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

.................................................................................................

java techie video 2hrs k8s 19/04/2025

k8s is a conatiner managememt tool

conatiner ....docker is a application where you can run any application tool

managememt ......take care of deploying , scheduling , scaling , load balancing of yoyr application

you write your code and push your image to conatiner like docker and rest k8s will take care

\*\*\*K8s components

cluster --> node --> Pod ---> conatiner

each pod has a sepaarte ip addres

replica set ----> backup of pod --one pod down , new will craete and start

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Service \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Service --------one pod gone down new pod with new ip

multiple pod can be group together into single service with lables and selectors

Kubernetes: Why Services Came into the Picture

1. Problem: Pods are Temporary (Ephemeral)

- Pods can be deleted or recreated at any time.

- Every time a Pod is recreated, it gets a new IP address.

- So directly connecting to Pods is unreliable.

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

2. Problem: Load Balancing is Needed

- Deployments create multiple replicas of the same Pod.

- Traffic needs to be evenly distributed between all replicas.

- ✅ Solution: Service automatically load balances traffic across all matching Pods.

3. Problem: Internal and External Communication

- Microservices need to talk to each other (like frontend → backend).

- Sometimes apps need to be exposed to the internet.

- ✅ Solution: Kubernetes provides different Service types:

- ClusterIP: Internal access only (default type).

- NodePort: Exposes the app on a specific port of each node.

- LoadBalancer: Exposes the app to the internet using a cloud load balancer.

- ExternalName: Maps the service to an external DNS name.

Summary:

- Pod IPs keep changing → Service gives stable IP and DNS.

- Multiple replicas → Service handles load balancing.

- Internal and external communication → Use Service types (ClusterIP, NodePort, LoadBalancer, ExternalName).

----------

\*\*\*\*\*deployment \*\*\*\*\*\*\*

deployment are k8s objects that are used for manganing the pods

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

you can craete this by using yaml file too

Summary:

- Manual Pod management is painful → Deployment automates it.

- Pod crashes → Deployment recreates them automatically (self-healing).

- Updating apps → Deployment supports rolling updates.

- Scaling needed → Deployment allows quick and easy scaling.

\*\*\*\*\*\*secrets and config map \*\*\*\*\*\*

use to store sensititive info of uour app like pass ,secretskey or apikey

they are placed outside pod and insid node . why two diff comp secrets and config map

in scerets its stores in encrypt form while in config map it stores in plain text

Summary:

- Hardcoded config → Use ConfigMaps to keep config flexible.

- Sensitive data in code → Use Secrets to store it securely.

- Same app, different configs → ConfigMaps/Secrets help manage environments.

- Need to update config without code change → ConfigMaps/Secrets can be mounted or injected easily.

\*\*\*\*\*\*\*\*ETCD\*\*\*\*\*\*\*\*\*\*

Kubernetes: Why etcd Came into the Picture

1. Problem: Kubernetes Needs to Store Cluster State

- Kubernetes has to remember all cluster data like:

- What Pods are running?

- What Nodes are part of the cluster?

- What Deployments, Services, ConfigMaps, Secrets exist?

- This data needs to be stored somewhere reliably.

- ✅ Solution: etcd stores the complete cluster state as key-value pairs.

2. Problem: High Availability and Consistency Required

- Cluster data must be safe even if one node crashes.

- Data must be consistent across all components.

- ✅ Solution: etcd is a distributed, consistent, highly available key-value store.

3. Problem: Fast Access and Coordination Between Components

- Kubernetes components like API Server, Scheduler, Controller Manager need fast access to current data.

- ✅ Solution: etcd allows fast reads/writes and watches for any data change.

4. Problem: Data Backup and Restore Needed

- If the cluster fails, we need a way to recover everything.

- ✅ Solution: etcd can be backed up and restored to recover full cluster state.

Summary:

- Cluster state storage → etcd stores everything as key-value data.

- Need for consistency & HA → etcd is distributed and fault-tolerant.

- Fast coordination → etcd gives real-time access and watches.

- Backup and recovery → etcd backups help restore cluster state.

Max limit is 1Mb to store secrets

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

\*\*\*\*\*\*\*\*\*\*\*\*architecture of K8s \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Organistaion ----Cluster

Manager---Master Node

project1----worker Node

developer---pods

inside a cluster their should be one master and one worker node

1.Master Node - it manges worker node and pods inside a container

It has 4 components

a.api manager

act as a cluster gateway . each request comig to your cluster will first serve by api server

it exposes few api to access operations going inside a cluster

we can access api server using either a command line tool kubectl or K8s Ui

kubectl get nodes

kubectl get pods

b.scheduler -it usually schedules pods across multilple nodes

it goes and check memory and cpu usage of each woker node and bbased on that it schedule pod to specific node

c. controller manger -it detects cluster state change like pod crashing , node crashing

- Keeps checking if the current state matches the desired state.

- If not, it makes the changes (like restarting Pods).

- Runs multiple controllers inside it, like:

- Node Controller → Checks node health and marks offline if needed.

- Replication Controller → Ensures the correct number of Pod replicas.

- Deployment Controller → Manages rolling updates and rollbacks.

- Job Controller → Manages batch jobs until they complete.

d. etcd (Cluster Memory / Database)

- A key-value store used to store all cluster data.

- Stores info like: Pods, ConfigMaps, Secrets, Nodes, etc.

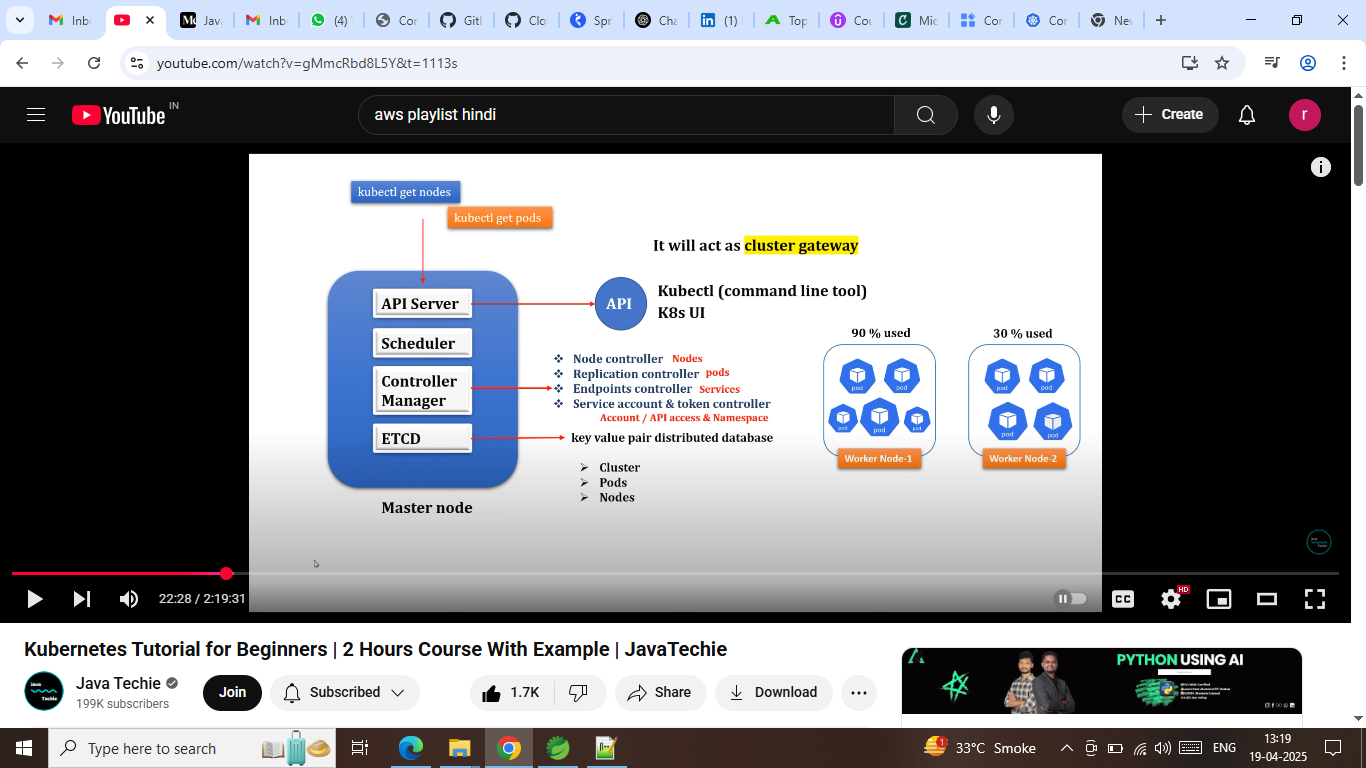
- Only the API Server communicates directly with etcd.

- Highly available and consistent (uses Raft protocol).

- Can be backed up and restored for disaster recovery.

Think of it as:

→ "Brain/memory of Kubernetes" – everything is saved here.



* 2. Worker Node :- it is a virtaual or physical machine which has pods in it
* components of worker node :-
* kubelet :- it is an agent running on each node , and kubelet communicate with master node using API server
* kubelet ensures that conatiners describe under prodsec are running healthy
* kube-proxy : - it is an network agent
* conatiner-runtime

