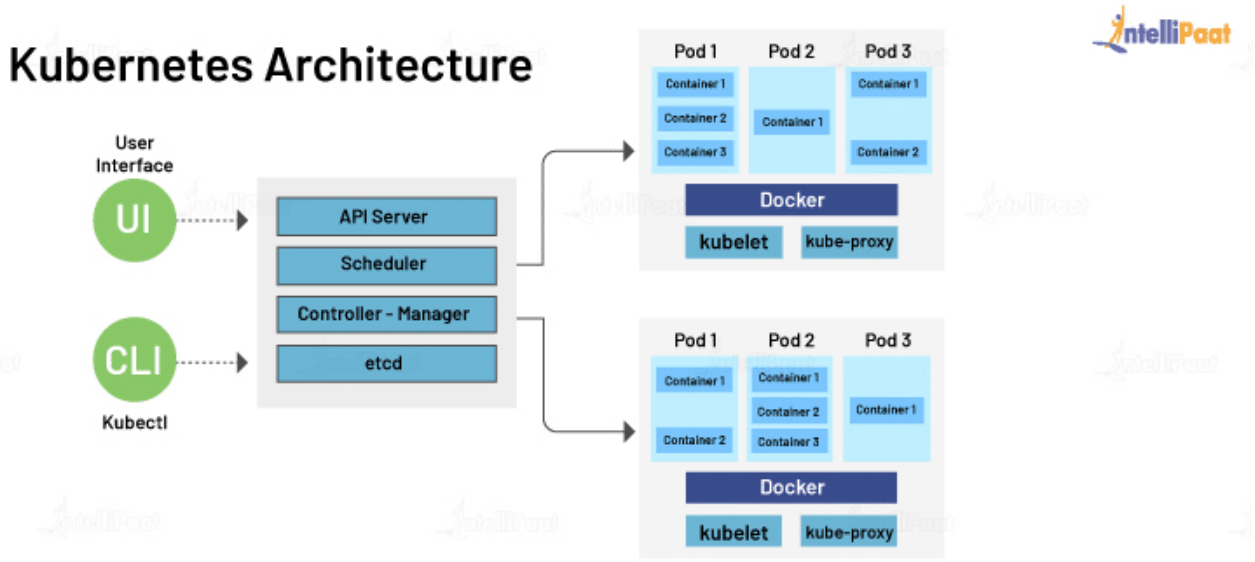
A kubelet keeps track of the state of a pod and ensures that every container is operating well.

- **Kube proxy**

The kube proxy is a network proxy that maintains network rules on nodes. It sends requests for work to the appropriate containers.

- **etcd**

This etcd component manages and holds the critical data that distributed systems require to operate. It is an open-source distributed key-value store that is used to share the state of a cluster. It helps with the setup of the overlay network for containers.



K8S SENARIO BASED QNS:

49. If an organization has a large distributed system with several data centers, virtual machines, and a huge number of employees working on various tasks, how can the tasks be managed with consistency with the help of Kubernetes?

The company can do well with something that offers scale-out capability, agility, and the DevOps practice to the cloud-based applications. Kubernetes, in this situation, can enable the customization of the scheduling architecture and support multiple container formats. This results in greater efficiency as well as provides support for various container networking solutions and container storage.

50. What do I need on-premises to run the Kubernetes architecture?

For Kubernetes, deciding the right storage and networking equipment is crucial as it facilitates interaction with resources for storage, load balancers, etc. A critical part of Kubernetes' value proposition is the ability to automate storage and the networking components.

51. How do I use kubectl to monitor the health and status of a Kubernetes cluster?

To check the status of a Kubernetes cluster using kubectl, you can use the following command:

kubectl get nodes

52. How do I utilize kubectl to retrieve a list of all pods within the active namespace?

To retrieve a list of all pods within the current namespace using kubectl, simply execute the following command:

kubectl get pods

AZURE KUBERNETES QUESTIONS

53. Explain Azure Kubernetes Service (AKS) and its key features.

AKS is a managed service that helps in the deployment, managing, and scaling of containerized applications using Kubernetes. A few features of AKS are:

- **Scalability**: AKs enable auto-scaling of applications by dynamic adjustment of the containers.
- **Integration**: It seamlessly integrates with other Azure Container services namely Azure Monitor, Azure Active Directory, Azure Policy, and so on.
- **Hybrid Cloud Support**: AKS supports hybrid cloud scenarios, allowing both on-premises and Azure cloud deployments.
- **Cost Efficient**: AKS has a pay-as-you-go policy that only asks for the cost of the services utilized.

54. What are the advantages of using AKS for deploying containerized applications compared to managing your own Kubernetes cluster?

The advantages of using AKs are:

- **Managed Services**: AKS is entirely managed by Azure, which means the services are taken care of by Azure services along with scaling, upgrading, and maintaining the Kubernetes cluster.
- **Simple Operations:** AKS eases operations such as cluster provisioning, node scaling, and cluster upgrades.
- **Build-in Availability**: AKS ensures high availability features such as automatic node repair and multiple availability zone support.

55. How do you configure and deploy a multi-container application on Azure Kubernetes Service (AKS)?

To configure and deploy the multi-container application, the steps are:

1. Containerize the application using Docker or another tool.
2. Create an AKS service cluster in the Azure subscription using the Azure portal
3. Authorize the AKS cluster using the Azure CLI by running:
4. az aks get-credentials –resource-group <resource-group-name> –name <aks-cluster-name>
5. Create a Kubernetes YAML deployment file that defines the configuration for the application.

6. Deploy the application to the AKS cluster using ‘Kebectl apply.
7. Monitor the performance
8. Update and scale the application as per the requirements.
9. Manage the configurations using AKS.

56. Explain the concept of a

In AKS node, pool is a group of nodes or virtual machines within a cluster that shares configuration settings. It enables resource optimization, scalability, availability, fault tolerance, and cost management.

57. How can you achieve high availability in Azure Kubernetes Service, and what considerations should be taken into account?

There are a few key strategies that ensure high availability in Azure Kubernetes Service (AKS):

Leverage Multiple Availability Zones: Distribute AKS nodes across Azure Availability Zones for redundancy. Enables seamless failover by utilizing nodes from different zones during issues.

Utilize Node Pools: Implement multiple node pools with diverse configurations. Each pool represents a logical node grouping, enhancing flexibility.

Enable Node Auto-Repair: Activate AKS Node Auto-Repair to automatically replace unhealthy nodes. Enhances resilience by swiftly addressing node failures.

Apply Pod Anti-Affinity: Define Kubernetes Pod Anti-Affinity rules to distribute pods across different AZs. Minimizes impact by preventing application concentration in a single zone.

Azure Traffic Manager: Employ Azure Traffic Manager for traffic distribution across AKS clusters in different regions. Facilitates regional failover, bolstering overall availability.

Regular Backups: Conduct routine backups for critical data and configurations. Document and test backup and restore procedures for reliability.

Monitoring and Alerts: Establish robust monitoring with Azure Monitor and Azure Security Center. Set up alerts for key metrics, ensuring proactive issue identification.

Scaling and Load Balancing: Implement horizontal pod autoscaling to adapt pod numbers based on resource usage. Deploy Azure Load Balancer for equitable traffic distribution.

Disaster Recovery (DR) Planning: Develop a comprehensive DR plan covering data replication and backup strategies. Include procedures for catastrophic failure handling.