Date : / /
Page No.

KUBERNETES

KUBERNETES (K8S)

→ Kubernetes is an open-source container management tool which automates container deployment, container scaling, and descaling and container load balancing (also called as container orchestration tool).

→ It is written in Golang.

→ It was first developed by Google and later donated to CNCF.

→ Kubernetes is an open source platform that manages Docker containers in the form of a cluster. Along with its automated deployment and scaling of containers, it provides healing by automatically restarting failed containers and rescheduling them when their hosts die.

FEATURES OF KUBERNETES

1. AUTOMATED SCHEDULING

Kubernetes provides advanced scheduler to launch containers on cluster nodes. It performs resource optimization.

2. SELF-HEALING CAPABILITIES

It provides rescheduling, replacing and restarting the containers which are died.

3. AUTOMATED ROLLOUTS AND ROLLBACKS.

It supports rollouts and rollbacks for the desired state of the containerized application.

4. HORIZONTAL SCALING AND LOAD BALANCING

Kubernetes can scale up and scale down the application as per the requirements.

Date : / /
Page No.

Ques What are microservices?

Microservices also known as microservice architecture is a architectural style that structures an application as a collection of services that are.

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- owned by a small team

As an architectural framework, microservices are distributed and loosely coupled, so one team's changes won't break the entire app.

The benefits to using microservices is that development teams are able to rapidly build new components of apps to meet changing business needs.

ORCHESTRATORS

Orchestrator basicelly helps in deploying and managing applications dynamicelly.

Date : / /
Page No.

Some works of orchestrator / Kubernetes

→ Deploy
→ Zero downtime updates
→ scale
→ self healing Containers

KUBERNETES Vs DOCKER.

	DOCKER SWARM	KUBERNETES
(i)	No Auto Scaling	(i) Auto scaling
(ii)	Does Auto load balancing	(ii) Manually configure your load-balancing settings
(iii)	It performs rolling updates to containers straightaway	(iii) K8S performs rolling updates to pods as a whole.
(iv)	Share storage volumes with any other containers	(iv) share storage volumes between multiple containers inside the same pods.
(v)	It uses third party tools like ELK	(v) K8S provides in-built tools for logging and monitoring.

Date : / /
Page No.

IMPORTANT TECHNOLOGIES IN KUBERNETES

~~Kubernetes~~ .

(i) KUBERNETES CLUSTER

Control Plane + Nodes

Kubernetes clusters are comprised of one master node and a number of worker nodes. These nodes can either be physical computers or virtual machines, depending on the clusters.

Kubernetes clusters allow containers to run across multiple machines and environments.
Virtual, physical, cloud-based, and on-premises

A Kubernetes cluster is a set of nodes that run containerized applications.

Control Plane / Master node + Worker node.

(ii) **Kubectl**

It is the Kubernetes CLI.
It communicates with the controller plane.

It is a command-line interface for running commands against Kubernetes clusters.

(iii) **CONTROL PLANE / MASTER NODE**

Kubernetes master is responsible for managing the entire cluster, coordinates all activities inside cluster and communicates with worker nodes to keep the Kubernetes and your application running.

Components of Control Plane / Master node :-

→ **API SERVER** - The API server is the entry point for all the REST commands used to control the cluster. All the administrative tasks are done by API server within master node. If we want to create, delete, update or display in Kubernetes it has to go through API server.

We can interact with API's using kubectl.

Date : / /
Page No.

→ **SCHEDULER** – It is a service in master responsible for distributing the workload. It is responsible for tracking and utilization of working load on each worker nodes and then placing the workload on which resources are available and can accept the workload.

→ **CONTROLLER MANAGER** – Also known as Controllers. It is a daemon which runs in nonterminating loop and is responsible for collecting and sending info to API server.

The key controllers are replication controller, end point controller, namespace controller and service controller.

Controllers are responsible for overall health of the entire cluster by ensuring that nodes are up and running all the time and correct pods are running as mentioned in specs file.

→ **etcd** – It is a distributed key-value lightweight database.
In Kubernetes, it is a central database for storing the current cluster state at any point of time and also used to store the config details such as subnets, config maps, etc.

(iv) **Worker Node**

Worker node contains all the necessary services to manage the networking between the containers, communicating with the control plane / master node and assign resources to the containers scheduled.

Its Components:

→ **Kubelet** = It is a primary node agent which communicates with the master node and executes on each worker node inside the cluster.

It gets the pod specs through the API server and execute the containers associated with the pods and ensures they are running and healthy.

→ **Kube-Proxy** – It is the core networking component inside the kubernetes cluster. It is responsible for maintaining the entire network configuration.

It maintains the distributed network across all the nodes, all the pods and across all the containers and also exposes

Date : / /
Page No.

services across the outside world.

It acts as a network proxy and load balancer for a service on a single worker node and manages the network routing for TCP and UDP packets.
It provides unique IP addresses to nodes

→ **PODS** — Pod is a group of containers that are deployed together on the same host.

With the help of pods we can deploy multiple dependent containers together so it acts as a wrapper around these containers so we can interact and manage them
It is a scheduling unit.

→ **DOCKER** — It is the containerization platform.

MINIKUBE (Free, for learning only)

minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes.

Date : / /
Page No.

MINIKUBE COMMANDS

(i) **minikube start**

this command will start the minikube cluster

(ii) **minikube pause**

to pause kubernetes without impacting deployed applications

(iii) **minikube unpause**

to unpause a paused instance

(iv) **minikube stop**

to stop/halt the cluster

(v) **minikube config set memory 16384**
Any number

to increase the default memory limit (requires a restart).

Date : / /
Page No.

(vi) minikube addons list

to browse the catalog of easily installed Kubernetes services

(vii) minikube start -p aged --kubernetes-version = v1.16.1
(is any older version)

to create a second cluster running an older kubernetes release

(viii) minikube delete --all

to delete all the minikube clusters

(ix) minikube status

to get the status of the cluster.

(x) minikube dashboard

to open the dashboard

KUBECTL COMMANDS FOR CLUSTER MANAGEMENT.

(i) kubectl cluster-info

Display endpoint information about masters and services in our cluster

(ii) Kubectl version

to display the kubernetes version.

(iii) Kubectl config view

to get the configuration of our cluster

(iv) Kubectl api-resources

to list the API ~~versions~~ resources that are available.

v) Kubectl api-versions.

to list the API version that are available

(vi) Kubectl get all --all-namespaces

to list everything

FOR DAEMONSETS

(i) Kubectl get daemonset

to list one or more daemonsets

(ii) Kubectl edit daemonset <daemonset-name>

to edit and update the definition of one or more daemonset.

(iii) Kubectl delete daemonset <daemonset-name>

to delete a daemonset

(iv) Kubectl create daemonset <daemonset-name>

to create a new daemonset

(v) Kubectl rollout daemonset

to manage the rollout of a daemonset

(vi) Kubectl describe ds <daemonset-name> -n <namespace-name>

to display the detailed stat of daemonsets within a namespace.

FOR DEPLOYMENT

(i) Kubectl get deployment

to list one or more deployments

(ii) Kubectl describe deployment deployment-name

to display the detailed state of one or more deployments

(iii) Kubectl edit deployment deployment-name

to edit and update the definition of one or more deployment on the server

(iv) kubectl create deployment deployment-name

to create a new deployment

(v) kubectl delete deployment deployment-name

to delete deployments

(vi) kubectl rollout status deployment deployment-name

to see the rollout of a deployment.

FOR EVENTS

(i) Kubectl get events

to list recent events for all resources in the system.

Date : / /
Page No.

(iii) kubectl get events --field-selector type=warning

to list warnings only.

(iii) kubectl get events --field-selector involvedobject.kind!=pod

to list events but excludes pod events

(iv) kubectl get events --field-selector involvedobject.kind=Node, involved object.name=node-name

to pull events for a single node with a specific name.

(v) kubectl get events --field-selector type!=Normal

to filter out normal events from a list of events.

For Logs

(i) kubectl logs pod-name

to print the logs for a pod.

(ii) kubectl logs --since=1h pod-name

to print the logs for the last hour for a pod

(iii) Kubectl logs --tail=20 $pod-name

to get the most recent 20 lines of logs

(iv) Kubectl logs -f $service-name [-c $container]

to get logs from a service and optionally select which container

(v) Kubectl logs -f $pod-name

to print the logs for a pod and follow new logs.

(vi) Kubectl logs -c $container-name $pod-name

to print logs for a container in a pod

(vii) Kubectl logs $pod-name pod.log

to output the logs for a pod into a file named pod.log.

(viii) Kubectl logs --previous $pod-name

to view the logs for a previously failed pod

Date : / /
Page No.

FOR MANIFEST FILES

(i) Kubectl apply -f manifest-file.yaml

to apply a configuration to an object by filename or stdin. Overrides the existing configuration.

(ii) Kubectl create -f manifest-file.yaml

to create objects

(iii) Kubectl create -f ./dir
↳ path

to create objects in all manifest files in a directory

(iv) Kubectl create -f 'url'

to create objects from a URL

(v) Kubectl delete -f manifest-file.yaml

to delete an object

Date : / /
Page No.

FOR NAMESPACES

(i) Kubectl create namespace <namespace-name>

to create a namespace with name mentioned.

(ii) Kubectl get namespace <namespace-name>

to list one or more namespaces

(iii) Kubectl describe namespace namespace-name

to display the detailed state of one or more namespace.

(iv) Kubectl delete namespace <namespace-name>

to delete a namespace.

(v) Kubectl edit namespace <namespace-name>

to edit and update the definition of a namespace.

(vi) Kubectl top namespace <namespace-name>

to display resource (CPU/memory/storage) usage for a namespace.

Date : / /
Page No.

FOR NODES

(i) kubectl taint node node-name

to update the taints on one or more nodes.

(ii) kubectl get node

to list one or more nodes

(iii) kubectl delete node node-name

to delete a node or multiple nodes

(iv) kubectl top node

to display resource usage (CPU/memory/storage) for nodes

(v) kubectl describe nodes | grep Allocated -A 5

to do resource allocation per node.

(vi) kubectl get pods -o wide | grep node-name

to pods running on a node

(vii) Kubectl Annotate node node-name

to annotate a node

(viii) Kubectl cordon node node-name

to mark a node as unschedulable

(ix) Kubectl uncordon node node-name

to mark a node as schedulable.

(x) Kubectl drain node node-name

to drain a node in preparation for maintainance.

(xi) Kubectl label node

to add or update the labels of one or more nodes.

FOR PODS

(i) Kubectl get pods

to list one or more pods

Date : / /
Page No.

(ii) Kubectl delete pod <pod-name>

to delete a pod

(iii) Kubectl describe pod <pod-name>

to display the detailed state of a pod.

(iv) Kubectl create pod <pod-name>

to create a pod

(v) Kubectl exec <pod-name> -c <container-name> <command>

to execute a command against a container in a pod.

(vi) Kubectl exec -it <pod-name> /bin/sh

to get interactive shell on a single container pod.

(vii) Kubectl top pod

to display resource usage for pods.

(viii) Kubectl annotate pod <pod-name> <annotations>

to add or update the annotations of a pod

(viii) kubectl label pod <pod-name>

to add or update the label of a pod.

FOR REPLICATION CONTROLLERS

(i) kubectl get rc

to list the replication controllers.

(ii) kubectl get rc --namespace="namespace-name"

to list the replication controllers by namespace.

FOR REPLICA SETS

(i) kubectl get replicasets

to list replicasets

(ii) kubectl describe replicasets <replicaset-name>

to display the detailed state of one or more replicasets

Date : / /
Page No.

(iii) Kubectl scale --replicas=[x]

to scale a ReplicaSet.

For Secrets

(i) Kubectl creat secret

to creat a secret.

(ii) Kubectl get secrets

to list secrets

(iii) Kubectl describe secrets

to list details about secrets

(iv) Kubectl delete secret <secret-name>

to delete a secret.

For Services

(i) Kubectl get services

to list one or more services.

(ii) Kubectl describe services

to display the detailed state of a service.

(iii) Kubectl expose deployment [deployment-name]

to expose a replication controller, service, deployment or pod as a new kubernetes service.

(iv) Kubectl edit services

to edit and update the definition of one or more services

FOR SERVICE ACCOUNTS

(i) Kubectl get serviceaccounts

to list service accounts

(ii) Kubectl describe serviceAccounts

to display the detailed state of one or more service accounts

Date : / /
Page No.

(iii) Kubectl replace serviceaccount

to replace a servic account

(iv) Kubectl delete serviceaccount (service-account-name)

to delete a servic account

FOR STATEFUL SET

(i) Kubectl get statefulset

to list statefulset

(ii) Kubectl delete statefulset/[statefulset-name] --cascade=false

to delete stateful set only (no pods)