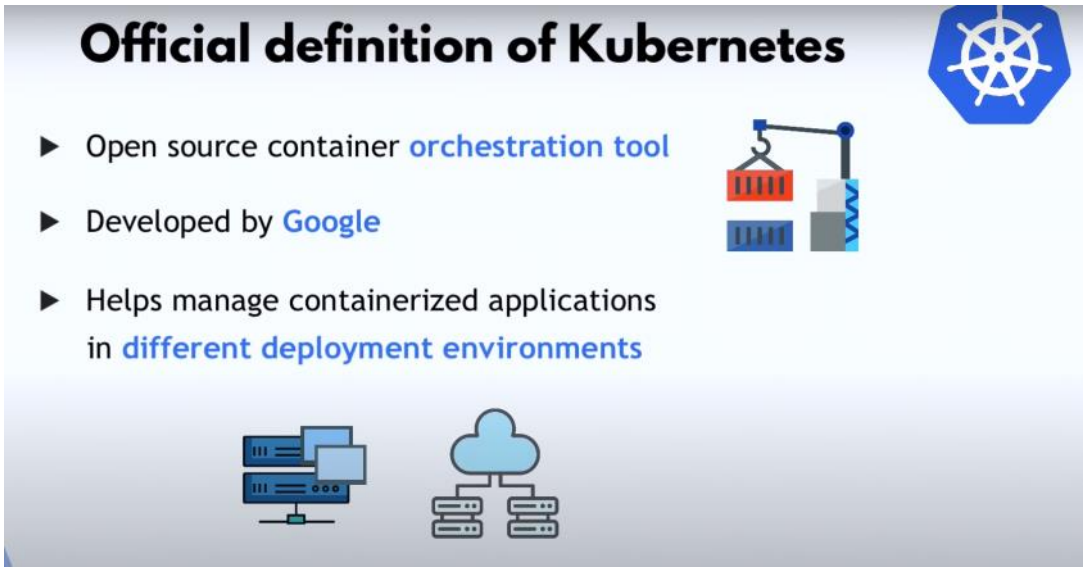
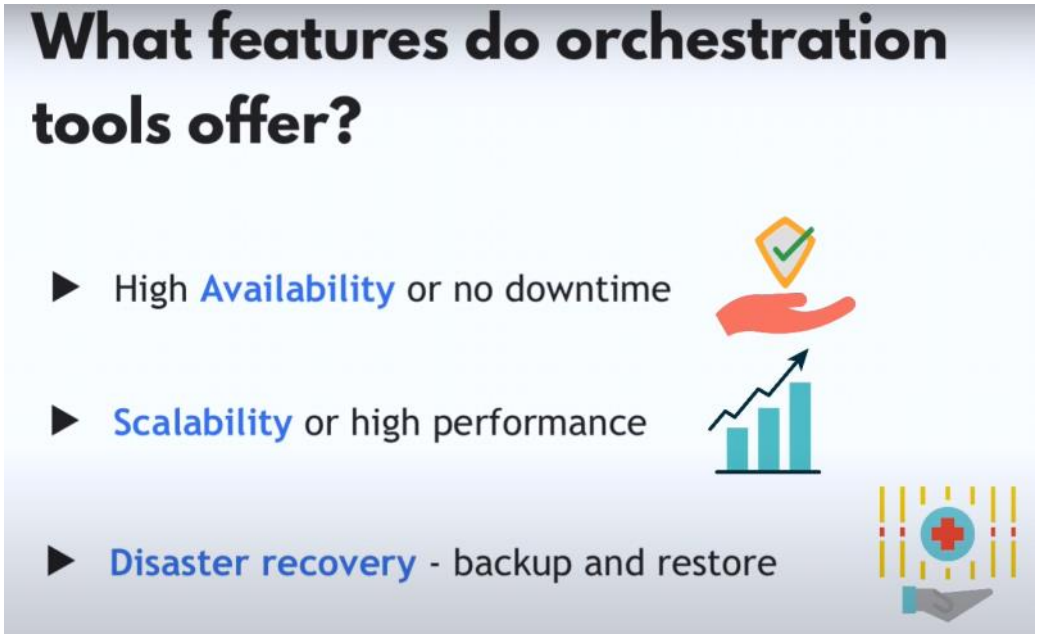
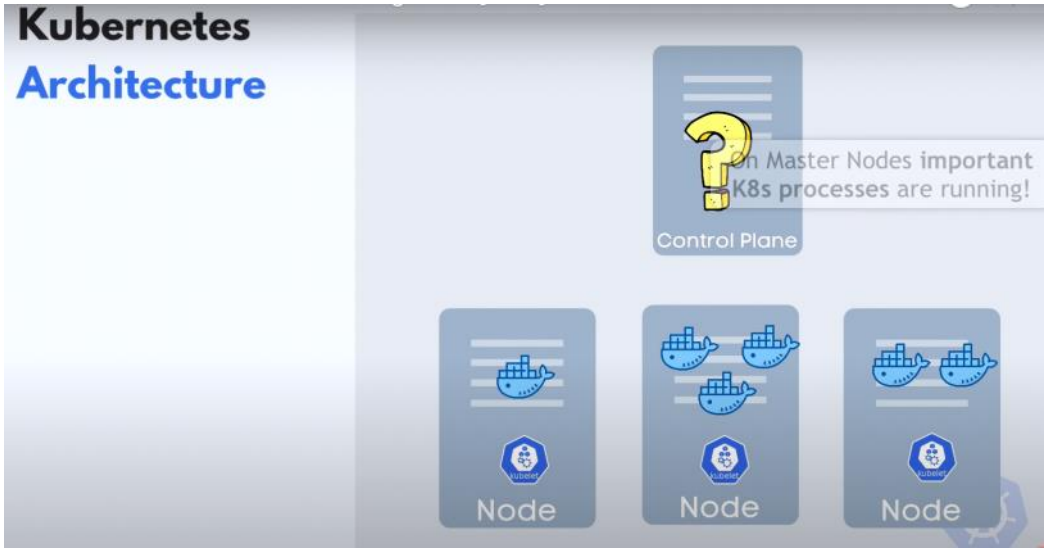
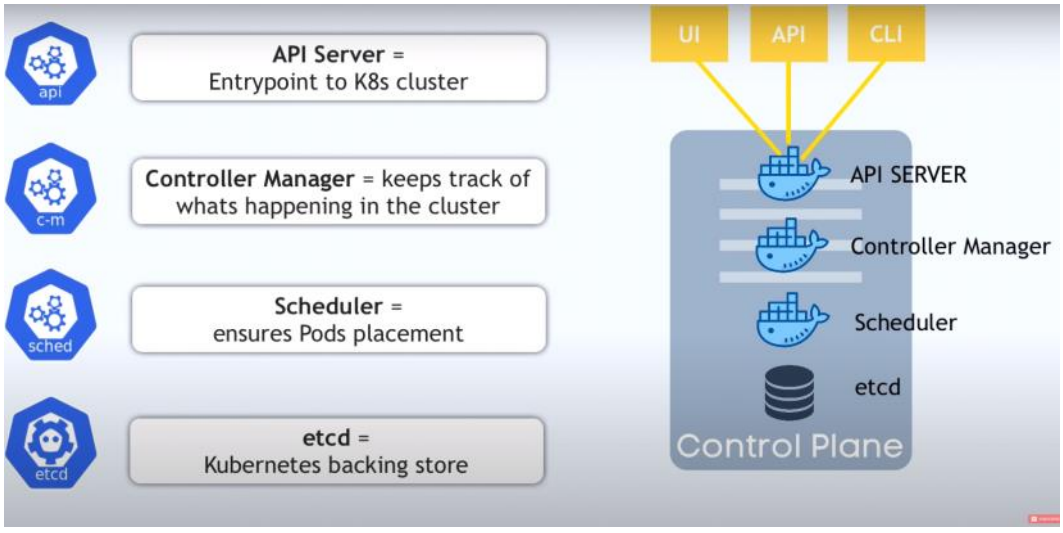
KUBERNETES

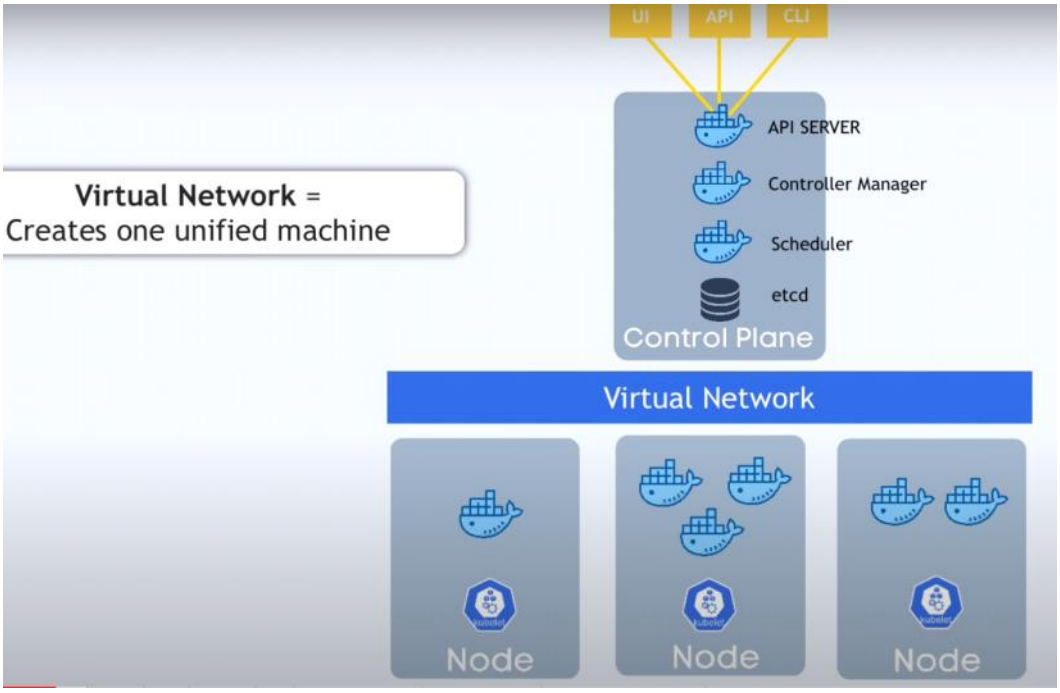
Tuesday, October 10, 2023 12:30 AM

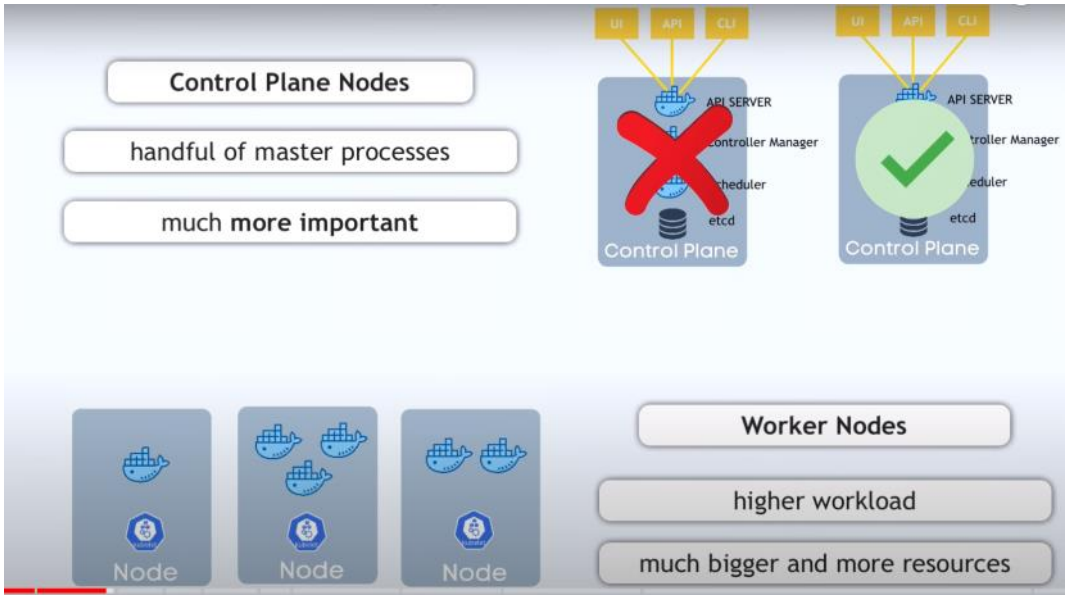


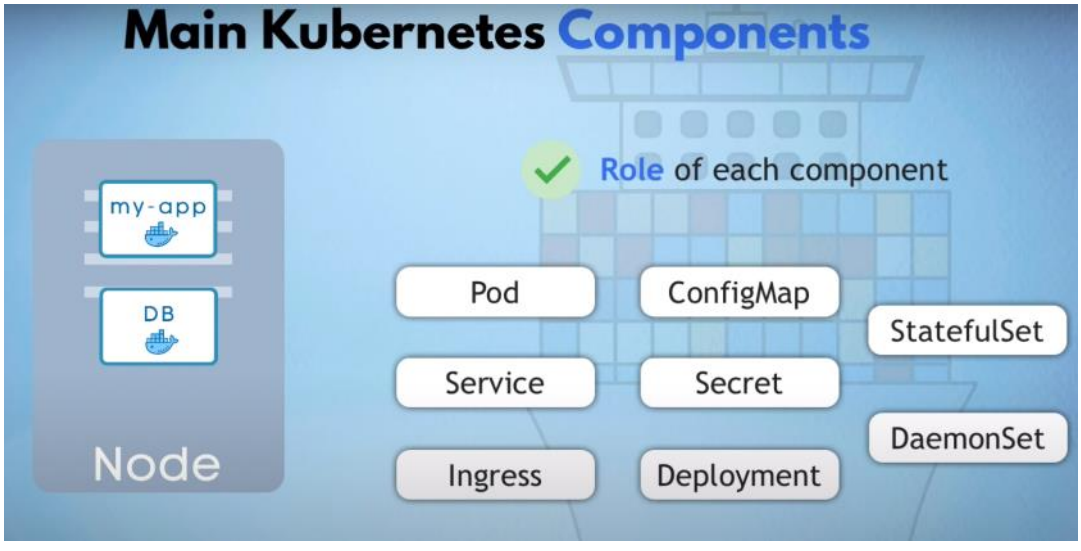

What features do orchestration tools offer?
High Availability or no downtime
Scalability or high performance
Disaster recovery - backup and restore

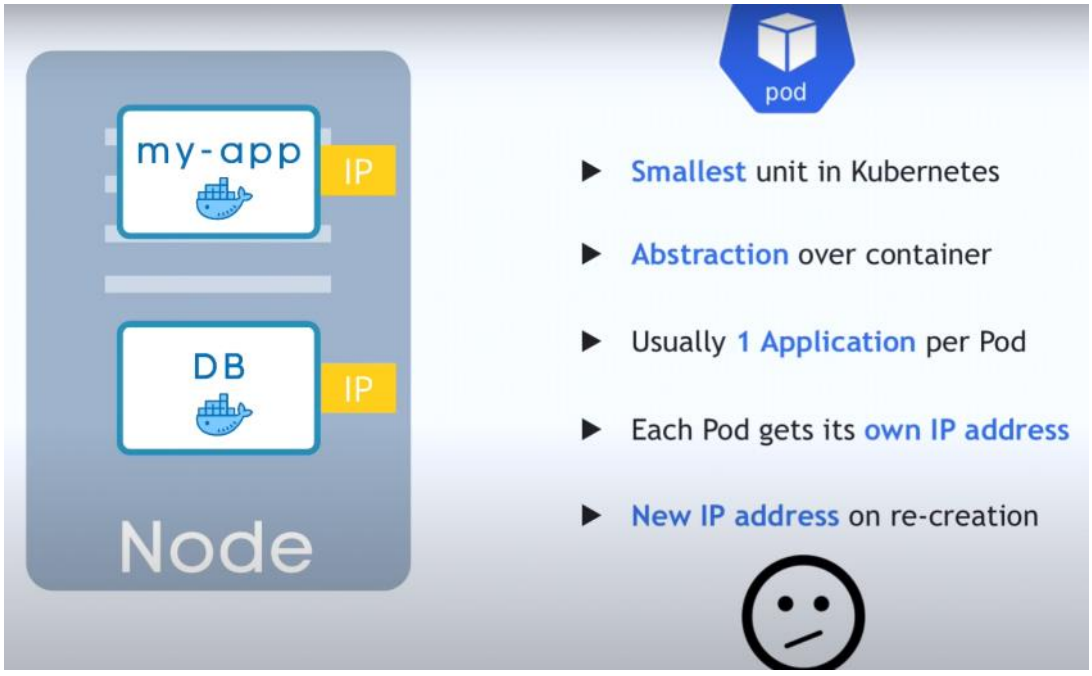

Kubernetes
Architecture
Control Plane
Node
Node
Node

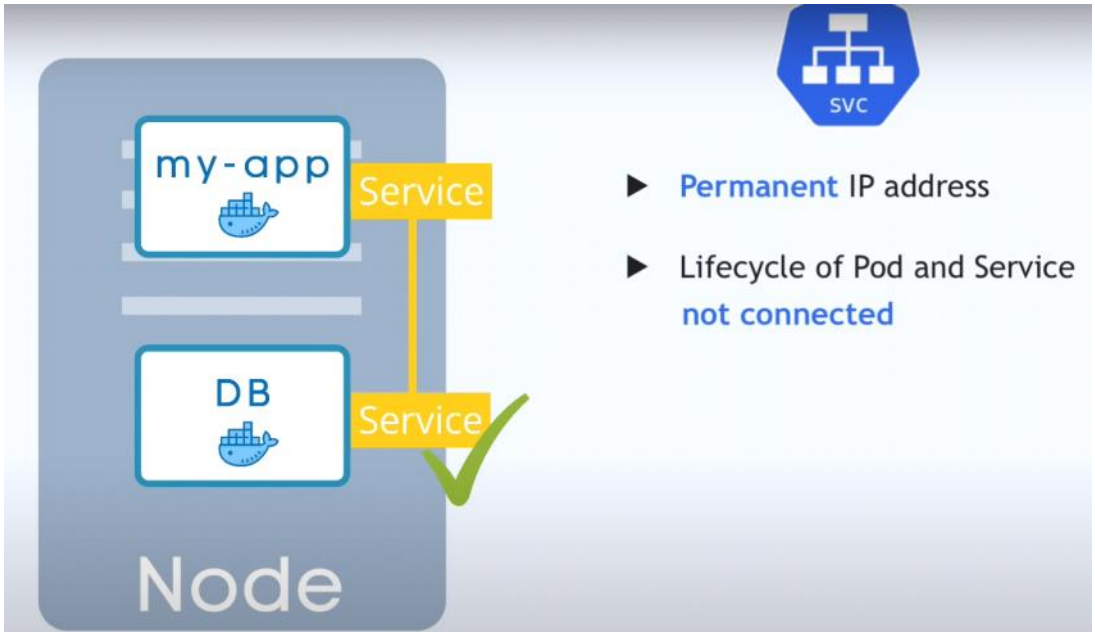

API Server = Entrypoint to K8s cluster
Controller Manager = keeps track of whats happening in the cluster
Scheduler = ensures Pods placement
etcd = Kubernetes backing store
UI
API
CLI
API SERVER
Controller Manager
Scheduler
etcd
Control Plane

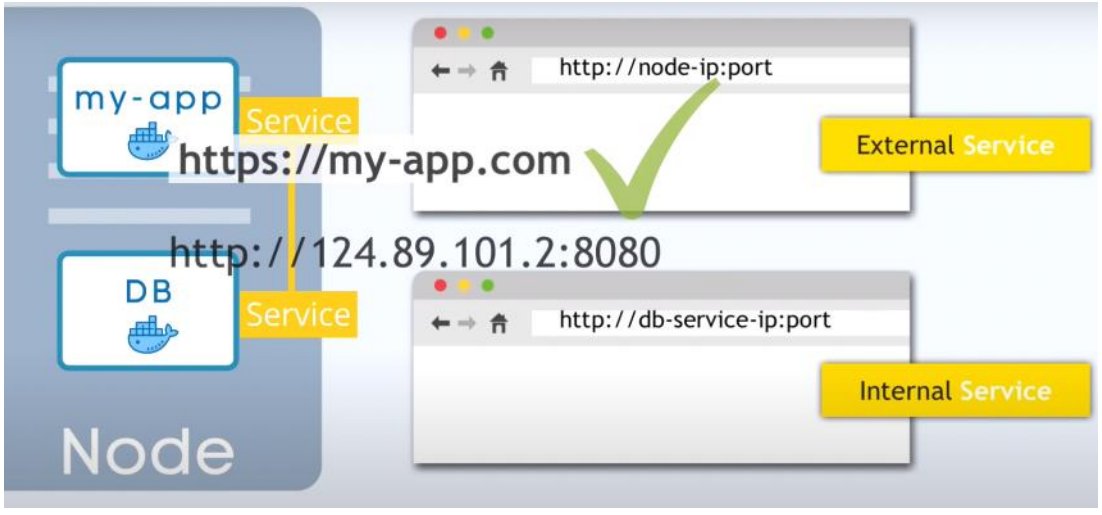

UI
API
CLI
API SERVER
Controller Manager
Scheduler
etcd
Control Plane
Virtual Network =
Creates one unified machine
Virtual Network
Node
Node
Node

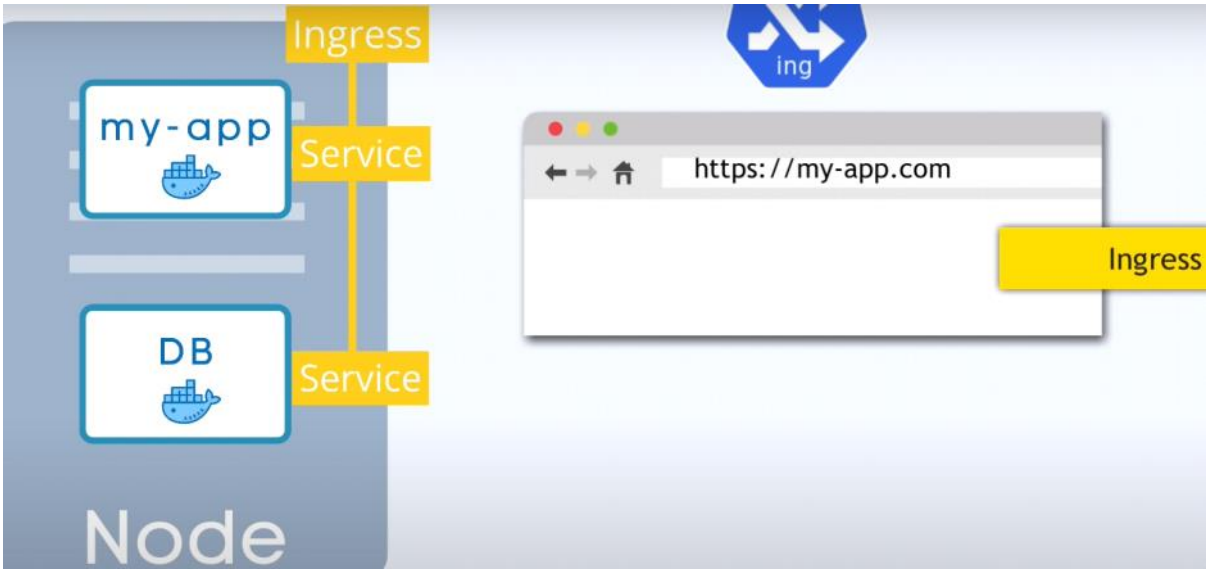

Control Plane Nodes
handful of master processes
much more important
UI
API
CLI
API SERVER
Controller Manager
Scheduler
etcd
Control Plane
Node
Node
Node
Worker Nodes
higher workload
much bigger and more resources

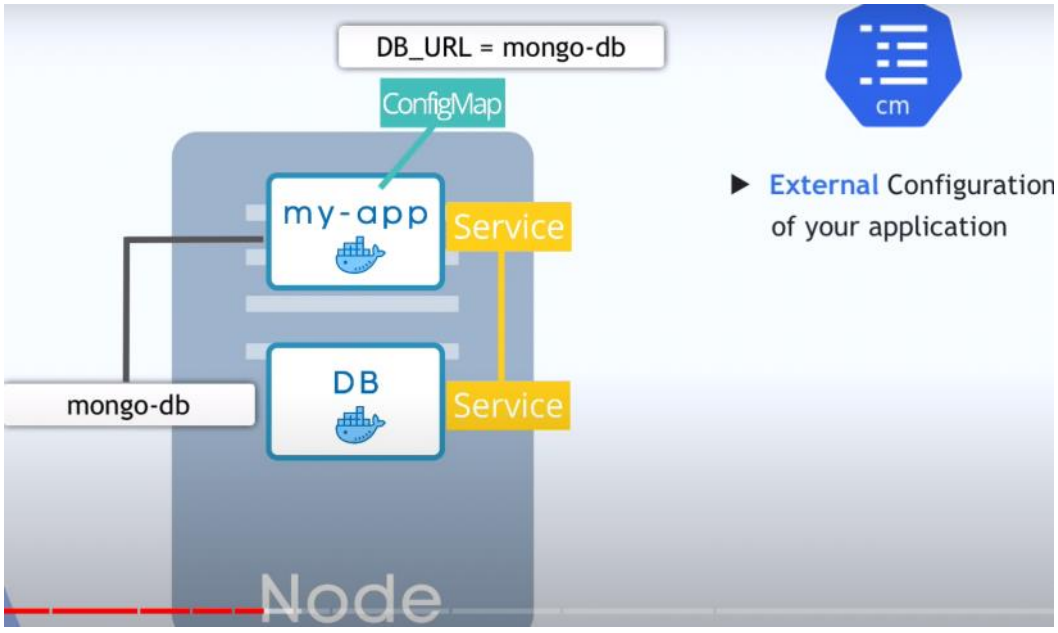

Main Kubernetes Components
Role of each component
my-app
DB
Node
Pod
ConfigMap
StatefulSet
Service
Secret
DaemonSet
Ingress
Deployment

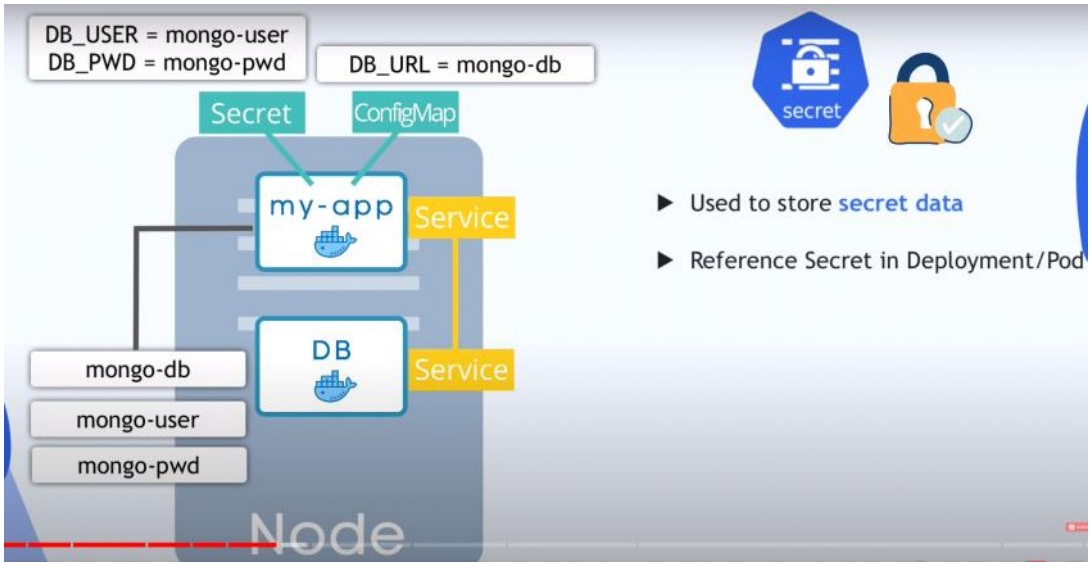

my-app
IP
DB
IP
Node
pod
Smallest unit in Kubernetes
Abstraction over container
Usually 1 Application per Pod
Each Pod gets its own IP address
New IP address on re-creation

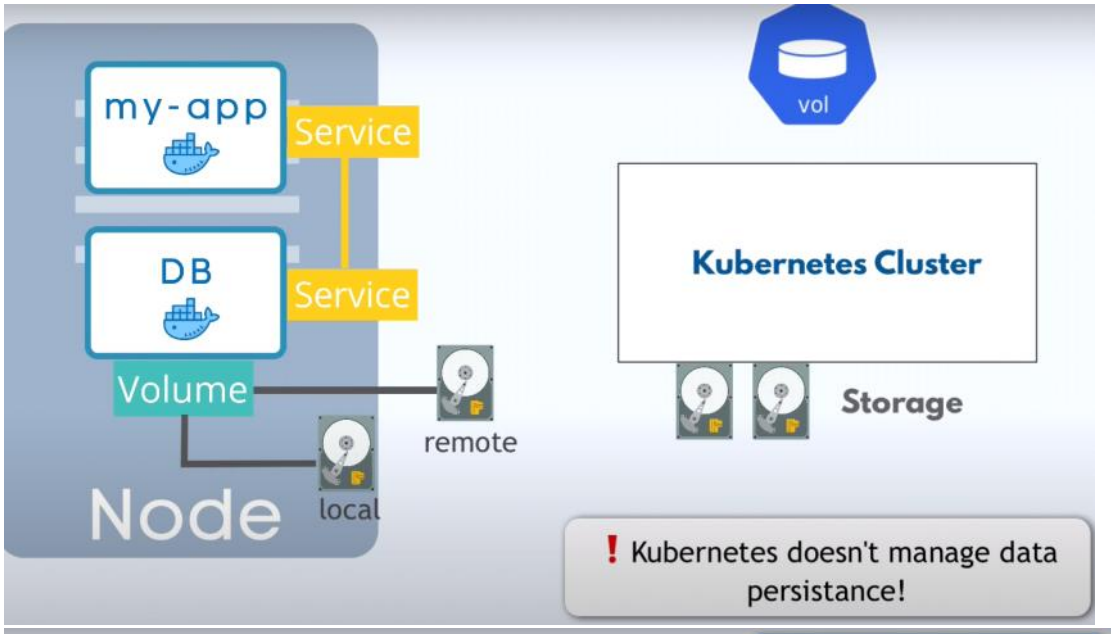

my-app
Service
DB
Service
Node
svc
Permanent IP address
Lifecycle of Pod and Service
not connected

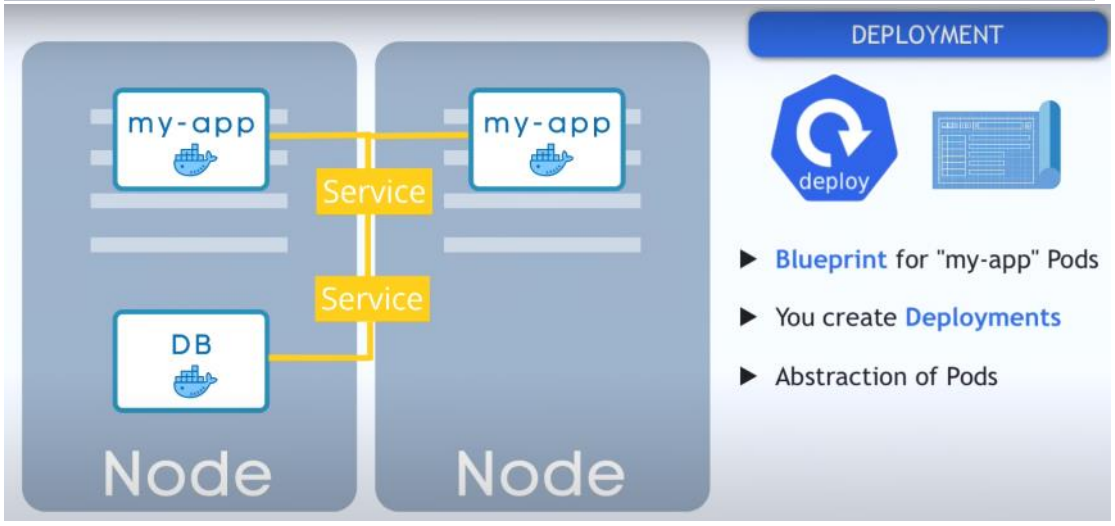

my-app
Service
https://my-app.com
DB
Service
http://124.89.101.2:8080
Node
http://node-ip:port
External Service
http://db-service-ip:port
Internal Service

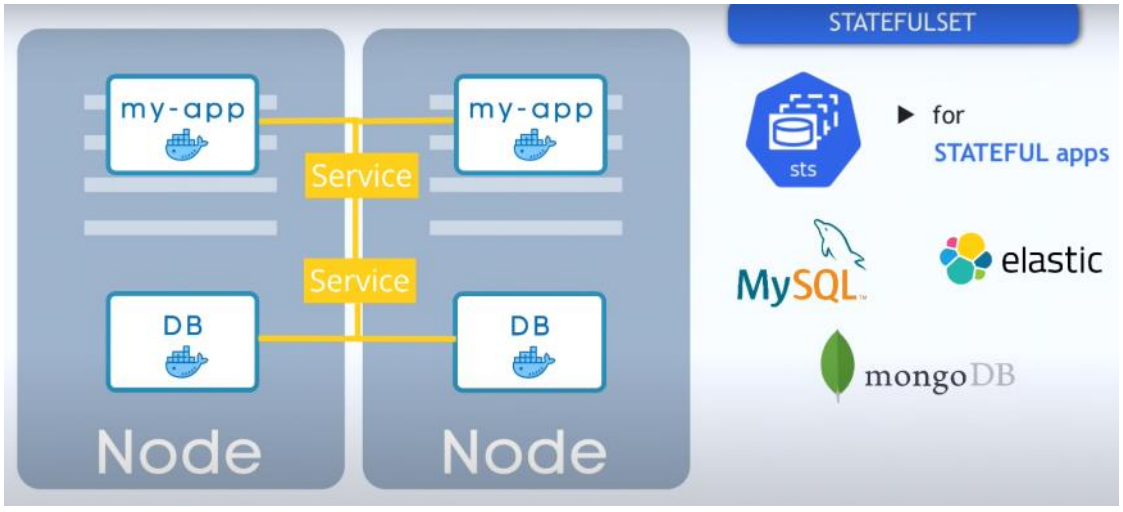

Ingress
my-app
Service
DB
Service
Node
ing
https://my-app.com
Ingress


DB_URL = mongo-db
ConfigMap
my-app
Service
DB
Service
mongo-db
Node
cm
External Configuration of your application

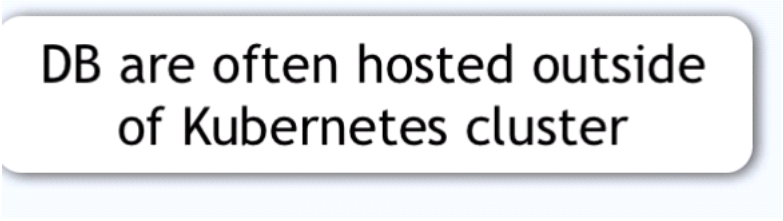

DB_USER = mongo-user
DB_PWD = mongo-pwd
DB_URL = mongo-db
Secret
ConfigMap
my-app
Service
DB
Service
mongo-db
mongo-user
mongo-pwd
Node
secret
Used to store secret data
Reference Secret in Deployment/Pod

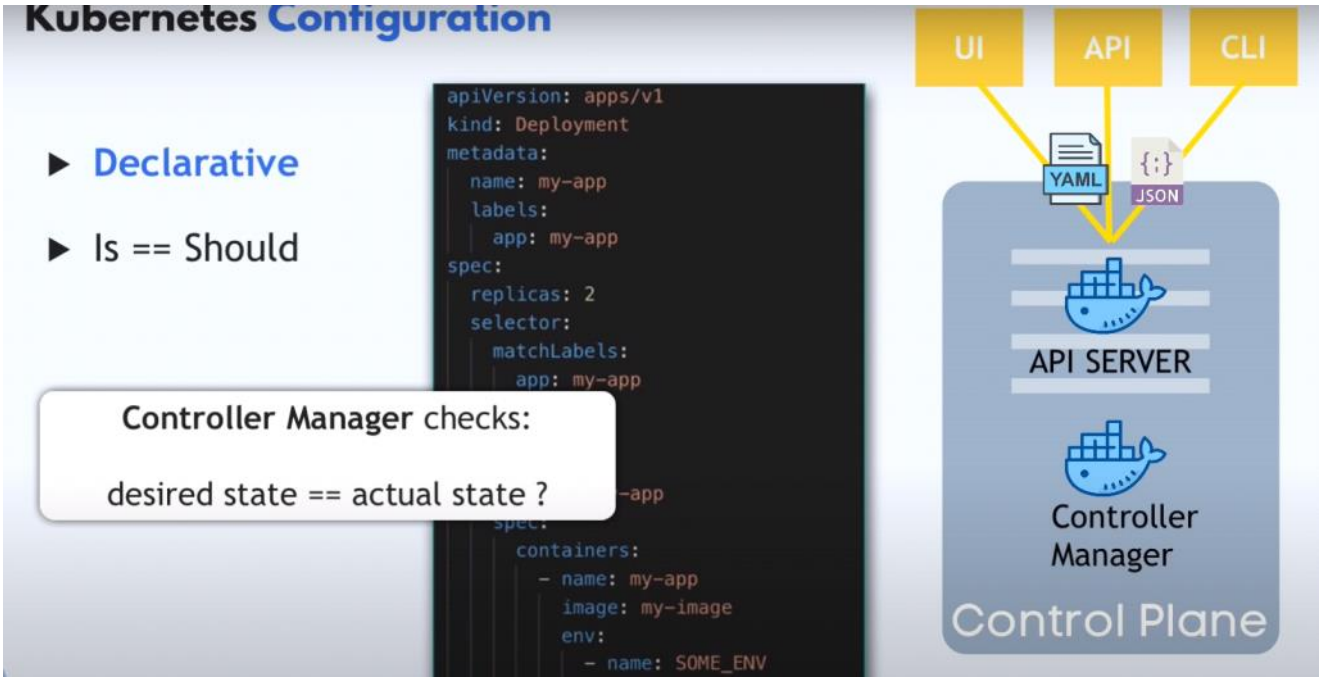

my-app
Service
DB
Service
Volume
remote
local
Node
vol
Kubernetes Cluster
Storage
! Kubernetes doesn't manage data persistance!

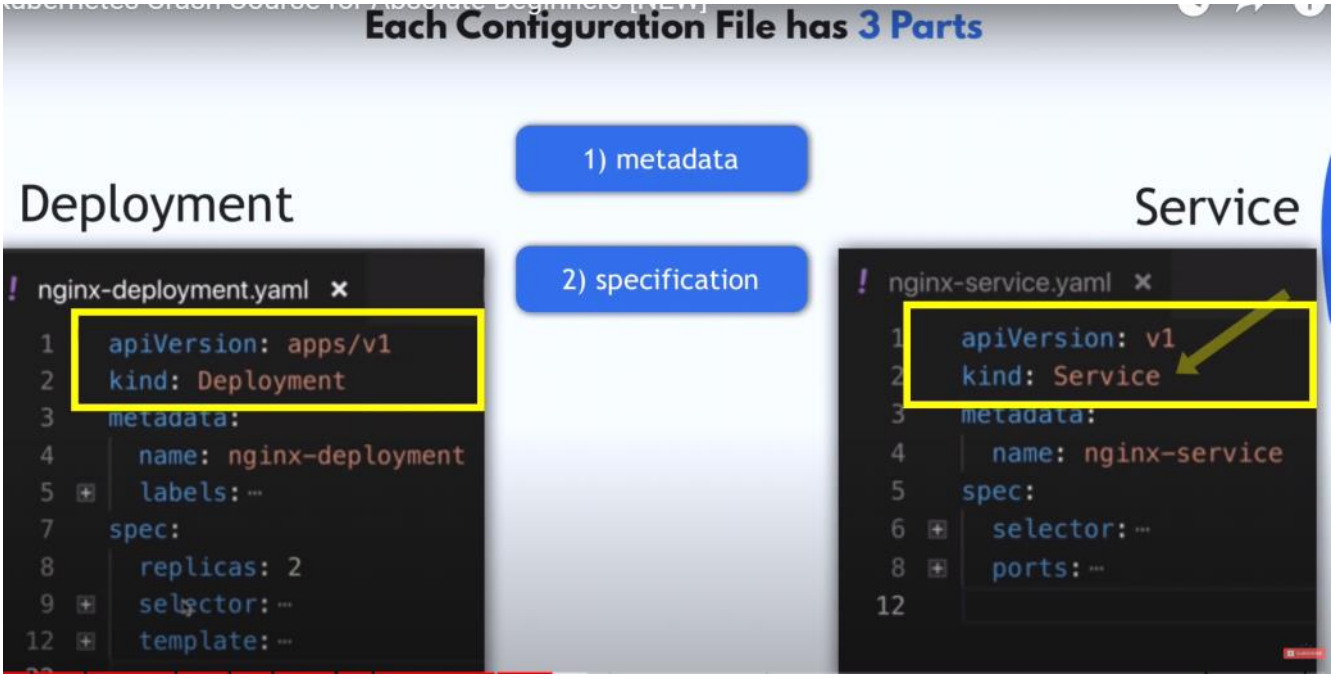

my-app
my-app
Service
Service
DB
Node
Node
DEPLOYMENT
deploy
Blueprint for "my-app" Pods
You create Deployments
Abstraction of Pods


my-app
my-app
Service
Service
DB
DB
Node
Node
STATEFULSET
sts
for
STATEFUL apps
MySQL
elastic
mongoDB

statefullSet is so as to avoid inconsistency in data







Each Configuration File has 3 Parts

1) metadata

2) specification

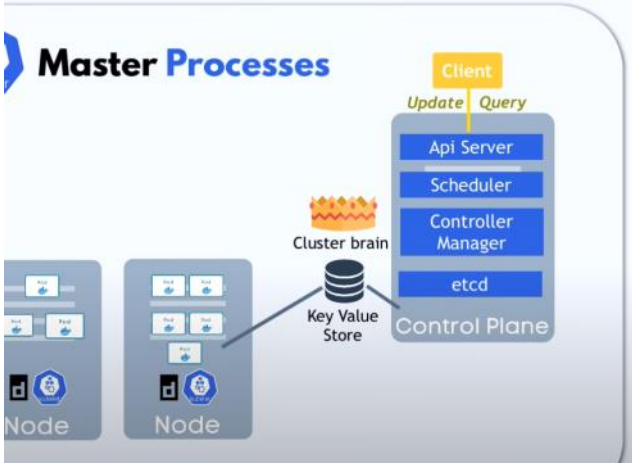
3) status

```
1   apiVersion: apps/v1
2   kind: Deployment
3   metadata:
4     name: nginx-deployment
5   labels: ⋯
    spec:
      replicas: 2
      selector: ⋯
      template: ⋯
22
```

3rd part: status

Desired? ≠ Actual?

Where does K8s get this status data?



Etcd holds the current status of any K8s component!

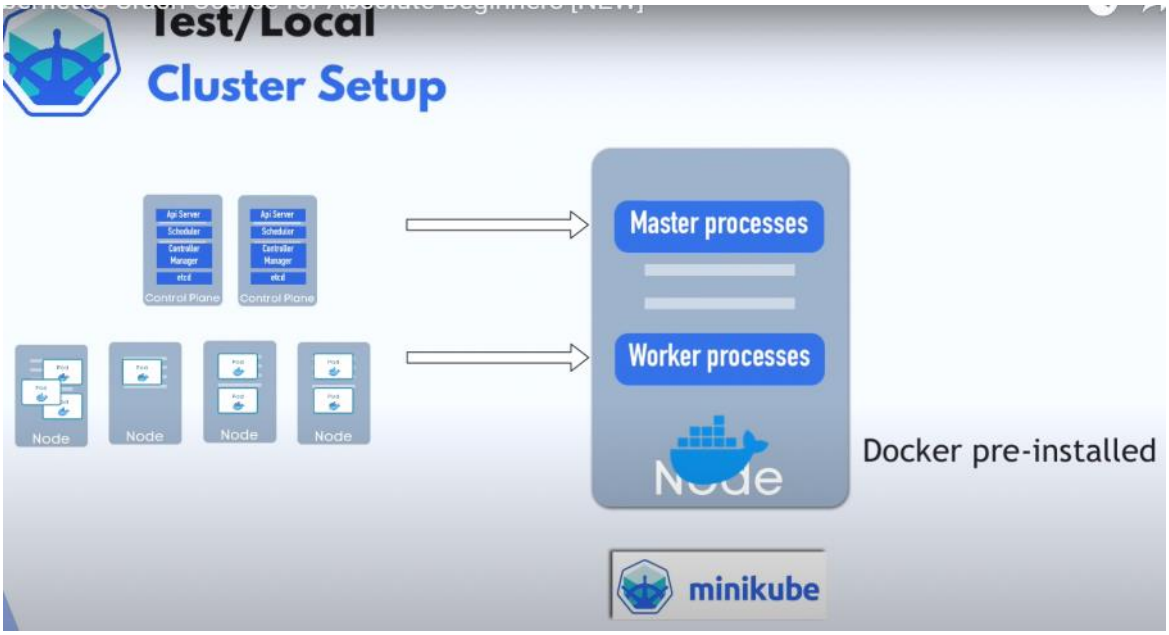
etcd

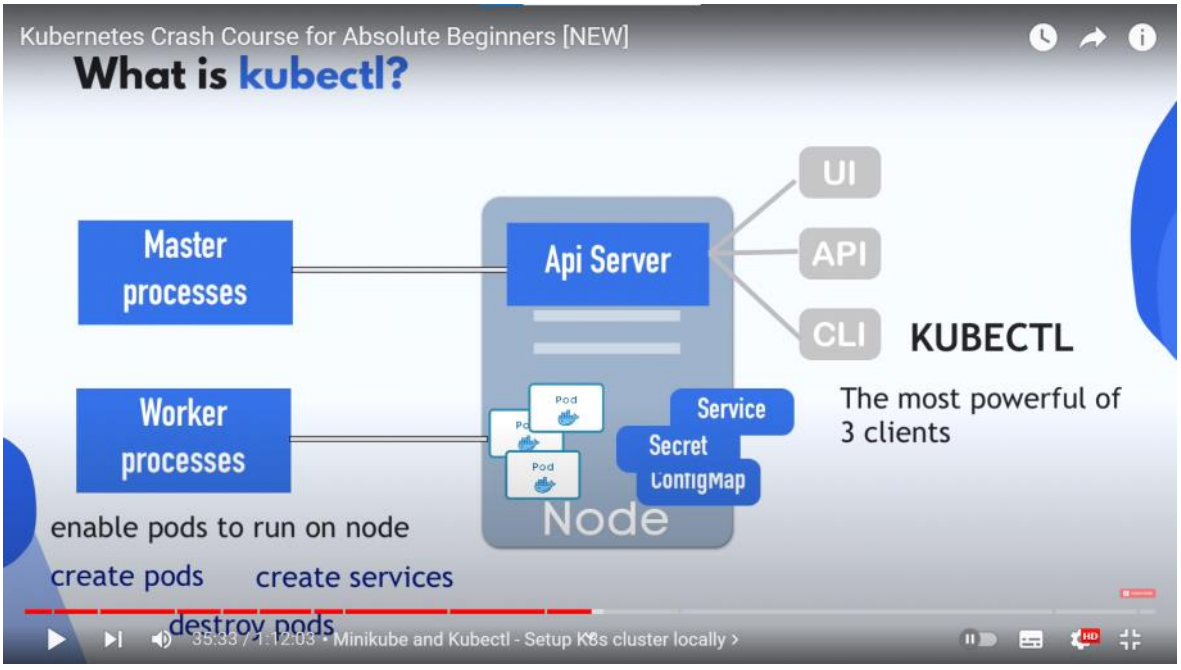
YAML Configuration Files

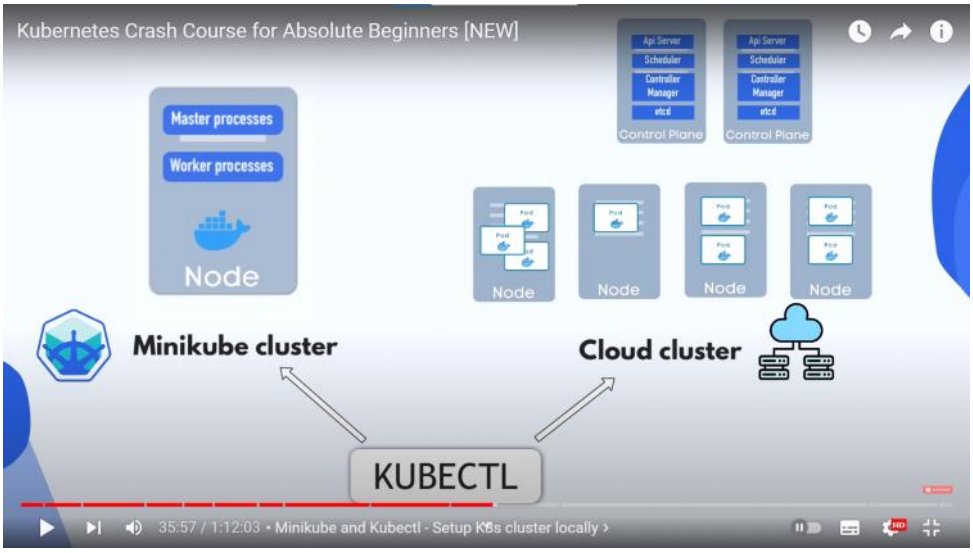
nginx-deployment.yaml

```
1   apiVersion: apps/v1
2   kind: Deployment
3   metadata:
4     name: nginx-deployment
5   labels: ⋯
7   spec:
8     replicas: 2
9   selector: ⋯
12  template: ⋯
22
```

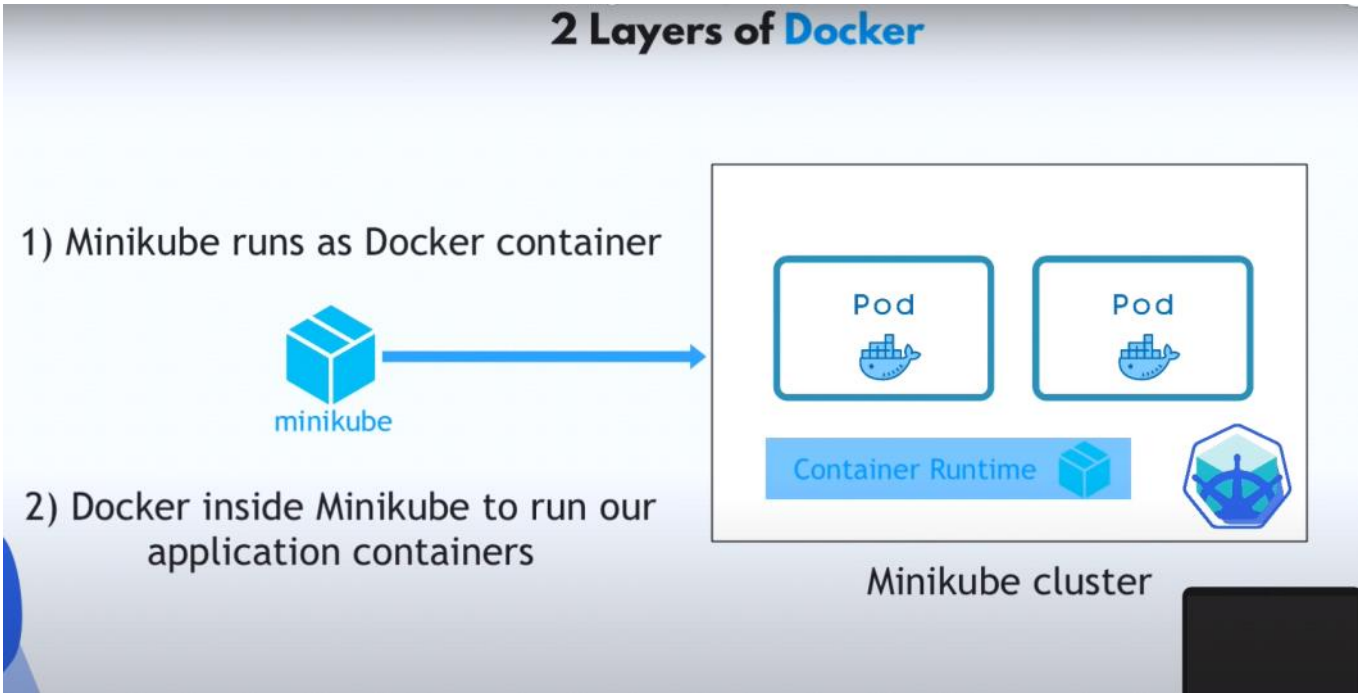
- "**human friendly**" data serialization standard for all programming languages
- syntax: **strict indentation!**
- store the config file with your code
- or own git repository


Test/Local
Cluster Setup
Api Server
Scheduler
Controller Manager
etcd
Control Plane
Node
Master processes
Worker processes
Docker pre-installed
minikube


Kubernetes Crash Course for Absolute Beginners [NEW]
What is kubectl?
Master processes
Worker processes
Api Server
UI
API
CLI
KUBECTL
The most powerful of 3 clients
Pod
Service
Secret
ConfigMap
Node
enable pods to run on node
create pods
create services
destroy pods
Minikube and Kubectl - Setup K8s cluster locally >



MINIKUBE LOCAL SETUP :



COMMANDS:
To start minikube container:
Minikube start
Or
Minikube start --driver docker

To check status:

Minikube status

To check nodes(kubectl gets preinstalled with minikube):
kubectl get node

To check pods
Kubectl get pod

To check everything
Kubectl get all

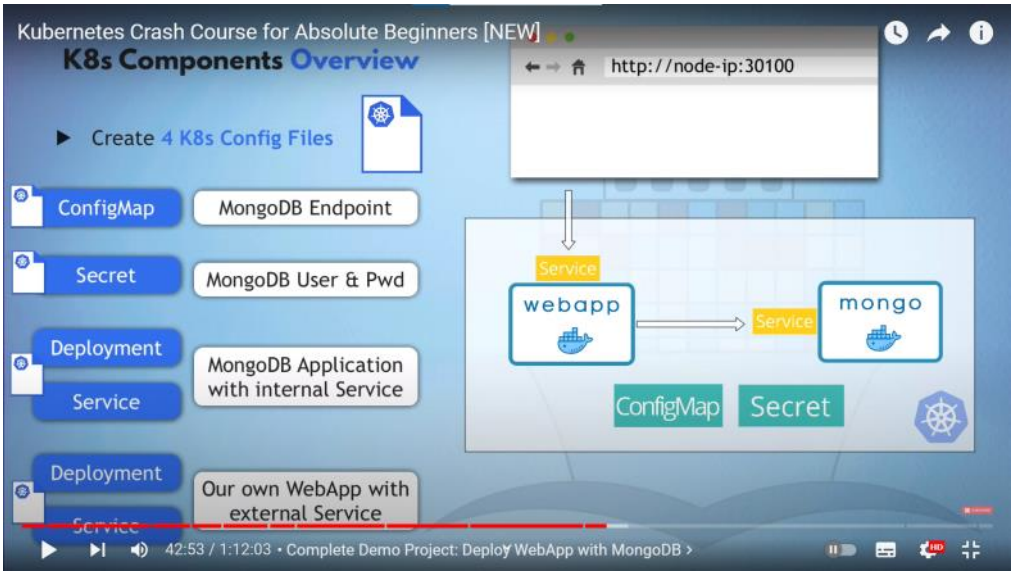
To get help:
Kubectl --help
Or
Kubectl get --help

To get details:
Kubectl describe service webapp-service
Or
Kubectl describe pod {podName}

To get logs:
Kubectl logs {podNameID}

To get minikube ip:
Minikube ip
Or
Kubectl get node -o wide

Taking any entire setup live using kubernetes:

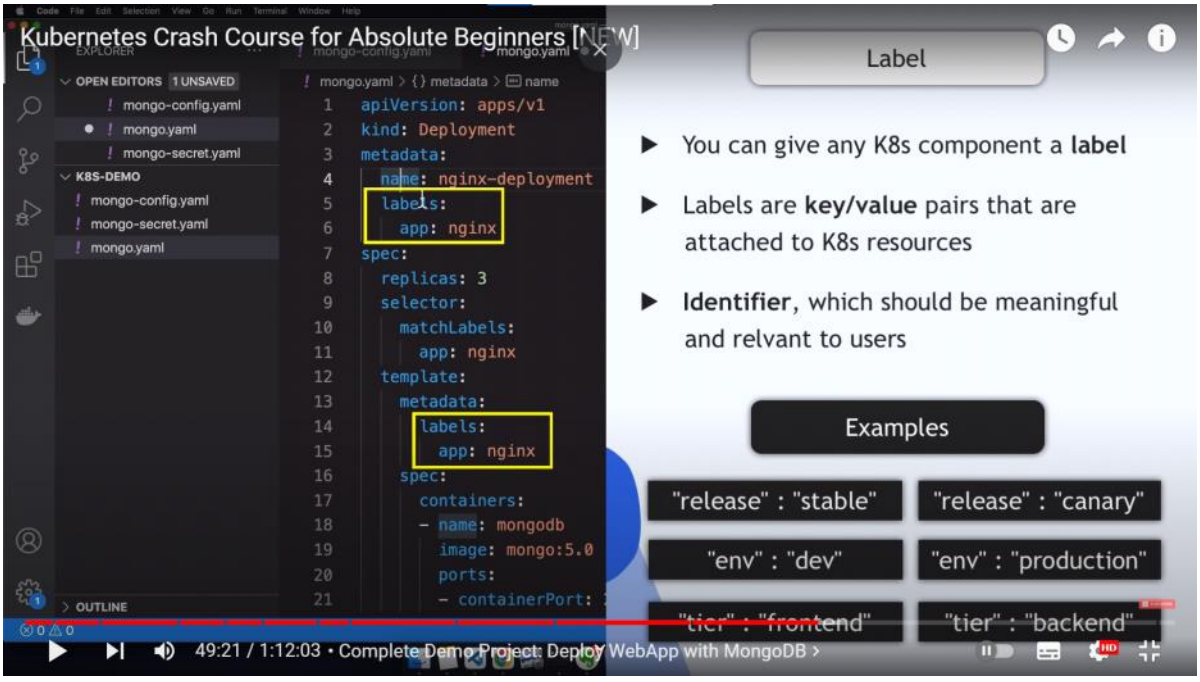


mongo-config.yaml

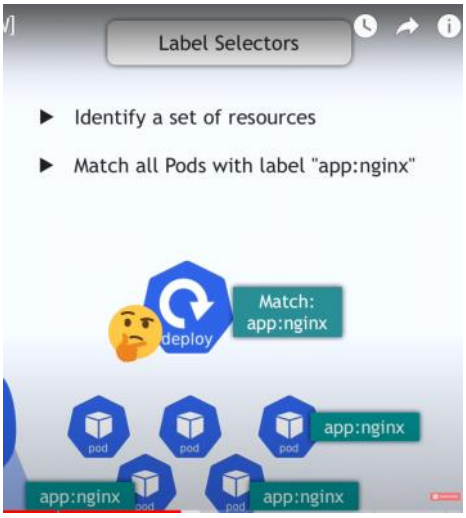
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mongo-config
data:
  mongo-url: mongo-service
```

mongo-secret.yaml

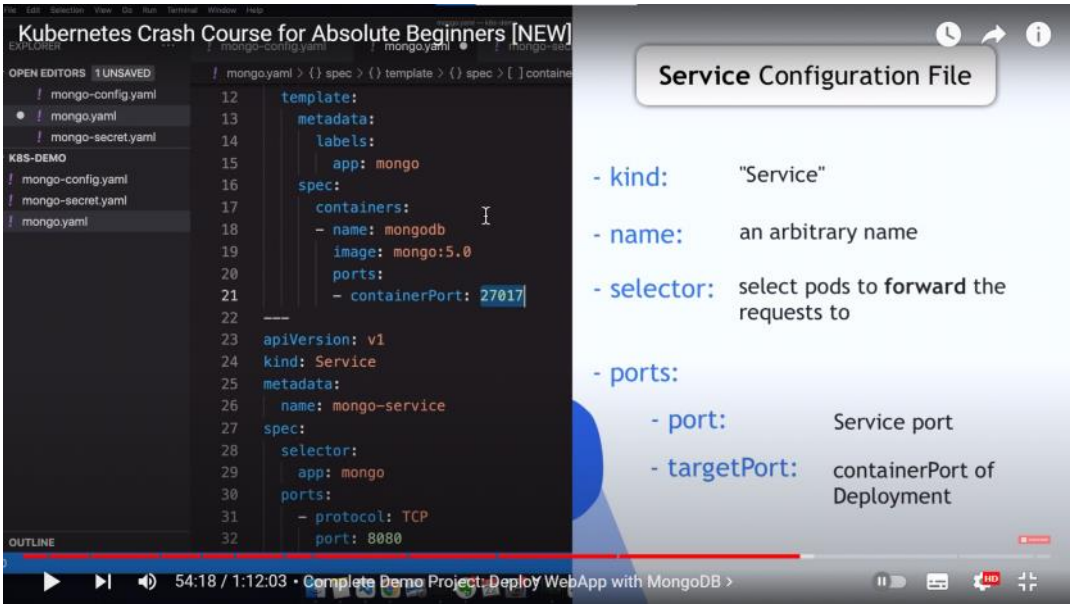
```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mongo-secret
type: Opaque
data:
  mongo-user: bW9uZ291c2Vy
  mongo-password: bW9uZ29wYXNzd29yZA==
```

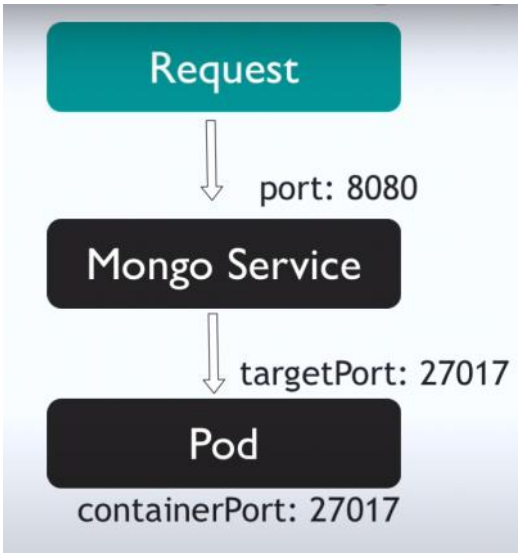

Kubernetes Crash Course for Absolute Beginners [NEW]
mongo.yaml
OPEN EDITORS 1 UNSAVED
mongo-config.yaml
mongo.yaml
mongo-secret.yaml
K8S-DEMO
mongo-config.yaml
mongo-secret.yaml
mongo.yaml
OUTLINE
mongo.yaml > {} metadata > name
apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx-deployment
labels:
app: nginx
spec:
replicas: 3
selector:
matchLabels:
app: nginx
template:
metadata:
labels:
app: nginx
spec:
containers:
- name: mongodb
image: mongo:5.0
ports:
- containerPort:
Label
You can give any K8s component a label
Labels are key/value pairs that are attached to K8s resources
Identifier, which should be meaningful and relvant to users
Examples
"release" : "stable"
"release" : "canary"
"env" : "dev"
"env" : "production"
"tier" : "frontend"
"tier" : "backend"
49:21 / 1:12:03 • Complete Demo Project: Deploy WebApp with MongoDB >


1 Use Case
Connecting Deployment to all Pod replicas
pod
pod
pod
mongo-101
mongo-111
mongo-210
app:nginx
app:nginx
app:nginx


Label Selectors
Identify a set of resources
Match all Pods with label "app:nginx"
Match:
app:nginx
deploy
pod
pod
pod
app:nginx
app:nginx
pod
pod
app:nginx


"---" =
You can have multiple YAML configurations within 1 file






```
env:
- name: MONGO_INITDB_ROOT_USERNAME
  valueFrom:
    secretKeyRef:
      name: mongo-secret
      key: mongo-user
- name: MONGO_INITDB_ROOT_PASSWORD
  valueFrom:
    secretKeyRef:
      name: mongo-secret
      key: mongo-password
```


env:
- name: USER_NAME
valueFrom:
secretKeyRef:
name: mongo-secret
key: mongo-user
- name: USER_PWD
valueFrom:
secretKeyRef:
name: mongo-secret
key: mongo-password
- name: DB_URL
valueFrom:
configMapKeyRef:
name: mongo-config
key: mongo-url