Kubernetes

Firstly, Kubernetes in easy, it rules the Futures

K8s= KUBERNETES

1. Differences between Docker and Kubernetes?

Feature	Docker	Kubernetes
Core Functionality	Builds, ships, and runs containers.	Orchestrates and manages containers across multiple nodes.
Role in Ecosystem	Focuses on individual container lifecycle management.	Manages clusters of containers for scalability and high availability.
Scope	Operates at the container level.	Operates at the cluster level.
Networking	Provides basic networking (bridge, host, overlay).	Offers advanced networking with service discovery, load balancing, and network policies.
Scaling	Manual scaling by starting additional containers.	Automatic scaling using Horizontal Pod Autoscaler based on resource usage.
Fault Tolerance	Limited to restarting failed containers manually or using Docker Compose.	Provides self-healing, reschedules failed containers, and maintains application availability.
Monitoring	Basic container logs and resource usage stats.	Integrates with tools like Prometheus and Grafana for advanced monitoring and alerting.
Deployment	Simple container startup using commands like docker run.	Complex deployment with YAML manifests for pods, deployments, and services.

1.1 Architecture of Kubernetes

MASTER NODE {CONTROL PLANE}

API SERVER Expose k8s to external world (it takes all incoming request)

SCHEDULER it will acts that

ETCD key value store cluster related information

Controller manager it will look into every application or pods running as a planned

Cloud Controller manager it is helper to connect to multiple clouds

WORKER NODE (DATA PLANE)

Kublet is responsible for running the pods, it will take care of every pods, if any issue comes immediately inform to master

Container RUN Time It will run the container

Kubeproxy it will responsible for networking that allocates ip address (AUTO SCALING CONCEPT COMES HERE)

2. What is Kubernetes ?

- It all about YAML Files
- Kubernetes is an open-source container orchestration platform used to automate deployment, scaling, and management of containerized application
- Pods are the smallest deployable units in Kubernetes that encapsulate one or more containers, storage, and a network.

2.1 K8s Production System

PODS

Defination of how to run containers in pod.yaml file (single or multiple containers called pod)

Kubectl is command line for Kubernetes

Replicaset Actually doing auto healing kubectl get all

Kubectl get pod -o wide (to get ip address)

3. What is a Kubernetes Cluster?

- A Kubernetes cluster consists of a master node (control plane) and worker nodes to manage and run containers.

3.1 Services

- Loadbalancing
- Service discovery : labels and selectors

Labels: it will create a label for each pod (In deployment. Yaml we have metadata there we create lable)

Metadata

Label

app:payment

- Expose to outside world

3.2 Types of Services

Cluste ip By Default behaviour(we can access loadbalncing between nodes and services discovery)

Nodeport people in our organization and network they can access application

Loadbalncer Service that Expose application to the outside world

4. What are the main components of Kubernetes architecture?

a. Control Plane: API Server, Scheduler, Controller Manager, etcd.
b. Worker Nodes: Kubelet, Kube-Proxy, Container Runtime.

5. What is kubelet?

Kubelet is an agent that runs on worker nodes and ensures containers are running in a Pod as specified.

5.1 RBAC

You are managing a Kubernetes cluster for a development team. The team includes Developers and QA Testers, and you want to enforce access control based on their roles.

6. **What is a Deployment in Kubernetes?**

   A Deployment manages a set of identical Pods, ensuring the desired number of Pods are running.

7. **What is the difference between a Deployment and a StatefulSet?**
   a. Deployment: Used for stateless applications.
   b. StatefulSet: Used for stateful applications that require stable network identities.

8. **What are Kubernetes Namespaces?**

   Namespaces are logical (Sepration) partitions within a Kubernetes cluster to segregate resources and users.

9. **What is etcd in Kubernetes?**

   Etcd is a distributed key-value store that stores Kubernetes cluster data persistently.

10. **What is the role of the API Server in Kubernetes?**

    The API Server is the front-end for the Kubernetes control plane, exposing REST APIs to interact with the cluster.

11. **What is a Service in Kubernetes?**

    A Service defines a logical set of Pods and a policy to access them, enabling communication within or outside the cluster.

12. **What are the types of Services in Kubernetes?**

    ClusterIP, NodePort, LoadBalancer, ExternalName.

13. **What is a ReplicaSet?**

    ReplicaSet ensures a specified number of pod replicas are running at any given time.

14. **What is the difference between a ReplicaSet and a ReplicationController?**

    ReplicaSet is the newer version with support for set-based selectors, while ReplicationController uses equality-based selectors.

15. **What is a ConfigMap in Kubernetes?**

    ConfigMaps are used to store non-confidential data in key-value pairs, such as configuration files or environment variables.

16. **What is a Secret in Kubernetes?**

    Secrets store sensitive information, like passwords or tokens, securely.

17. **What is a DaemonSet?**

    A DaemonSet ensures a copy of a Pod runs on all or some nodes in the cluster.

18. **What is Ingress in Kubernetes?**

    Ingress is an API object that manages external access to services in the cluster, typically HTTP/HTTPS.

18.0 Why Ingress class name

This reduces the risk of traffic being routed incorrectly, especially in complex environments with multiple controllers.

- **public-ingress:** Handles public-facing services.
- **private-ingress**: Handles internal services.

18.1 ingress controller

An Ingress Controller is a specialized Kubernetes controller that manages HTTP and HTTPS traffic routing between external clients and services within a Kubernetes cluster

```
apiVersion: Networking.k8s.io/v1
Kind: Ingress
Metadata:
    Name: test-ingress
Spec:
```

18.2 host-based routing and path-based routing

Here's a comparison between host-based routing and path-based routing in Kubernetes, summarized in a table:

Feature	Host-Based Routing	Path-Based Routing
Definition	Routes traffic based on the host/hostname in the HTTP request (e.g., app.example.com, api.example.com).	Routes traffic based on the URL path in the HTTP request (e.g., /app, /api).
Use Case	Useful when hosting multiple applications/services on the same domain or in subdomains.	Useful when a single domain serves multiple paths or services (e.g., API, UI, etc.).
Configuration	Specified using the host field in the Ingress resource.	Specified using the path field in the Ingress resource.
Traffic Separation	Routes traffic to different services based on the hostname.	Routes traffic to different services based on URL paths.
Example	Route app.example.com to app-service and api.example.com to api-service.	Route /app to app-service and /api to api-service on the same domain (example.com).
Key Advantage	Easy to segregate traffic for different applications using domain names.	Cost-effective, as multiple services can run under the same domain using different URL paths.
Common Scenarios	Multi-tenant applications, microservices with unique subdomains.	API gateways, apps with shared domains, and single-page applications serving different components.

18.3 Load balancer and Ingress when to Use Which?

Scenario	Recommended Approach
You want to expose a single service to the internet.	Use LoadBalancer for direct and simple access.
You have multiple services and need centralized routing.	Use Ingress to manage traffic effectively and reduce costs.
Cost is a major factor in your decision.	Use Ingress, as it consolidates multiple services under one load balancer.

Feature	LoadBalancer	Ingress
Definition	A Service type in Kubernetes that provisions a dedicated external IP address using a cloud provider's load balancer for exposing applications.	A Kubernetes resource that provides HTTP/HTTPS routing to services based on hostnames or paths.
Purpose	Used for exposing a single service directly to the internet or external networks.	Used to manage and route external HTTP/HTTPS traffic to multiple services in the cluster.
Scope	Handles a single service at a time.	Handles multiple services through centralized routing rules.
Cost	Each LoadBalancer service creates a new load balancer instance in the cloud, which can be expensive.	Ingress uses a single ingress controller, reducing cost by routing multiple services through the same load balancer.
Traffic Routing	Traffic is routed directly to the associated service.	Traffic is routed to services based on rules (e.g., host-based, path-based).
Configuration	Simple to configure and deploy, with minimal settings.	Requires additional setup of an Ingress Controller and detailed configuration of routing rules.
Use Case	Suitable for small-scale deployments or services that need direct exposure to the internet.	Ideal for large-scale deployments with multiple services requiring centralized routing.
TLS Support	TLS termination must be configured at the Load Balancer level, depending on the cloud provider.	Supports TLS termination natively and easily managed in the Ingress resource.
Cloud Dependency	Completely dependent on the cloud provider's load balancer implementation (e.g., AWS ALB, Azure Load Balancer, GCP Load Balancer).	Works with any Kubernetes cluster as long as an Ingress Controller (e.g., NGINX, Traefik) is deployed.
Performance	Directly connects to a service, providing high performance for single-service access.	Performance depends on the Ingress Controller but supports routing for multiple services simultaneously.
Limitations	- Cannot handle routing based on hostnames or paths.- Creates a load balancer for every service, increasing cost.	- Requires an Ingress Controller to work.- Needs additional effort to set up and manage routing rules.
Common Scenarios	- Exposing a standalone service to the internet (e.g., a database or a backend API).	- Routing HTTP/HTTPS traffic for multiple microservices under a single domain.- Centralizing SSL/TLS management for web-based applications.

19. **What is Horizontal Pod Autoscaler (HPA)?**

    HPA automatically scales the number of Pods based on resource usage like CPU or memory.

20. **What is the difference between Horizontal Pod Autoscaler and Vertical Pod Autoscaler?**
    - HPA scales the number of Pods.
    - VPA adjusts the resource requests/limits of Pods.

21. **What is a Persistent Volume (PV) in Kubernetes?**

    A PV is a piece of storage provisioned in the cluster, independent of Pods.

22. **What is a Persistent Volume Claim (PVC)?**

    A PVC is a request for storage by a Pod.

23. **What are Kubernetes Operators?**

    Operators are custom controllers that extend Kubernetes functionality for specific applications.

24. **What is the role of Kube-Proxy?**

    Kube-Proxy manages networking rules to allow communication to and from Pods.

25. **What are taints and tolerations in Kubernetes?**
    - Taints are applied to nodes to restrict Pod scheduling.
    - Tolerations allow Pods to be scheduled on tainted nodes.

26. **What is a Node in Kubernetes?**

    A Node is a physical or virtual machine where Pods run.

27. **How does Kubernetes achieve high availability?**

    By running multiple master nodes, using etcd replication, and distributing workloads across nodes.

28. **What is a Kubernetes Job?**

    A Job creates one or more Pods to complete a task and ensures the task completes successfully.

29. **What is a CronJob in Kubernetes?**

    A CronJob runs Jobs on a schedule.

30. **What is the difference between a Service and an Ingress?**
    - Service provides internal/external Pod access.
    - Ingress manages HTTP/HTTPS routing for external traffic.

31. **How would you troubleshoot a Pod stuck in Pending state?**
    - Check events, node capacity, and resource requests/limits using:
    - kubectl describe pod <pod-name>

32. **How do you debug a crashing Pod?**
    - View logs:
    - kubectl logs <pod-name>
    - Check events and configurations.

33. **How would you scale a Deployment in Kubernetes?**
    - Scale using kubectl:
    - kubectl scale deployment <deployment-name> --replicas=5

34. **How do you expose a Deployment to external traffic?**
    - Use a Service of type LoadBalancer or NodePort.

35. **How do you implement rolling updates in Kubernetes?**
    - Update the Deployment:
    - kubectl set image deployment/<deployment-name> <container-name>=<new-image>

36. **How do you rollback a Deployment in Kubernetes?**
    - Use:
    - kubectl rollout undo deployment <deployment-name>

37. **What happens if a Pod crashes in Kubernetes?**
- Kubernetes recreates the Pod based on the ReplicaSet/Deployment specification.

38. **How do you secure Secrets in Kubernetes?**
- Use encryption at rest and RBAC to restrict access

38.1 RBAC

Role-Based Access Control (RBAC) in Kubernetes is a security mechanism used to control access to resources in the cluster. It defines who can do what on specific resources.

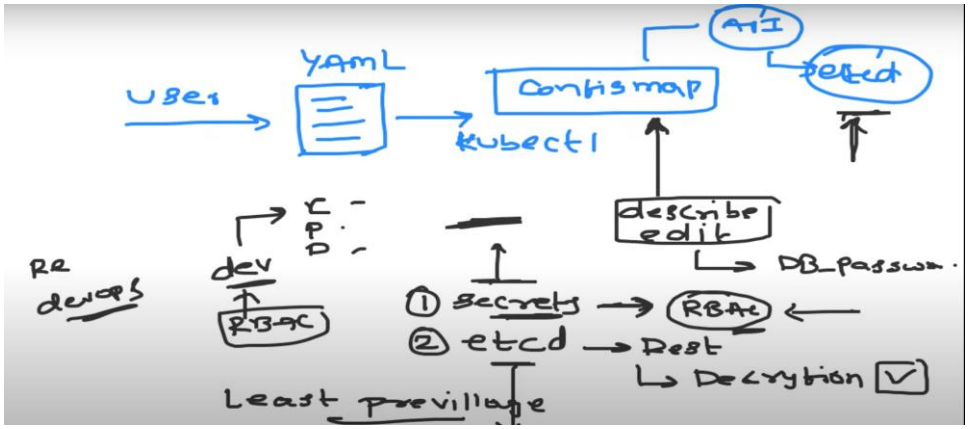
Role: Defines permissions within a namespace.
- ClusterRole: Defines permissions cluster-wide.
- RoleBinding: Binds a Role to a user/group.
- ClusterRoleBinding: Binds a ClusterRole to a user/group.

39. **How do you manage cluster-wide configurations?**
- Use ConfigMaps, Secrets, and Helm charts.

40. **How do you monitor a Kubernetes cluster?**
- Use tools like Prometheus, Grafana, or Kubernetes Dashboard.
-

41. **Custom resources**

CRD Custom resource Definition
CR
Custom Controller



To see environmental variables for Yamal file (Kubectl exec -it pod name -- /bin/bash)

```
spec:
  containers:
  - name: app-container
    image: nginx
    volumeMounts:
    - name: config-volume
      mountPath: /etc/config
  volumes:
  - name: config-volume
    configMap:
      name: app-config
```

CRD (Custom Resource Definition)

A **Custom Resource Definition** (CRD) is a way to extend Kubernetes' API to define your own resource types. It allows you to manage application-specific configurations or abstractions as Kubernetes-native objects, just like built-in resources like Pods, Deployments, or Services.

CR (Custom Resource)

A **Custom Resource** is an instance of a CRD. Once the CRD is defined and registered, you can create resources based on it.

Custom Controller

A **Custom Controller** is a program that watches for changes to your Custom Resources (CRs) and takes action based on the desired state defined in the resource's spec. It ensures the cluster's actual state matches the desired state specified in your CR.

42. Config maps and secretes

Its solving the problem of storing application, data later of time used by application

ConfigMaps vs Secrets in Kubernetes

ConfigMaps and Secrets are Kubernetes objects used to manage configuration data for applications. Both allow you to decouple configuration and sensitive data from your application code, but they are used in different scenarios.

Comparison Table: ConfigMaps vs Secrets

Aspect	ConfigMap	Secret
Purpose	Store non-sensitive configuration data as key-value pairs.	Store sensitive data like passwords, API keys, and certificates.
Data Encoding	Plain text.	Base64-encoded by default (but not encrypted).
Encryption	Does not support encryption out-of-the-box.	Can be encrypted using tools like Kubernetes Secrets Encryption Provider.
Usage	General configuration like environment variables, file paths, or feature flags.	Sensitive data that must not be exposed in plain text (e.g., credentials).
Default Namespace	Shared across all namespaces unless specified otherwise.	Shared across all namespaces unless specified otherwise.
Size Limit	~1MB (same for both ConfigMaps and Secrets).	~1MB (same for both ConfigMaps and Secrets).
Access Methods	Can be accessed as environment variables, command-line arguments, or mounted volumes.	Can also be accessed as environment variables or mounted volumes.
Visibility	Visible in plain text in API responses and kubectl output.	Base64-encoded, but still visible in kubectl output unless encryption is enabled.
Security	Not secure; suitable for non-sensitive data only.	Provides a basic level of security; for better security, use encryption.

Aspect	ConfigMap	Secret
Examples	Application configurations, log levels, or feature toggles.	Database passwords, OAuth tokens, SSL/TLS certificates, or API keys.

Example Scenarios

Using ConfigMaps

- **Store non-sensitive application configurations such as:**

  Database hostnames.-> Application log levels.-> Feature toggles.

Using Secrets

- **Store sensitive data such as: Database credentials.**

API keys-> SSH keys. ->TLS certificates.
