Kubernetes Notes – Based on Java Techie Video (2 hrs)

Date: 19/04/2025

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

🔹 Kubernetes is a container orchestration (management) tool.

- It helps in deploying, scheduling, scaling, and load balancing containerized applications.

- Containers (like Docker) are used to package and run applications.

- You write code → Build a Docker image → Push it to a container registry → Kubernetes takes care of everything else.

-------------------------------------------------------------------------------

★ Kubernetes Core Concept Flow:

Cluster → Node → Pod → Container

- Each Pod has a separate IP address.

🔁 ReplicaSet:

- Acts as a backup.

- If one Pod goes down, ReplicaSet automatically creates a new one.

-------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\* SERVICES IN KUBERNETES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

➡ Why Services are Needed:

1. Pods are temporary:

- Pods can be deleted/recreated.

- New Pod = New IP → Not reliable to connect directly.

- ✅ Solution: Service gives a stable IP and DNS name.

2. Load Balancing:

- Multiple replicas need even traffic distribution.

- ✅ Service automatically load balances across Pods.

3. Internal & External Access:

- Microservices need internal communication.

- Some apps need internet access.

- ✅ Kubernetes provides 4 Service types:

- ClusterIP → Internal only (default)

- NodePort → Exposes app on port of each node

- LoadBalancer → Exposes app to internet (cloud-based)

- ExternalName → Maps to external DNS

Summary:

- Pods change → Service gives a fixed access point.

- Load balancing → Handled automatically by Service.

- Internal/external routing → Based on Service type.

-------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\* DEPLOYMENT IN KUBERNETES \*\*\*\*\*\*\*\*\*\*\*\*\*\*

➡ What is Deployment?

- Deployment is a K8s object used to manage Pods.

- Ensures self-healing, scaling, and rolling updates.

📌 Example (using CLI):

kubectl create deployment spring-boot-k8s --image=springboot-k8s-demo:1.0 --port=8080 --replicas=4

📌 Can also be created using YAML.

Summary:

- Manual Pod creation is tough → Deployment automates it.

- Pod crash → Deployment recreates it.

- Version upgrade → Supports rolling updates.

- Scaling → Easily scale Pods using Deployment.

-------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\* CONFIGMAP & SECRETS \*\*\*\*\*\*\*\*\*\*

➡ Why do they exist?

- Store application config values and secrets \*\*outside\*\* the Pod.

- ConfigMap → Stores data in \*\*plain text\*\*.

- Secret → Stores \*\*sensitive data (encoded/encrypted)\*\* like passwords, API keys.

💡 Why separate?

- Security: Secrets are encrypted.

- Flexibility: ConfigMaps are readable and editable easily.

📌 Max size for Secrets = 1MB

Summary:

- Hardcoded config? → Use ConfigMap.

- Sensitive data? → Use Secret.

- Different configs for different environments? → Use both.

- Need to update config without touching code? → Inject them via env variables or volume mounts.

-------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ETCD IN K8s \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

➡ What is etcd?

- etcd is a \*\*key-value database\*\* used to store the \*\*entire Kubernetes cluster state\*\*.

📌 What it stores:

- Pod details

- Services, ConfigMaps, Secrets

- Node info, deployments, etc.

📌 Who uses it:

- Only the API server talks to etcd.

📌 Why etcd?

1. Reliable cluster state storage

2. High availability & consistency (uses Raft protocol)

3. Fast access for other components (with watches)

4. Supports backup/restore

Summary:

- Cluster memory → etcd

- Highly available, consistent

- API server reads/writes all state here

- Backups help restore full cluster if needed

-------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\* KUBERNETES ARCHITECTURE \*\*\*\*\*\*\*\*\*\*

➡ Organization → Cluster

➡ Manager → Master Node

➡ Projects → Worker Node

➡ Developer → Pods

📌 Inside every cluster:

- Minimum 1 Master Node

- 1 or more Worker Nodes

---------------------------------------------------

1. MASTER NODE – Controls the whole cluster

It contains 4 key components:

a) API Server:

- Gateway to the cluster

- Handles all requests (via kubectl or dashboard)

- Validates and forwards requests

Commands:

kubectl get nodes kubectl get pods

b) Scheduler:

- Assigns Pods to Nodes based on resource usage (CPU/RAM)

- Ensures best-fit placement

c) Controller Manager:

- Continuously compares current state vs desired state

- Takes actions like restarting failed Pods

It includes:

- Node Controller → Checks node health

- Replication Controller → Maintains replica count

- Deployment Controller → Handles updates & rollbacks

- Job Controller → Manages batch jobs

d) etcd:

- Stores all cluster data as key-value

- Cluster’s memory (explained above)

---------------------------------------------------

2. WORKER NODE – Runs the actual applications (Pods)

Components:

a) kubelet:

- Agent running on every worker node

- Communicates with Master Node via API Server

- Ensures containers described in PodSpec are running and healthy

b) kube-proxy:

- Handles networking

- Forwards requests to correct Pods

c) Container Runtime:

- Engine that actually runs containers (e.g., Docker, containerd)

-------------------------------------------------------------------------------

🟩 Final Notes:

- Kubernetes is for \*\*container orchestration\*\*, not just running containers.

- It makes your app scalable, reliable, and easier to manage.

- You define your app once, and Kubernetes handles the rest.

* Practical demo :-
* Install minkube and kubectl
* Set path in env for your acc
* Once done open cmd and check
* minikube version
* we need a driver to start minikube so option are hyer-v , virtual box and docker . so I have use docker as driver
* minikube start --driver=docker
* for this docker desktop should be up and running
* minikube status ---------------to check minikube started successfully
* kubectl cluster-info --------to check cluster created or not
* kubectl get node ------------provide us a single node

now create a project and also its dockerfile

you need to dockerize the application

then from dcoker repo pull this image and run inside a k8s cluster

now how k8s will read your docker repo so run a below command

minkube docker-env

'minkube' is not recognized as an internal or external command,

operable program or batch file.

C:\Users\Babu>minikube docker-env

SET DOCKER\_TLS\_VERIFY=1

SET DOCKER\_HOST=tcp://127.0.0.1:64549

SET DOCKER\_CERT\_PATH=C:\Users\Babu\.minikube\certs

SET MINIKUBE\_ACTIVE\_DOCKERD=minikube

REM To point your shell to minikube's docker-daemon, run:

REM @FOR /f "tokens=\*" %i IN ('minikube -p minikube docker-env --shell cmd') DO @%i

Isko add kr y line cmd mai to read docker repo

Now add --- @FOR /f "tokens=\*" %i IN ('minikube -p minikube docker-env --shell cmd') DO @%i

Now k8s can read local docker repo

docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

registry.k8s.io/kube-apiserver v1.32.0 c2e17b8d0f4a 4 months ago 97MB

registry.k8s.io/kube-scheduler v1.32.0 a389e107f4ff 4 months ago 69.6MB

registry.k8s.io/kube-controller-manager v1.32.0 8cab3d2a8bd0 4 months ago 89.7MB

registry.k8s.io/kube-proxy v1.32.0 040f9f8aac8c 4 months ago 94MB

registry.k8s.io/etcd 3.5.16-0 a9e7e6b294ba 7 months ago 150MB

registry.k8s.io/coredns/coredns v1.11.3 c69fa2e9cbf5 8 months ago 61.8MB

registry.k8s.io/pause 3.10 873ed7510279 11 months ago 736kB

gcr.io/k8s-minikube/storage-provisioner v5 6e38f40d628d 4 years ago 31.5MB

now create docker image so move to project dire first

C:\Users\Babu>cd C:\Users\Babu\Downloads\springboot-k8s-example

Cls in cmd to clear everything

Now build image through below command

C:\Users\Babu\Downloads\springboot-k8s-example>docker build -t springboot-k8s-demo:1.0 .

C:\Users\Babu\Downloads\springboot-k8s-example>docker images ---to verify

Now we have image and run this image inside a pod so we have to create a deployment object

Two ways using yaml file or directly command prompt

So now using command -----

kubectl create deployment spring-boot-k8s --image=springboot-k8s-demo:1.0 --port=8080

to verify the deployment --- kubectl get deployment

now describe the deployment you can see all replicas and everything

kubectl describe deployment spring-boot-k8s

to see k8s successfully pull image and created pod command is

C:\Users\Babu\Downloads\springboot-k8s-example>kubectl get pods

To verify it pulled image and created a pod and a container is ruuning check logs

C:\Users\Babu\Downloads\springboot-k8s-example>kubectl logs spring-boot-k8s-994b94d96-26nbd

So now springboot app started running inside a pod container – show karenga sts jaise console type

Ab koi op chaiye jo uske ui ko access kare so service create karni hongi by below command

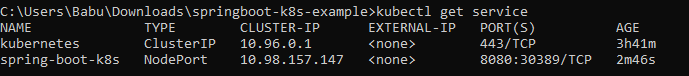
So expose the deployment in service object which we will create

C:\Users\Babu\Downloads\springboot-k8s-example>kubectl expose deployment spring-boot-k8s --type=NodePort

service/spring-boot-k8s exposed

type=NodePort -----------y service type hai

kubectl get service -------to check service created with type of nodeport

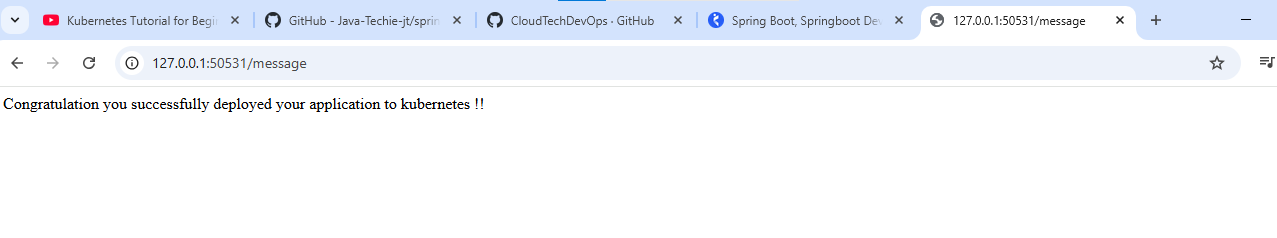


So now all the request will come to service and it will redirect to corresponding port depend upon avialble

To get url to acces to end point

C:\Users\Babu\Downloads\springboot-k8s-example>minikube service spring-boot-k8s –url

You will get url --- <http://127.0.0.1:50531>



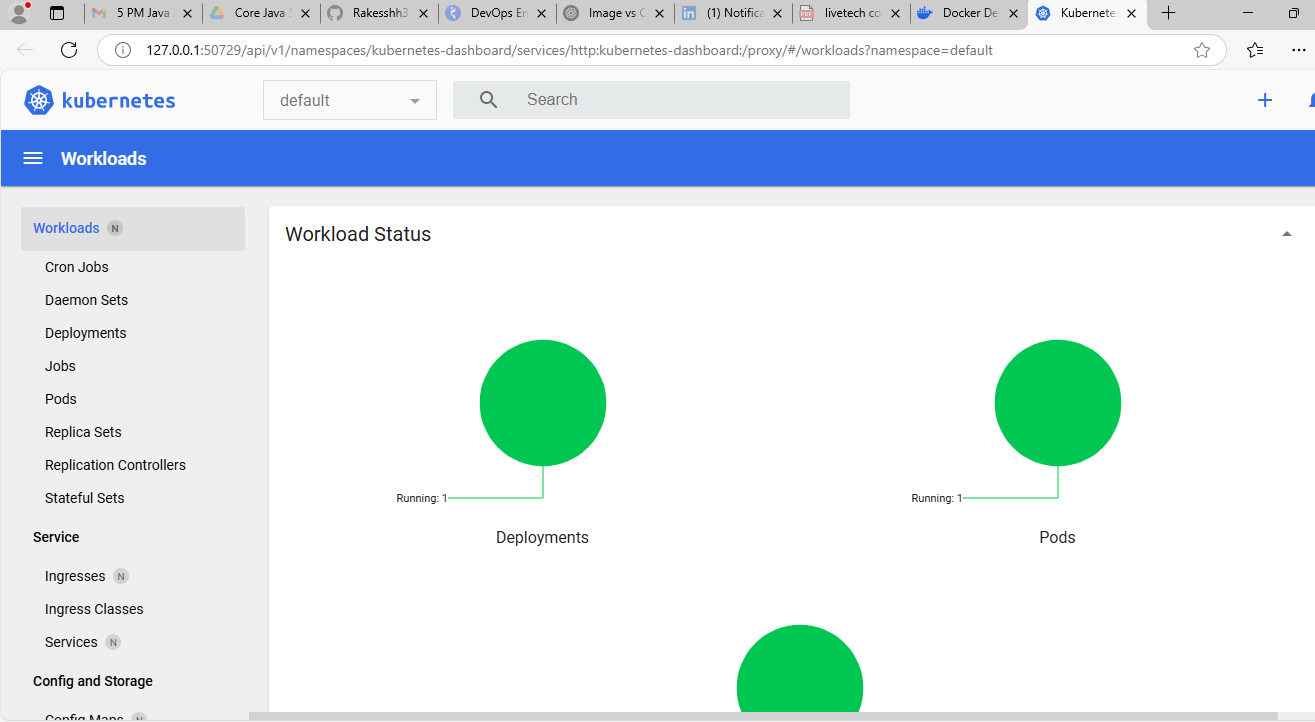
Aisa ayeng hogya hai deploy application

Now kill using ctrl c command

And now see the dashboard where health

minikube dashboard ----------show the nodes and components and their ststaus

it will k8s dashboard



Ek try kar y dashboard mai pod del kr dek who khud se chalu honga tode der mai that is self healing

Now cleanup every component

kubectl delete service spring-boot-k8s --- del service

kubectl delete deployment spring-boot-k8s --- del deployment

minikube stop ---- it stops local k8s cluster

C:\Users\Babu\Downloads\springboot-k8s-example>minikube delete ---del everyting

