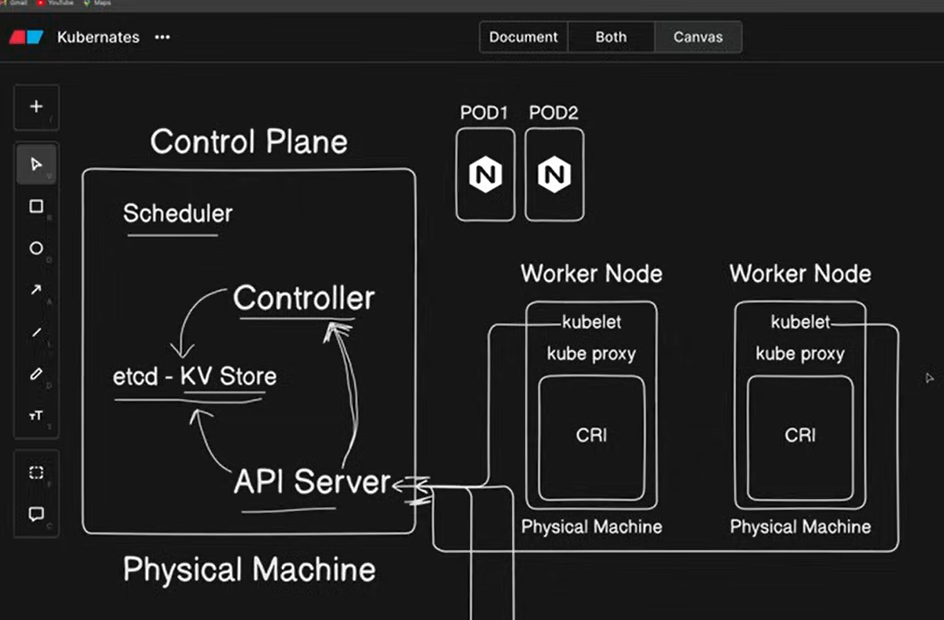
**Kubernetes :-**

[**https://www.youtube.com/watch?v=a-nWPre5QYI**](https://www.youtube.com/watch?v=a-nWPre5QYI)

****

**It is also called as k8s.**

It is open source system for **Automatic deployment, management of the containerized application, scaling**

Support different infrastructure :- virtual machine OR cloud.

1. What is container orchestration?

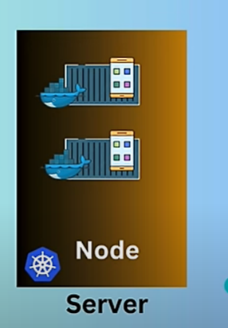


Architecture of Kubernetes :-

When you deploy kubernetes, you get a cluster.

Two important parts are :-

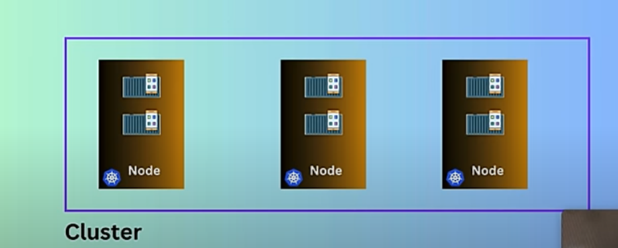
1. Master (control plane)
2. Worker Node

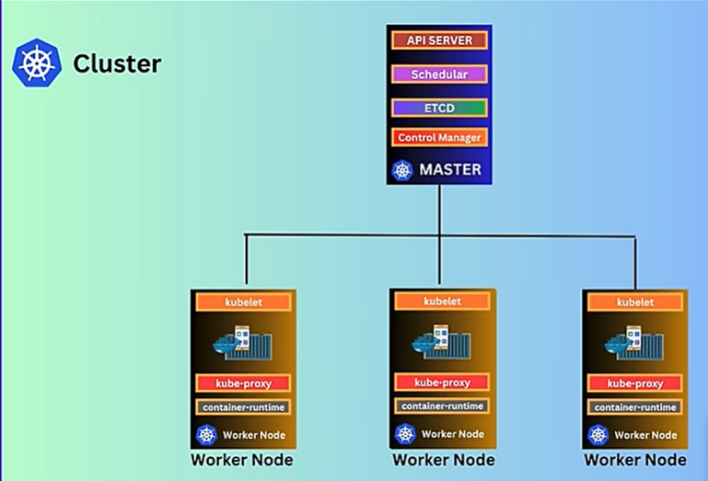


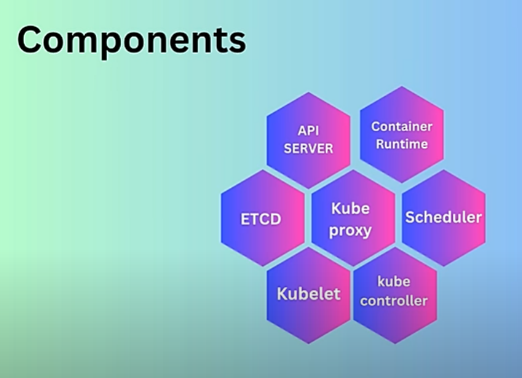
**Node (Minion) :-** itis one type of server anywhere in cloud or VM.

**Cluster :-** it is collection of Nodes.

We need to manage these nodes. We have master to manages these nodes.

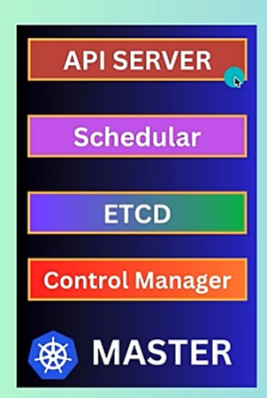




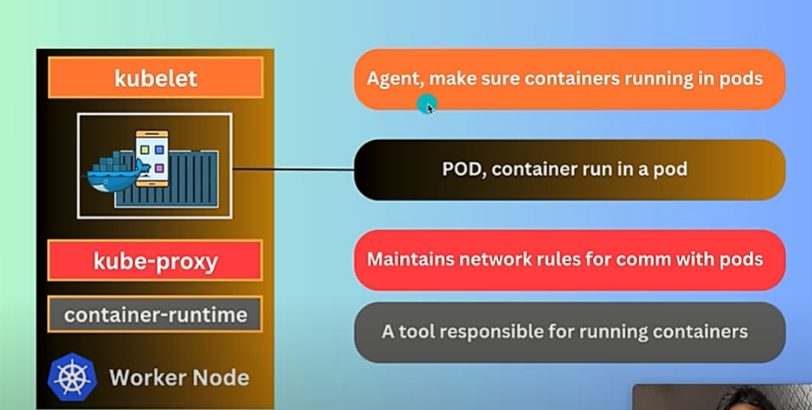
.

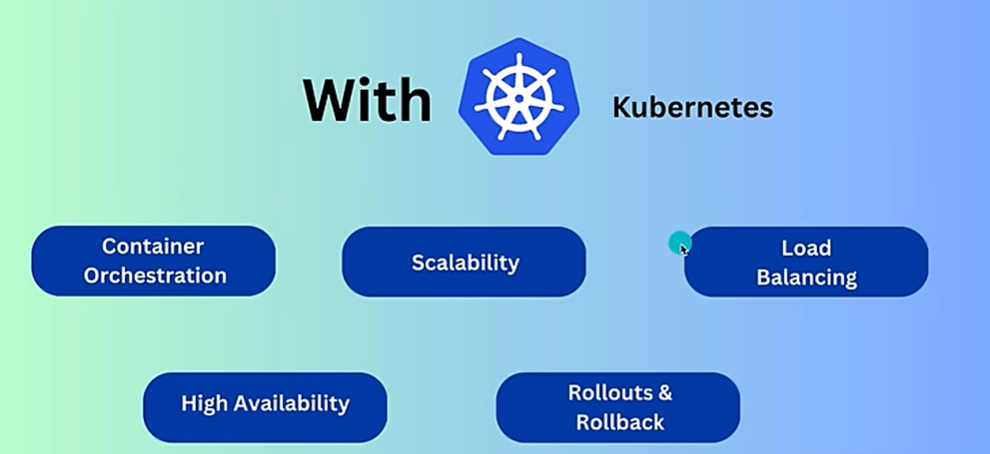
1. What is pod ?

* It is single instance of running process in the cluster.
* It can run one or more container and share the same resources.

1. Components of the master node.

* **API Server :-** it provides commandline interface [cube control]

1. Components of worker nodes.
2. Features of kubernetes



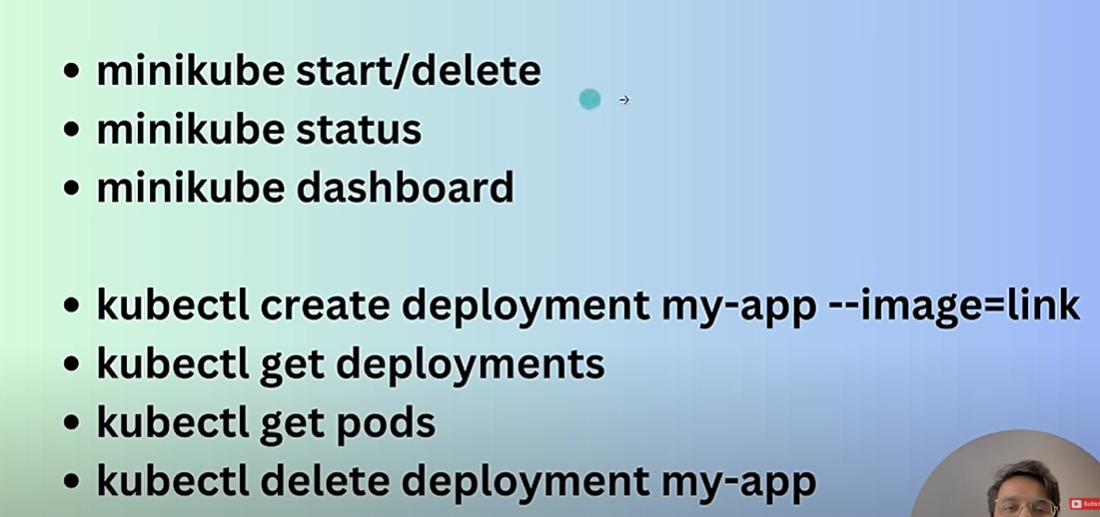
1. Installation ks8 on windows

<https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/>

install using chocolatey.

First we have to install chocolatey on our system. Run Following command in Powershell [Run as admin]

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))



Now we will create :- PODS

We will first start any docker container

I have create image of my JWT-Auth

1. Build JWT-Auth project :- **./mvnw clean package -DskipTests**
2. Create Dockerfile and following command in this file

# Use lightweight OpenJDK 17 image  
FROM openjdk:17-jdk-slim  
  
# Set working directory inside container  
WORKDIR /app  
  
# Copy the WAR file into the container  
COPY target/JwtAuthenticationServer.war app.war  
  
# Expose the port (change if your app runs on a different one)  
EXPOSE 8080  
  
# Run the WAR file  
ENTRYPOINT ["java", "-jar", "app.war"]

**OR**

2. docker-compose.yml file [instead of creating 2 images – mongodb + jwt-auth image] – create one docker compose file

And add the following command

version: "3.8"  
  
services:  
 mongodb:  
 image: mongo:6.0  
 container\_name: my-mongo  
 ports:  
 - "27017:27017"  
 volumes:  
 - mongo\_data:/data/db  
  
 app:  
 build: .  
 container\_name: auth-server  
 ports:  
 - "8082:8082"  
 depends\_on:  
 - mongodb  
 environment:  
 SPRING\_DATA\_MONGODB\_URI: mongodb://mongodb:27017/ADServer  
  
volumes:  
 mongo\_data:

**docker-compose up -d [it create container which contain mongodb and jwt auth server]**

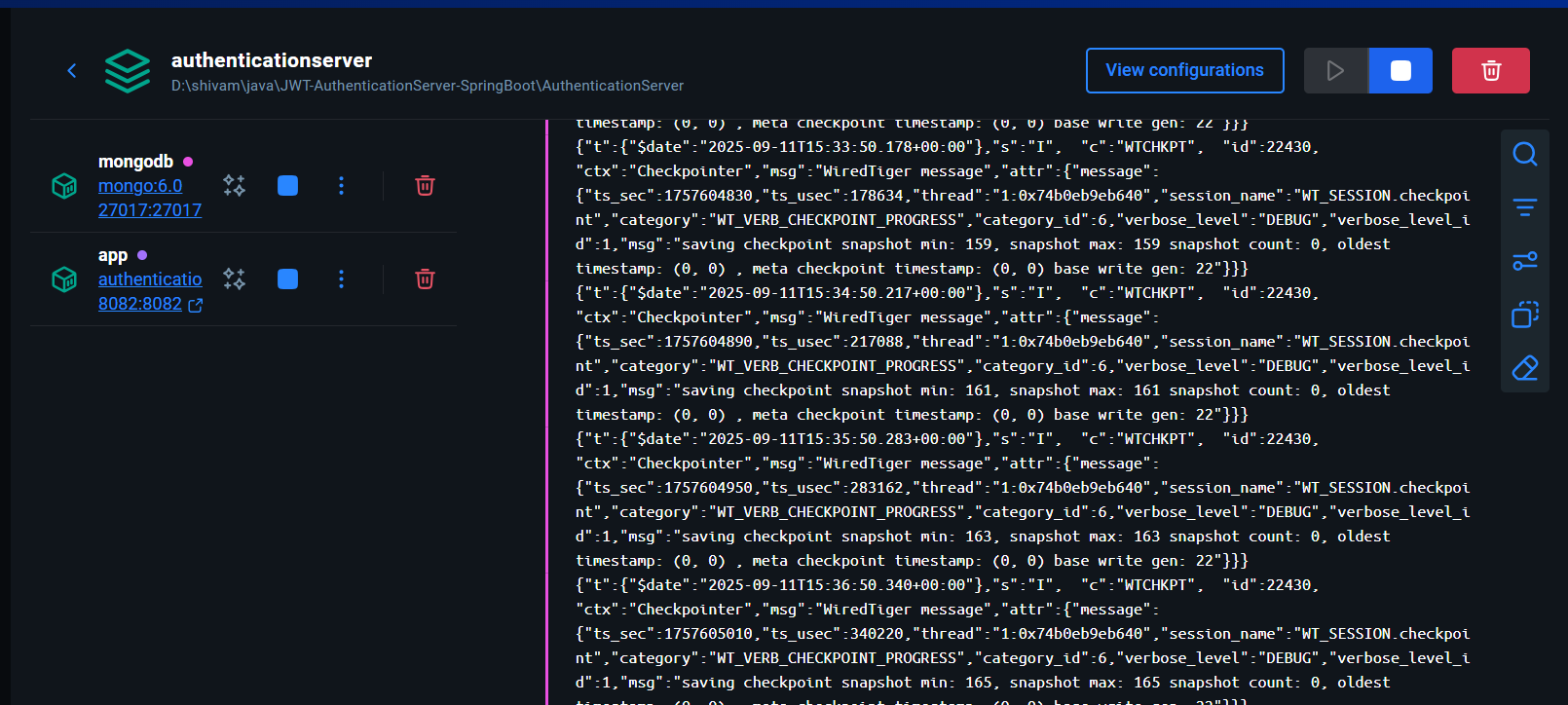
**docker tag local-image:tagname new-repo:tagname**

**docker push new-repo:tagname**

**docker tag authenticationserver-app:latest docker-username/authenticationserver-app:v1**

**docker tag authenticationserver-app:latest shivampatel612/jwt-auth-repo:v1**

**docker push shivampatel612/jwt-auth-repo:v1**

****

**After pushing the image to docker, we will deploy it into minikube Kubernetes**

**minikube status**

**kubectl command is used to create pods**

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

docker tag authenticationserver-app:latest jwt-auth-image:v1

docker push shivampatel612/jwt-auth-image:v1

kubectl create deployment jwt-auth-server --image= shivampatel612/jwt-auth-image:v1

kubectl delete -n default deployment jwt-auth-server

kubectl delete deployment jwt-auth-server

kubectl get deployments

kubectl get pods

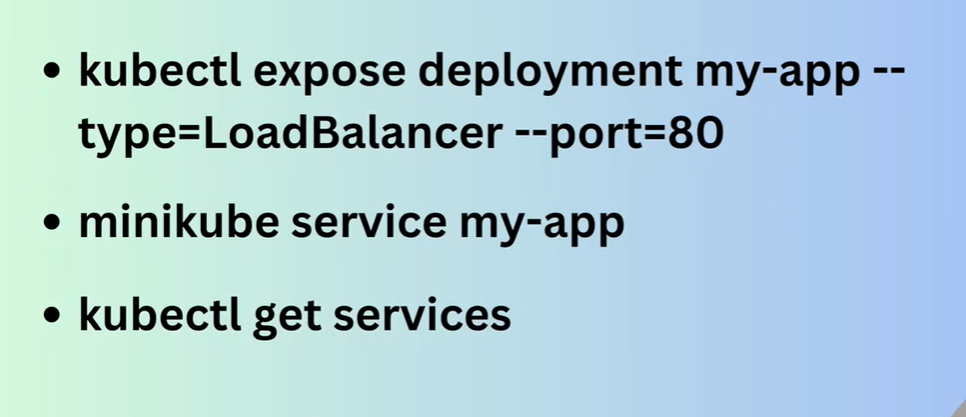
**kubectl expose deployment authenticationserver-app --port=8082 –type=LoadBalancer**

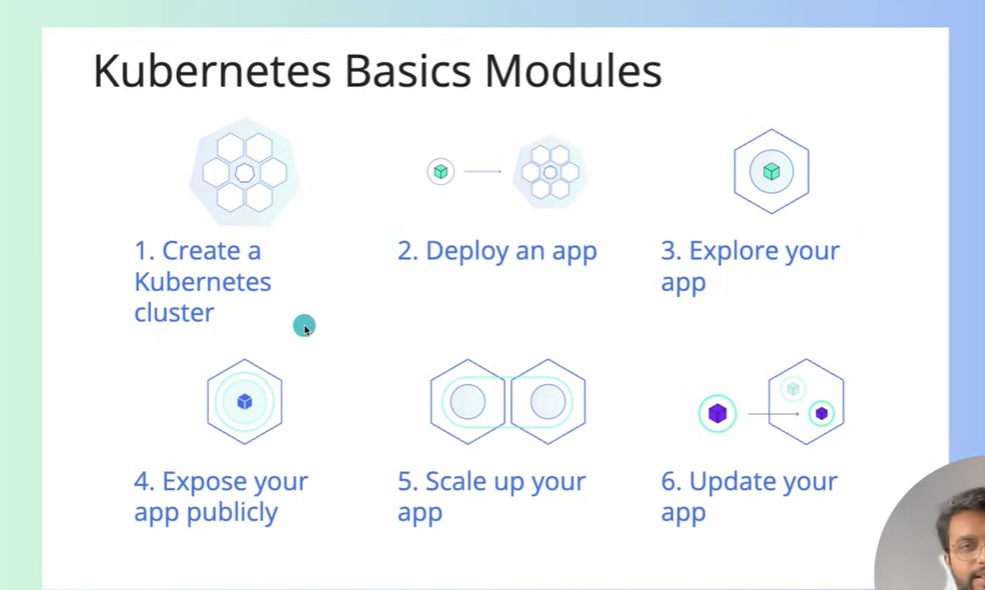
**kubectl get services**

**minikube service <deployment-name>**

**minikube service authenticationserver-app**







Now, if we have made change in our app and want to run that version of the app in the pods then we have to build the image, push that new image to docker hub then we have to update the kubectl image means our deployment is pointing the new image

Command :-

kubectl set image deployment <deployment-name> <container-name>=<image-name> :version

**kubectl get services**

**minikube service authenticationserver-app**

Rollback

kubectl rollout status deployment <deployment-name>

kubectl rollout undo deployment <deployment-name>

**Scaling out application**

If there a heavy traffic in our app then kubernete automatically creates multiple instance according to requirement

kubectl scale deployment jwt-auth-server --replicas=4

as we have multiple pods, if any instance of application is down, other pod restart our application and there is not down time.

Yaml configuration file :- we use yml file because we don’t have to write many command and to create deployment replicas just we give configurations in yml file and run the following command to create deployment

Create yml:- you can create yml file any where with any name.

kubectl apply -f location-of-the-file/filename.yml

kubectl apply -f Deployment-spring-app.yml

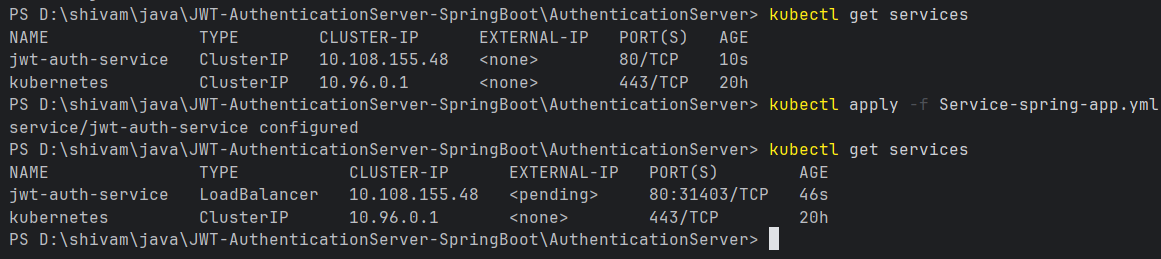
You can easily update and again apply the above command [if you want to upgrade or downgrade replicas]

apiVersion: *apps/v1*kind: *Deployment*metadata:  
 name: jwt-auth-server-deployment  
 labels:  
 app: jwt-auth-app  
spec:  
 replicas: 2  
 selector:  
 matchLabels:  
 app: jwt-auth-app  
 template:  
 metadata:  
 labels:  
 app: jwt-auth-app  
 spec:  
 containers:  
 - name: jwt-auth-continer  
 image: shivapatel1102001/jwt-auth-image:v1  
 ports:  
 - containerPort: 80

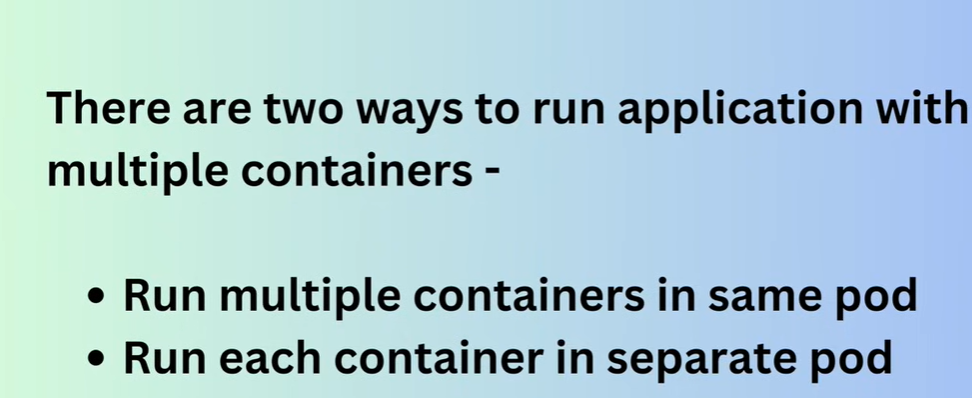
Now we will create service yml file :- service-spring-app.yml

kubectl apply -f Service-spring-app.yml

apiVersion: *v1*kind: *Service*metadata:  
 name: jwt-auth-service  
spec:  
 selector:  
 app.kubernetes.io/name: jwt-auth-app  
 type: LoadBalancer  
 ports:  
 - protocol: TCP  
 port: 80  
 targetPort: 8082



apiVersion: *v1*kind: *Service*metadata:  
 name: jwt-auth-service  
spec:  
 selector:  
 app.kubernetes.io/name: jwt-auth-app  
 type: LoadBalancer  
 ports:  
 - name: http  
 port: 8082  
 targetPort: 8082



1. Multiple container in same pod

Create deployment and service file.yml

We can also merge both the file using --- (three -)

And run command

kubectl apply -f deployment-file-name.yml

1. Same or each container in separate pods



**Persistence Volume :-** we will create 2 files

host-pv.yml - persistence volume

host-pvc.yml – persistence volume claim

