Kubernetes Concepts: Definitions and Features

1. Kubernetes

Definition:

An open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications across a cluster of machines.

Features/Uses:

- Automates container deployment and management
- Supports load balancing and service discovery
- Handles self-healing by restarting failed containers
- Provides rolling updates and rollbacks
- Scales applications dynamically based on load
- Works with various container runtimes (Docker, containerd, etc.)

2. Pod

Definition:

The smallest deployable unit in Kubernetes, representing one or more tightly coupled containers sharing network and storage.

Features/Uses:

- Groups containers that must work together
- Shares IP address and storage volumes
- Scheduled as a single unit on cluster nodes
- Supports co-located and tightly coupled container patterns

3. Deployment

Definition:

A controller managing a set of identical Pods, ensuring the desired number of Pods are running and handling updates and rollbacks.

Features/Uses:

Kubernetes Concepts: Definitions and Features

- Declarative updates for Pods and ReplicaSets
- Rolling updates without downtime
- Automatic rollback on failures
- Scaling Pods up or down easily
- Maintains desired application state

4. ReplicaSet

Definition:

Ensures a specified number of Pod replicas are running at any given time.

Features/Uses:

- Maintains desired number of Pod copies
- Automatically replaces failed or deleted Pods
- Used by Deployments to manage Pods
- Supports scaling for high availability

5. Service

Definition:

An abstraction that defines a logical set of Pods and a policy to access them.

Features/Uses:

- Provides stable IP and DNS for Pods
- Load balances traffic across Pods
- Supports service discovery inside the cluster
- Can expose services externally using NodePort or LoadBalancer
- Enables communication between microservices

6. Ingress and Ingress Controller

Definition:

- Ingress: A Kubernetes resource defining rules for routing external HTTP/S traffic to Services.
- Ingress Controller: The component that implements Ingress rules and manages traffic routing.

Kubernetes Concepts: Definitions and Features

Features/Uses:

- Enables host and path-based routing
- Provides SSL/TLS termination
- Centralizes external access management
- Supports load balancing and URL rewrites
- Reduces need for multiple external IPs

7. Ingress Resource

Definition:

A Kubernetes object that specifies the routing rules (hostnames, paths) for external traffic to reach Services inside the cluster.

Features/Uses:

- Defines routing rules in YAML manifests
- Works with Ingress Controllers to enforce routing
- Supports multiple domains and paths
- Simplifies external access configuration

8. ConfigMaps and Secrets

Definitions:

- ConfigMaps: Store non-sensitive configuration data for Pods.
- Secrets: Store sensitive information securely, such as passwords and tokens.

Features/Uses:

- Decouple configuration and sensitive data from container images
- Provide environment variables or config files to Pods
- Secrets use base64 encoding and can be encrypted at rest
- Simplify configuration updates without rebuilding containers

9. Namespace

Kubernetes Concepts: Definitions and Features

Definition:

A Kubernetes feature that partitions cluster resources between multiple users or teams, creating virtual clusters within a cluster.

Features/Uses:

- Organizes resources by teams, projects, or environments
- Provides resource isolation and limits
- Enables role-based access control (RBAC) per namespace
- Supports resource quotas for fair usage

10. Persistent Volume (PV)

Definition:

A storage resource in the cluster that exists independently of Pods and persists data beyond Pod lifecycles.

Features/Uses:

- Provides durable storage for stateful applications
- Supports various storage backends (NFS, cloud volumes, local disks)
- Used with Persistent Volume Claims (PVC) for dynamic provisioning
- Enables data persistence across Pod restarts or failures

11. StatefulSet

Definition:

A controller that manages Pods with stable network identities and persistent storage for stateful applications.

Features/Uses:

- Provides unique, stable Pod identities (hostnames)
- Maintains ordered deployment and scaling
- Ensures persistent storage per Pod
- Suitable for databases, message queues, and other stateful workloads

Kubernetes Concepts: Definitions and Features

12. Horizontal Pod Autoscaler (HPA)

Definition:

Automatically adjusts the number of Pods in a Deployment, ReplicaSet, or StatefulSet based on resource usage metrics.

Features/Uses:

- Monitors CPU, memory, or custom metrics
- Scales Pods up or down dynamically
- Helps maintain application performance and cost efficiency
- Supports automatic resource management

13. DaemonSet

Definition:

Ensures that a copy of a Pod runs on every node (or selected nodes) in the cluster.

Features/Uses:

- Runs node-level services like log collectors or monitoring agents
- Automatically adds Pods on new nodes
- Removes Pods when nodes are deleted
- Provides consistent services across all nodes

14. Job

Definition:

Runs a batch or finite task to completion, ensuring Pods execute successfully and stop afterward.

Features/Uses:

- Executes one-time or periodic tasks
- Guarantees completion of a specified number of successful runs
- Supports parallel execution of multiple Pods
- Useful for backups, batch processing, and maintenance tasks

Kubernetes Concepts: Definitions and Features

15. ResourceQuota and LimitRange

Definitions:

- ResourceQuota: Limits the total resource usage within a namespace.
- LimitRange: Defines minimum, maximum, and default resource limits per Pod or Container.

Features/Uses:

- ResourceQuota prevents a namespace from over-consuming cluster resources
- LimitRange enforces fair resource allocation per Pod
- Helps maintain cluster stability and fairness
- Supports governance of resource usage by teams or projects

Kubernetes Commands for DevOps

Cluster Information & Configuration

kubectl version

kubectl config view

kubectl config use-context <context-name>

kubectl cluster-info

kubectl get nodes

Deploying Applications

kubectl apply -f <file.yaml>

kubectl create deployment <name> --image=<image-name>

kubectl expose deployment <name> --type=NodePort --port=80

Viewing Resources

kubectl get all

kubectl get pods

kubectl get services

kubectl get deployments

kubectl describe pod <pod-name>

kubectl describe deployment <name>

kubectl logs <pod-name>

Updating Applications

kubectl set image deployment/<name> <container-name>=<new-image>

kubectl rollout status deployment/<name>

kubectl rollout undo deployment/<name>

Deleting Resources

kubectl delete pod <pod-name>

kubectl delete deployment <deployment-name>

kubectl delete service <service-name>

kubectl delete -f <file.yaml>

Namespace Management

kubectl get namespaces

kubectl create namespace <namespace>

kubectl delete namespace <namespace>

kubectl config set-context --current --namespace=<namespace>

Executing Commands in Pods

kubectl exec -it <pod-name> -- /bin/bash

kubectl cp <pod-name>:/path/to/file /local/path

Port Forwarding

kubectl port-forward pod/<pod-name> 8080:80

Resource Status & Events

kubectl get events

kubectl top nodes

kubectl top pods

Generate YAML Manifests

kubectl create deployment <name> --image=<image> --dry-run=client -o yaml > deployment.yaml

Useful Shortcuts

k get po

k get svc

Pro DevOps Workflows

CI/CD pipelines with kubectl apply

Monitoring with kubectl top

Automated rollouts & rollbacks

Resource scaling via:

kubectl scale deployment <name> --replicas=5