

K8S

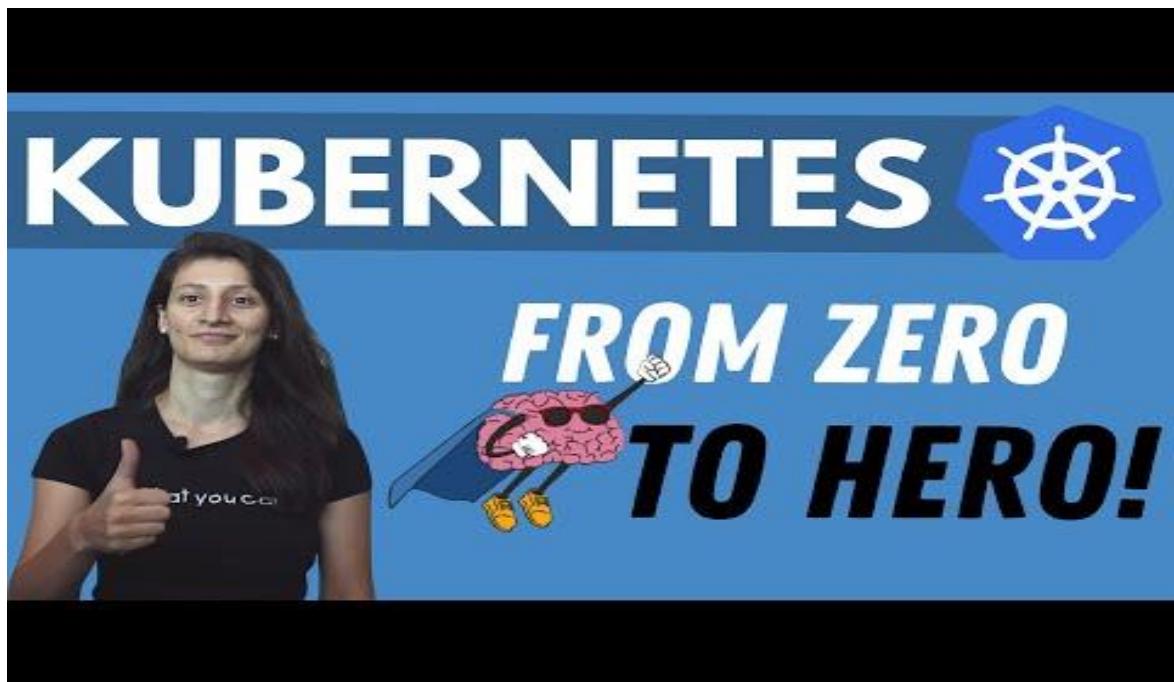
K8S	1
Fundamentals	2
Things to KNOW	3
SERVICE and INGRESS	3
ConfigMap.....	4
SECRETS.....	5
VOLUMES.....	5
Deployment and Statefull set	6
Architecture	8
Worker node	8
MASTER NODE	9
API Server.....	9
Scheduler	9
Controller Manager : detects cluster state changes	10
etcd : cluster store	10
CLUSTER	12
Minikube and kubectl local setup.....	13
API SERVER - Kubectl	13
Kubectl	13
Layers.....	13
deployment.....	14
replicaset.....	14
Debugging logs – same as in docker	15
Kubectl to use configuration file: apply & update	15
Configuration-file	16
Connect service to pods to deployment.....	18
Service listen on port 80 and forward to pod port 8080:.....	19
Get deployment status (from etcd): -o yaml.....	20
Demo on mongodb	21
Architecture	22
mongo.yaml deployment file;.....	23
Creating secrets:	24
Mongo db internal service inside same deployment file:	28

mongo express deployment , service and configMap:.....	30
Mongo-express.yaml config file:.....	30
mongo-express.yml with configmap:	33
Mongo express external service	34
Execution flow	36
Namespaces	36
Ingress.....	42
Implementation	47
Additional use cases for ingress	50
SSL configuration	51
Persistant Data using VOLUMES.....	53
DIFFERENT VOLUME TYPES.....	59
StatefulSet.....	62
Services	70
ClusterIP services	71
Headless Service	79
ClusterIP.....	81
NodePort.....	82
HELM.....	85

Fundamentals

Url:

[Kubernetes Tutorial for Beginners \[FULL COURSE in 4 Hours\]](#)



Things to KNOW

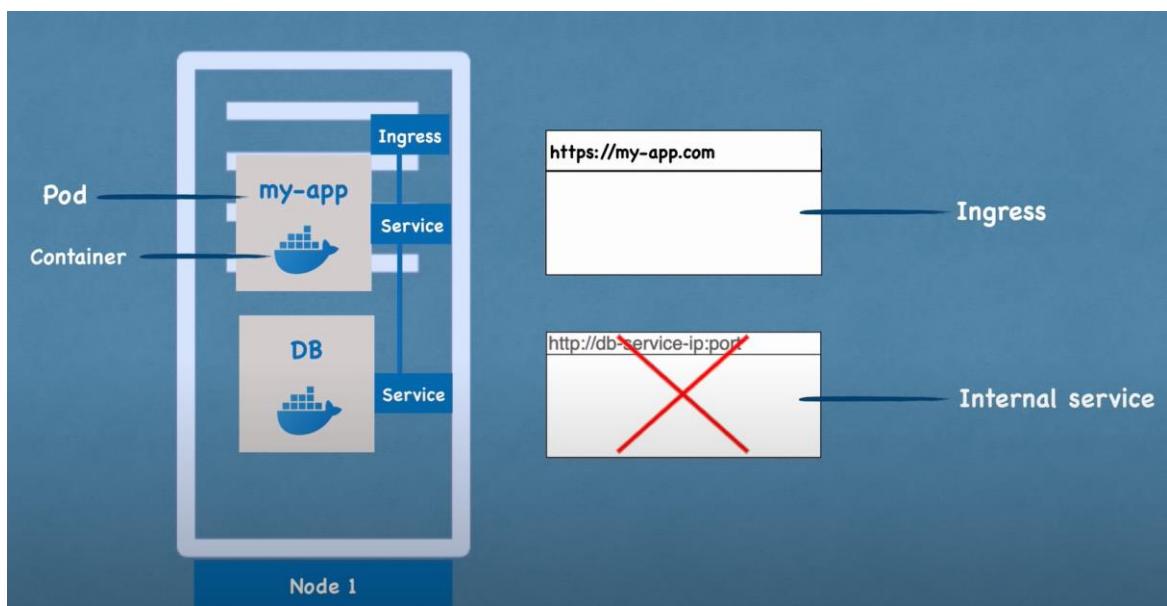
Hierarchy: Cluster -> Deployment -> ReplicaSet -> pod -> container

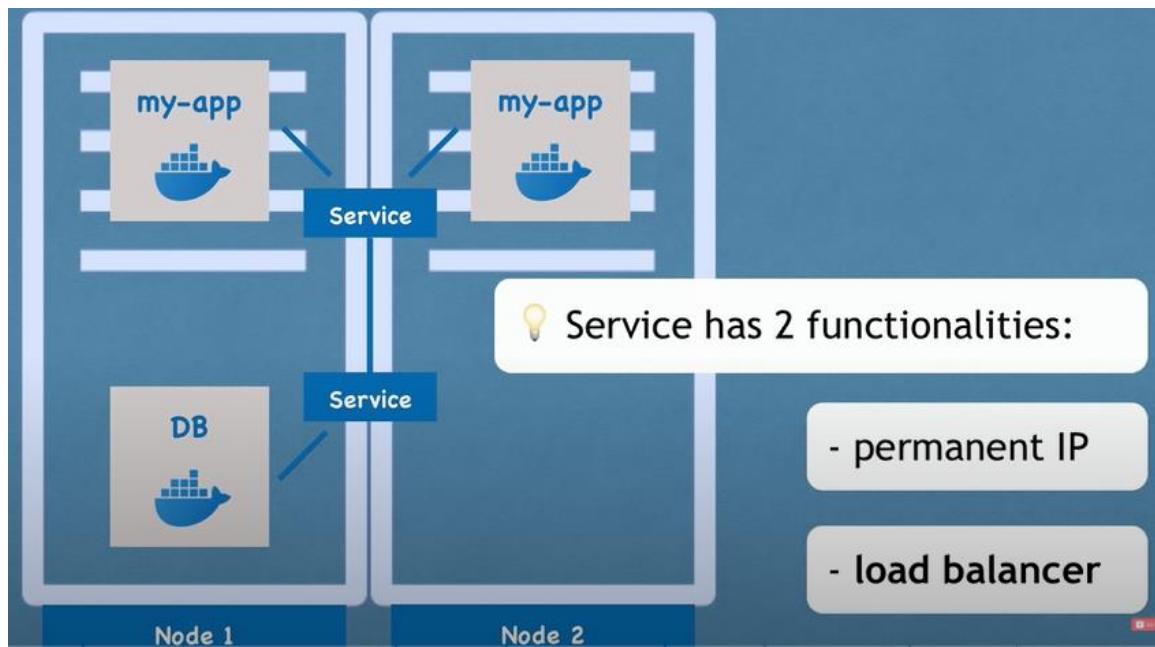
Control manager : get cluster state and push to etcd (cluster store)

Etcdb: cluster store in a key: value sets)

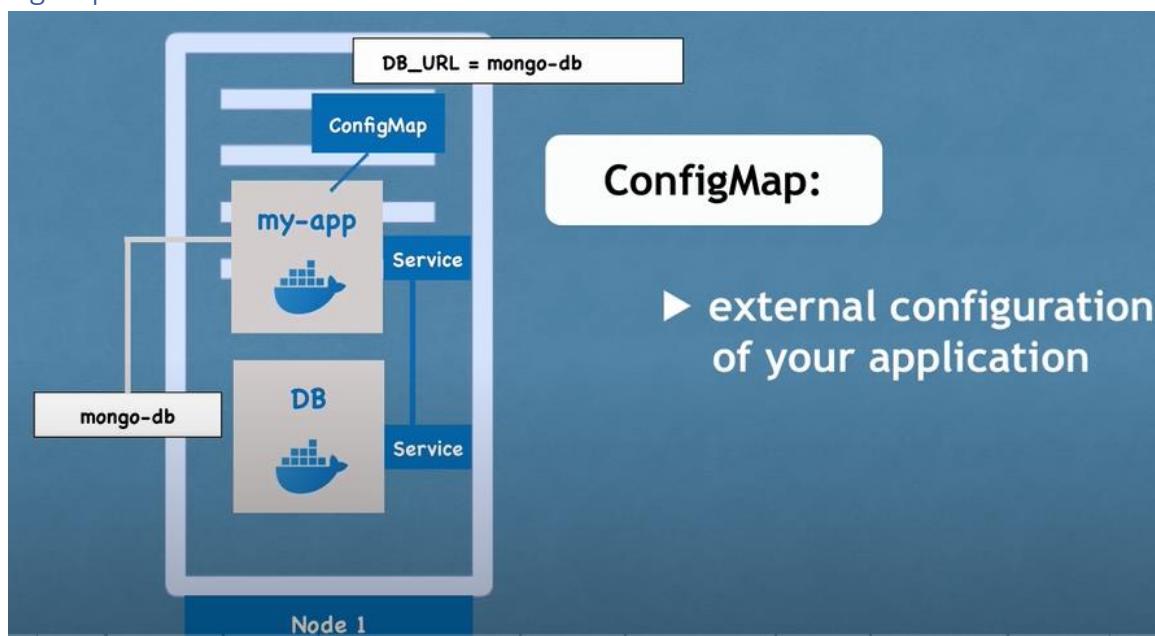
Deployment + Pod (**metadata: labels: app:**) connect to service (**spec: selector: app:**)

SERVICE and INGRESS





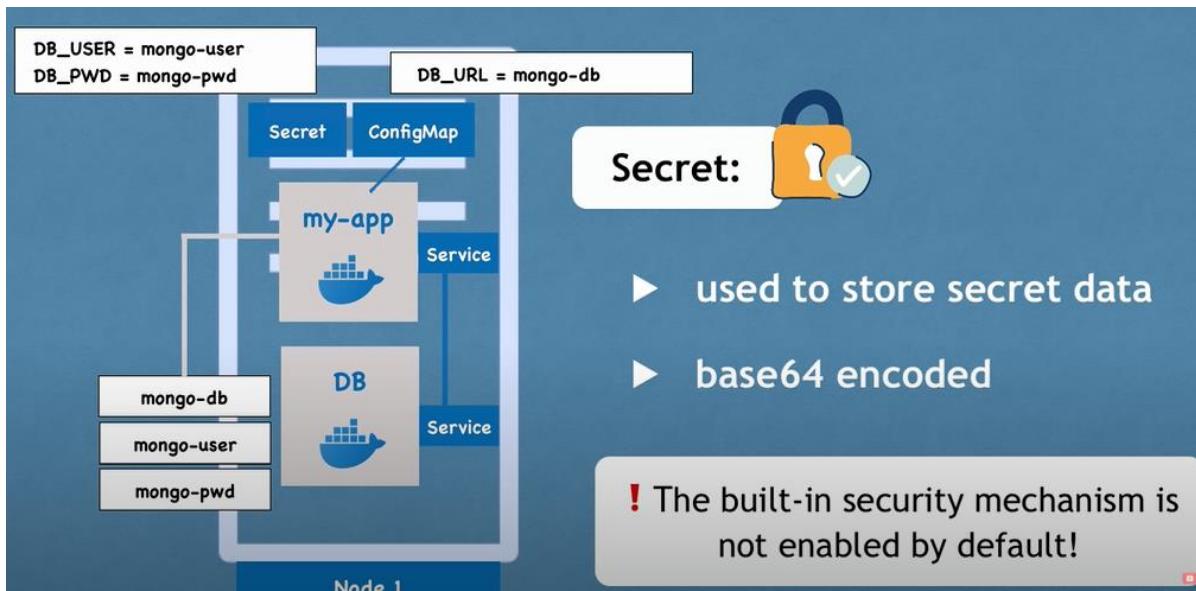
ConfigMap



DONT PUT CREDENTIAL in ConfigMap

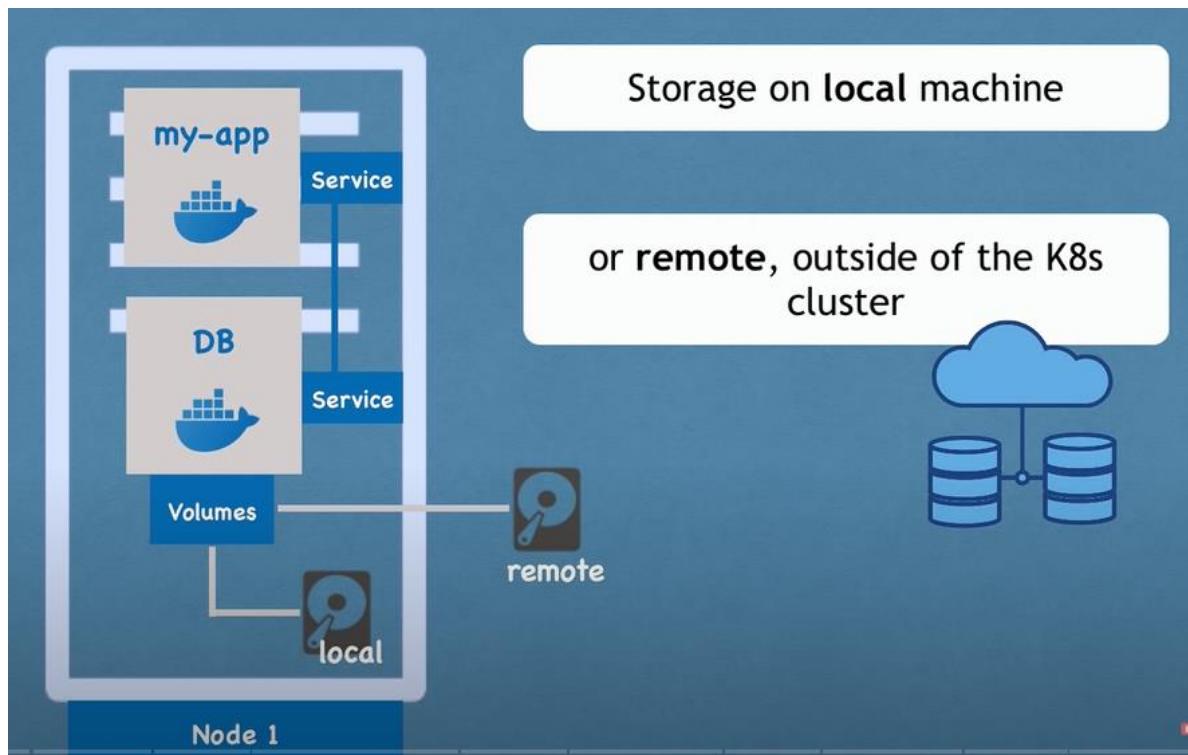
In the application we can use it as env variables or properties files

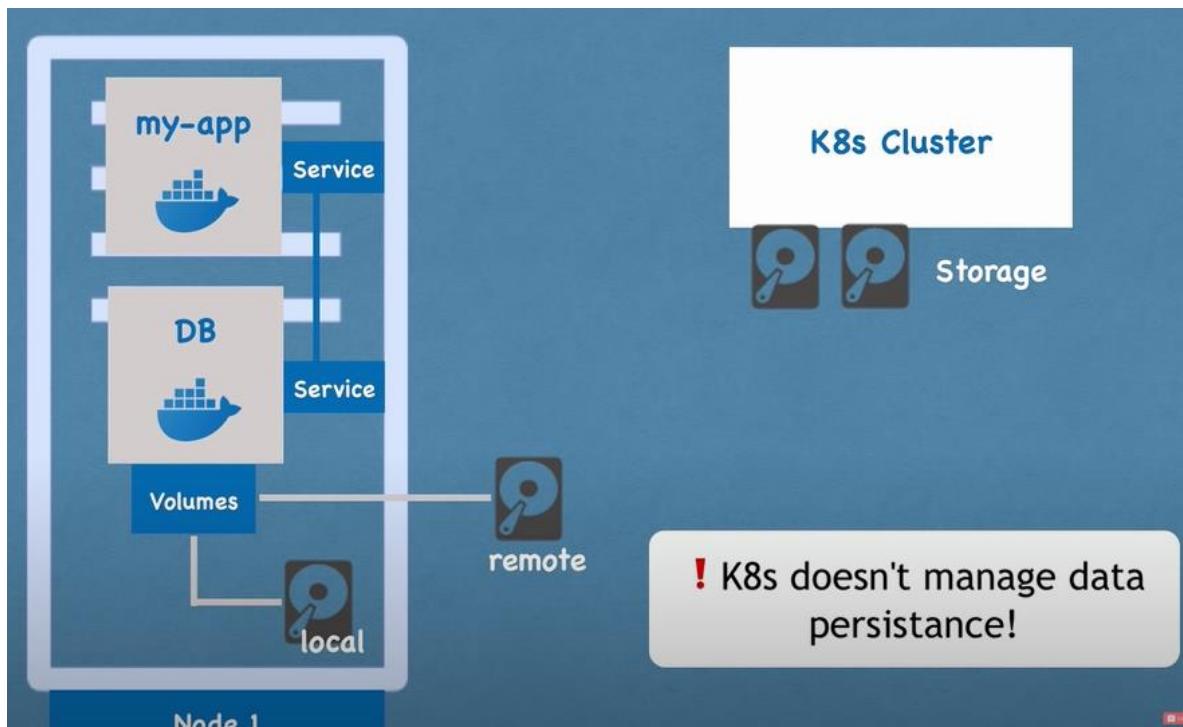
SECRETS



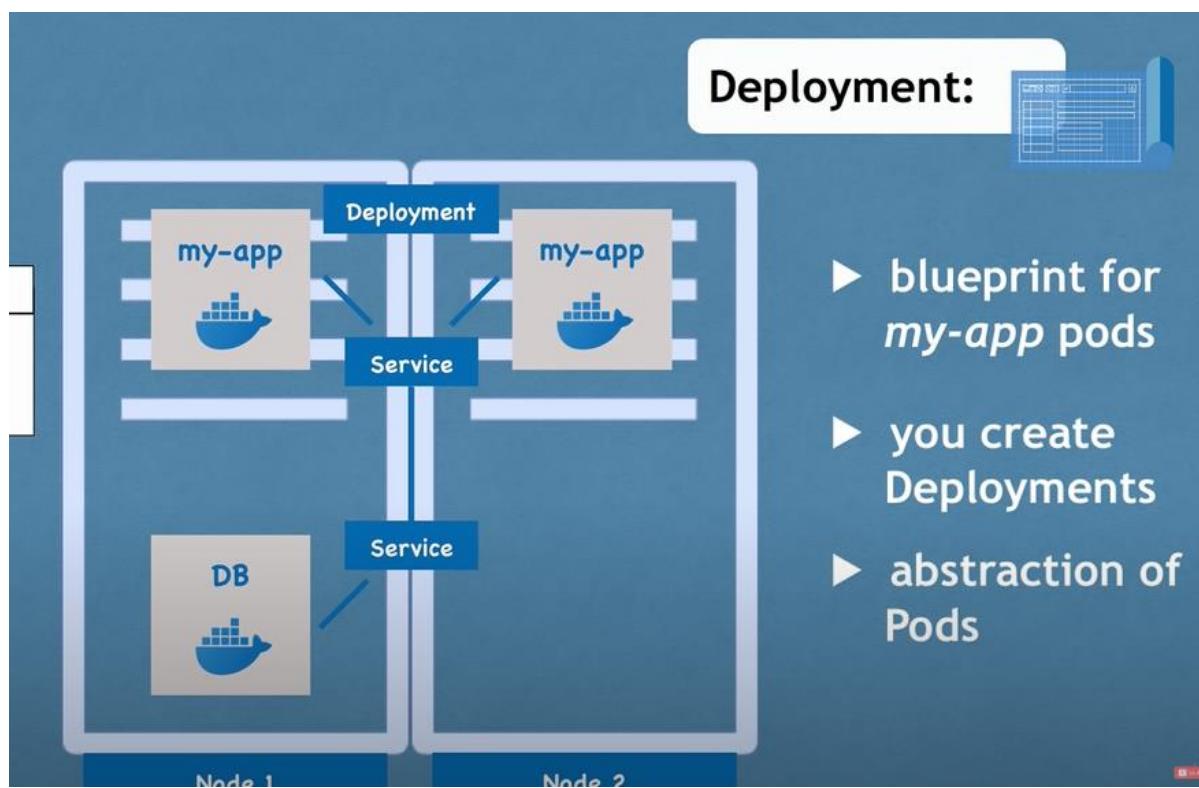
In the application we can use it as env variables or properties files

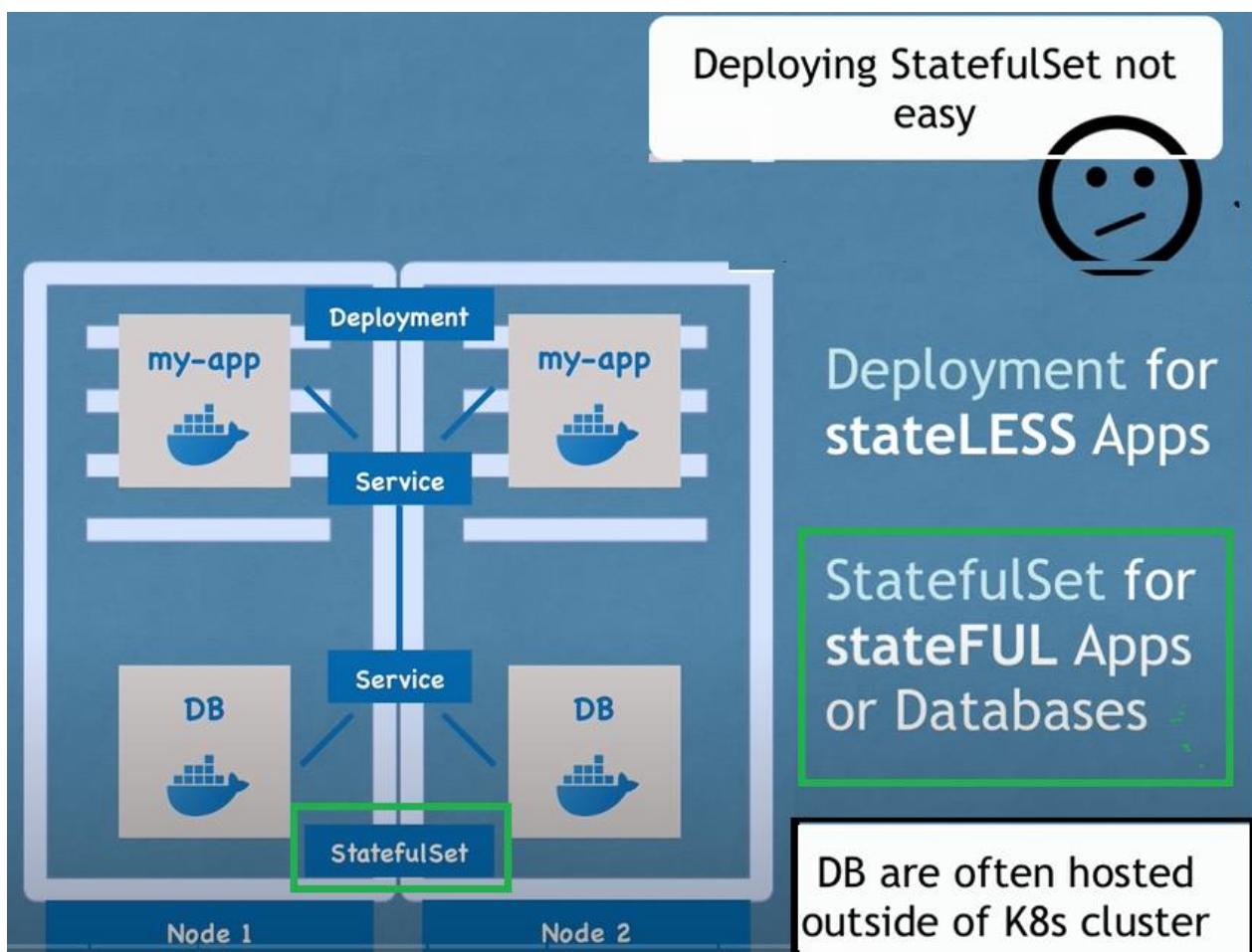
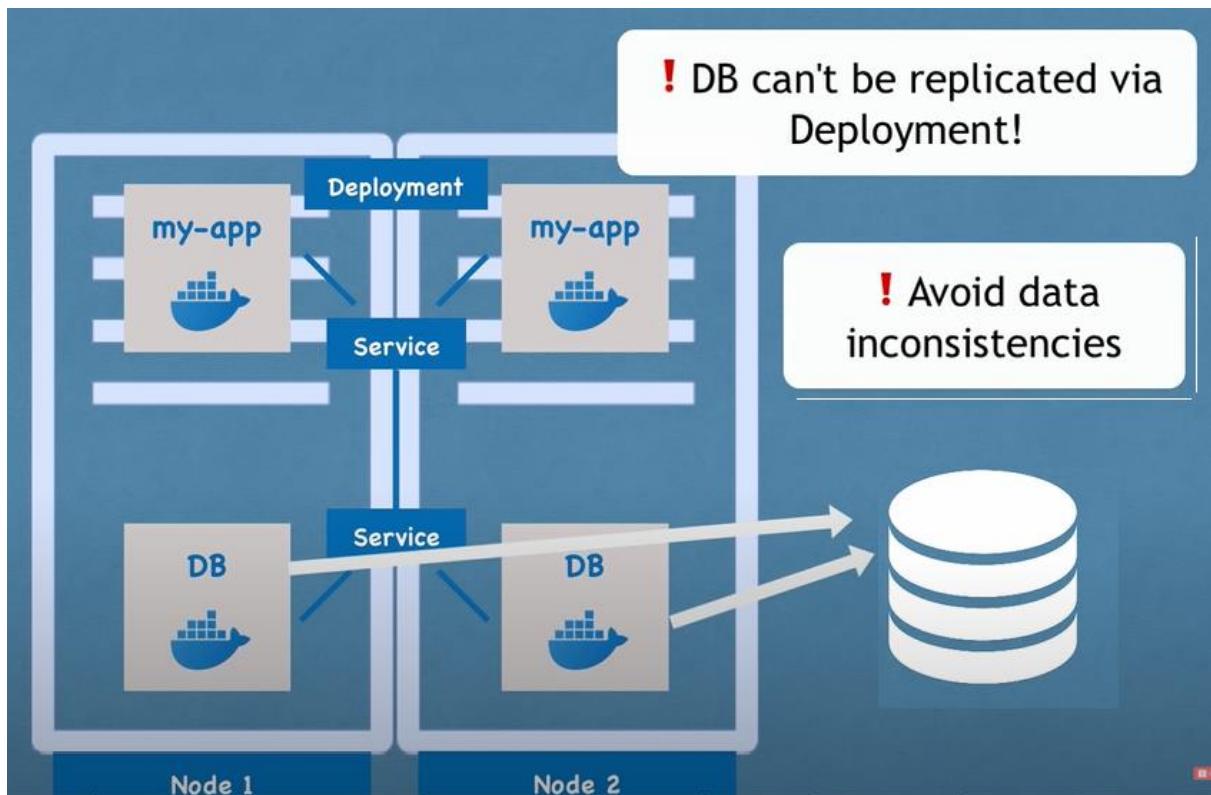
VOLUMES





Deployment and Statefull set



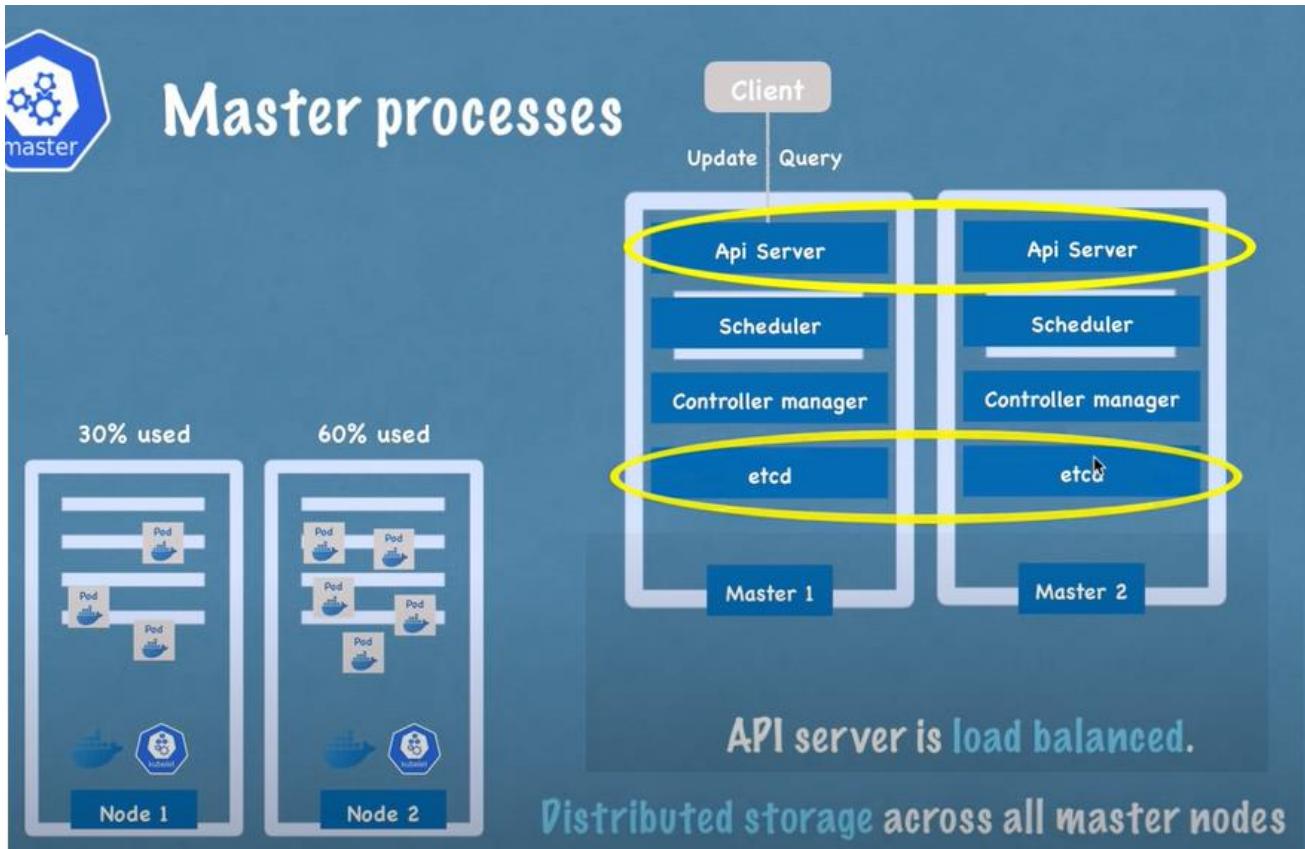




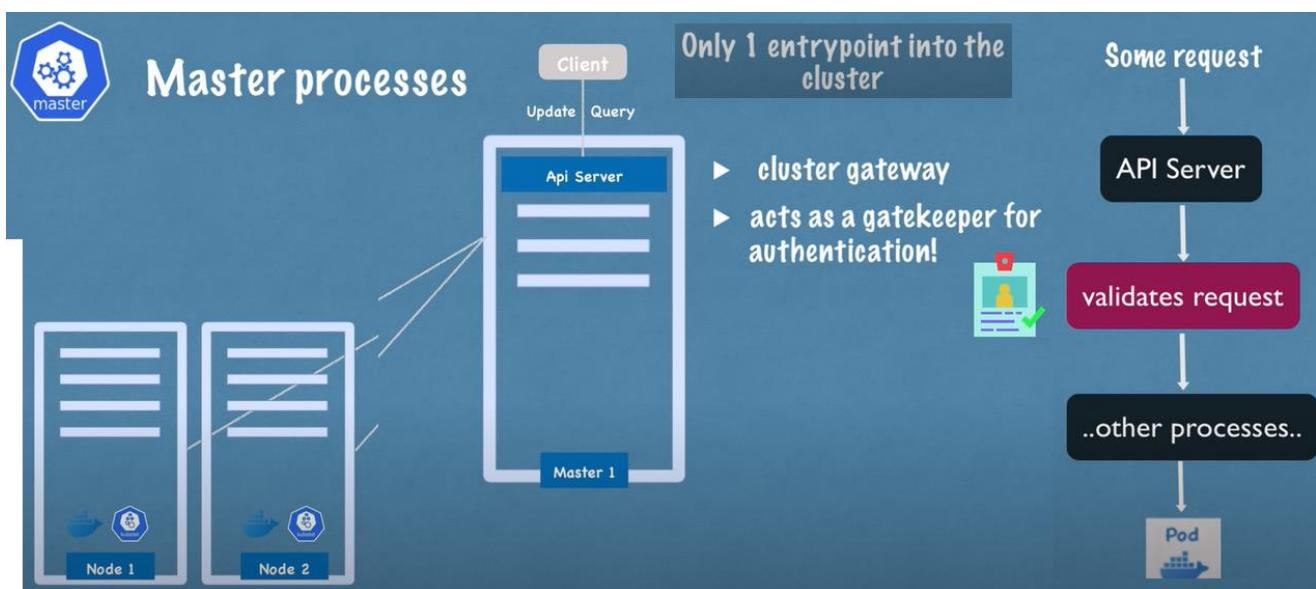
Worker machine in K8s cluster

- ▶ each Node has multiple Pods on it
- ▶ 3 processes must be installed on every Node
 1. container runtime: usually for docker
 2. kublet : interacts with node and pod
starts the pod with container insdie
 3. kublet proxy: forward the requests
- ▶ Worker Nodes do the actual work

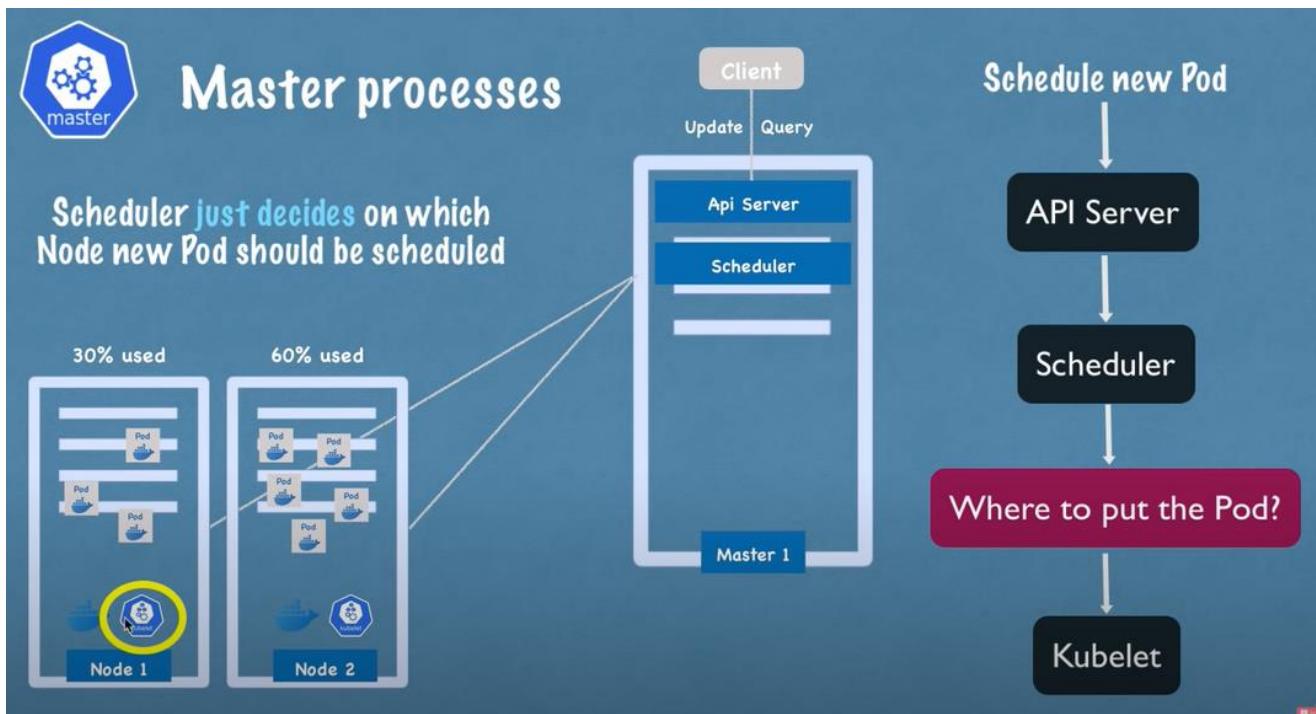
MASTER NODE



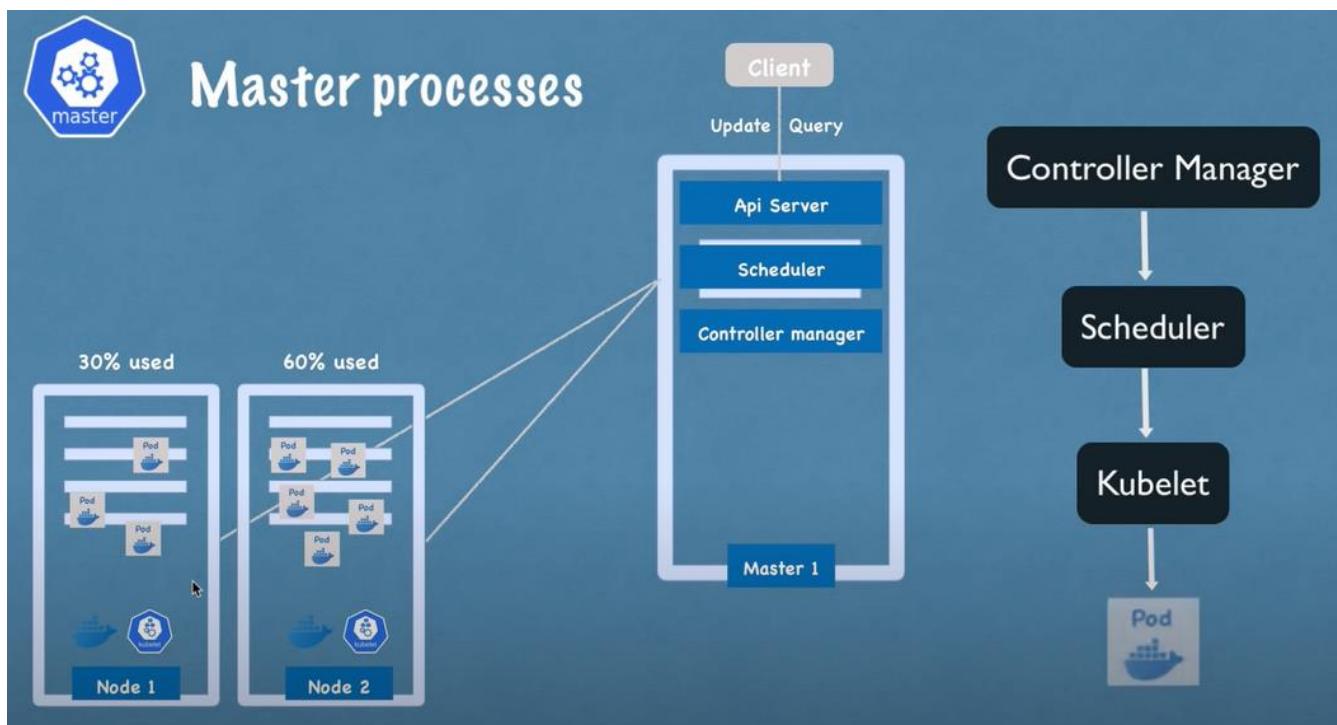
API Server



Scheduler



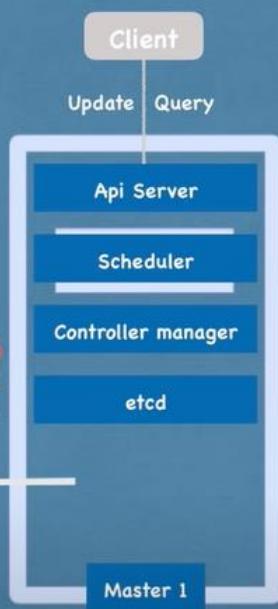
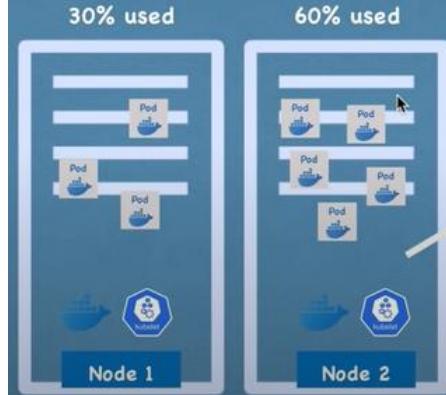
Controller Manager : detects cluster state changes



etcd : cluster store



Master processes



- ▶ Is the cluster healthy?
- ▶ What resources are available?
- ▶ Did the cluster state change?

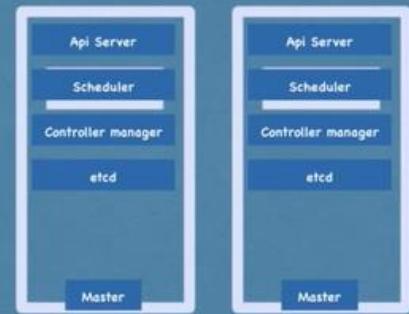
Application data is NOT stored in etcd!

CLUSTER

Example Cluster Set-Up

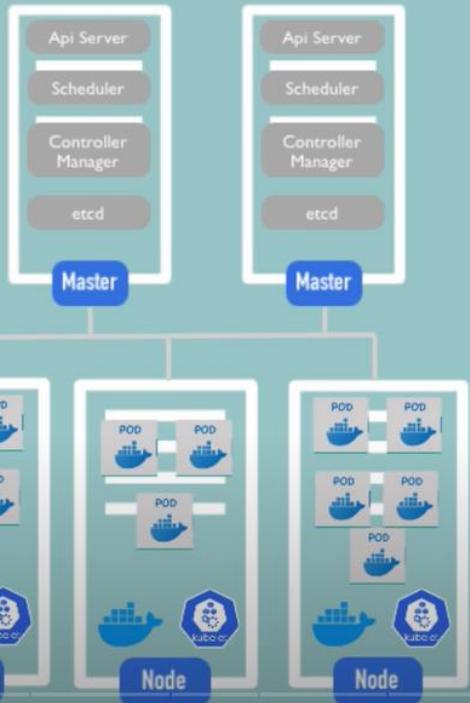
2 Master Nodes

3 Worker Nodes



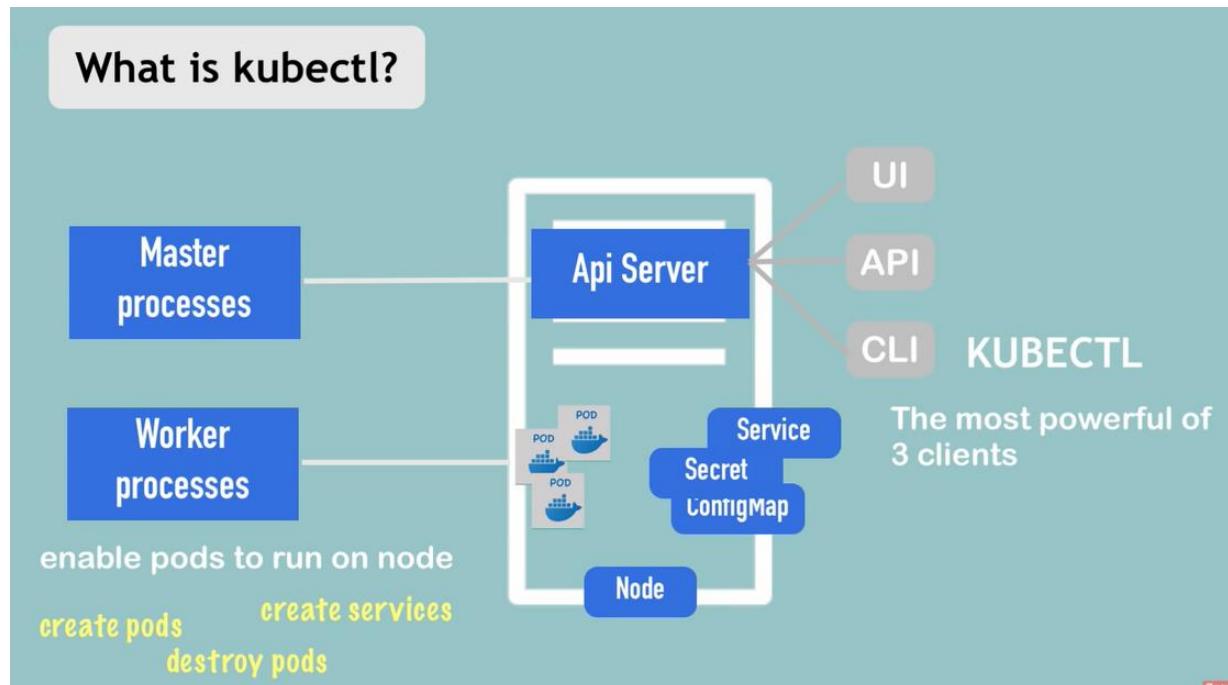
Production Cluster Setup

- Multiple Master and Worker nodes
- Separate virtual or physical machines



Minikube and kubectl local setup

API SERVER - Kubectl



Kubectl

Layers



deployment



```
deployment-name          image to use
[~]$ kubectl create deployment nginx-depl --image=nginx
deployment.apps/nginx-depl created
[~]$ kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx-depl  0/1     1           0           17s

[~]$ kubectl create deployment nginx-depl --image=nginx
deployment.apps/nginx-depl created
[~]$ kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx-depl  0/1     1           0           17s
[~]$ kubectl get pod
NAME                           READY   STATUS             RESTARTS   AGE
nginx-depl-7d9447675c-j9j8k  0/1    ContainerCreating   0          31s
```

Can be edited to update image version / etc

`kubectl edit deployment <deployment name>`

`Kubectl delete deployment <deployment name >`

Example:

```
# Edit the deployment 'mydeployment' in YAML and save the modified config in its annotation
```

```
kubectl edit deployment/mydeployment -o yaml --save-config
```

replicaset

Another abstract layer between deployment and pod:

Deployment -> replicaset -> pod :

```

pod name = deploymentname + replica hash + pod hash
[~]$ kubectl get pod
NAME                               READY   STATUS    RESTARTS   AGE
nginx-depl-7d9447675c-j9j8k      0/1     Pending   0          10s
[~]$ kubectl get pod
NAME                               READY   STATUS    RESTARTS   AGE
nginx-depl-7d9447675c-j9j8k      0/1     Pending   0          10s

deployment name           replica name = replica hash           pod hash
[~]$ kubectl get replicaset
NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-depl-7d9447675c            1         1         1            1/1        10s
[~]$ kubectl get replicaset
NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-depl-7d9447675c            1         1         1            1/1        10s

```

Debugging logs – same as in docker

Kubectl logs <pod name>

Kubectl describe <pod-name>

To solve tty Error: Unable to use a TTY - input is not a terminal or the right kind of file

Run: \$ alias kubectl='winpty kubectl'

Kubectl exec –it <pod name> -- bin/bash

Kubectl to use configuration file: apply & update

Kubectl apply –f <file name>

Example for nginx-deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.16
          ports:
            - containerPort: 80
```

Kubectl apply -f nginx-deployment.yaml:

```
[~]$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
[~]$ kubectl get pod
NAME                           READY   STATUS    RESTARTS   AGE
nginx-deployment-594cc45b78-pq5dx   1/1     Running   0          7s
[~]$ kubectl get deployment
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   1/1     1           1           52s
```

Configuration-file

contains 3 parts

```
! nginx-deployment.yaml x
```

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5  +  labels: ...
6  spec:
7    replicas: 2
8    selector: ...
9    template: ...
10   status:
11     availableReplicas: 1
12     conditions:
13       - lastTransitionTime: "2020-01-24T10:54:59Z"
14         lastUpdateTime: "2020-01-24T10:54:59Z"
15         message: Deployment has minimum availability.
16         reason: MinimumReplicasAvailable
```

Each configuration file has 3 parts

1) metadata

2) specification

3) STATE

K8s updates state
continuously!

- by etcd

Template – apply to specific pod:

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5  >  labels: ...
6  spec:
7    replicas: 2
8    selector: ...
9  >  template:
10   metadata:
11     labels:
12       app: nginx
13     spec:
14       containers:
15         - name: nginx
16           image: nginx:1.16
17           ports:
18             - containerPort: 8080
```

Template

- has its own "metadata" and "spec" section
- applies to Pod

Connect service to pods to deployment

The diagram illustrates the connection between a Deployment and a Service. On the left, a 'Deployment' configuration is shown with its metadata and spec sections highlighted. The 'labels' field under 'metadata' and the 'matchLabels' field under 'spec.selector' both contain the value 'app: nginx'. On the right, a 'Service' configuration is shown with its spec section highlighted, which includes a 'selector' field with the same 'app: nginx' value. This visualizes how the Deployment's label requirement is matched by the Service's selector.

```
Deployment
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: nginx-deployment
5   labels:
6     app: nginx
7 spec:
8   replicas: 2
9   selector:
10    matchLabels:
11      app: nginx
12 template:
13   metadata:
14     labels:
15       app: nginx
16 > app: nginx
```

```
Labels & Selectors
Service
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: nginx-service
5 spec:
6   selector:
7     app: nginx
8 > ports: ...
```

The diagram illustrates the connection between a Deployment and a Pod. On the left, a 'Deployment' configuration is shown with its spec section highlighted. The 'replicas' field is set to 2, and the 'selector' field contains a 'matchLabels' field with the value 'app: nginx'. On the right, a 'Connecting Deployment to Pods' section contains a 'Pod' configuration with its spec section highlighted. The 'selector' field in the pod's spec also contains a 'matchLabels' field with the value 'app: nginx'. This visualizes how the Deployment's label requirement is matched by the Pod's selector.

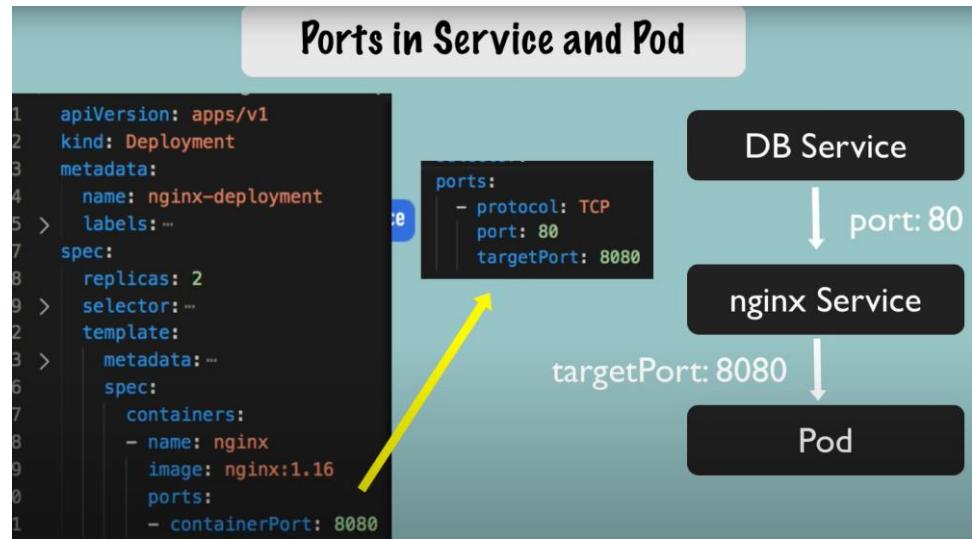
```
Deployment
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: nginx-deployment
5   labels:
6     app: nginx
7 spec:
8   replicas: 2
9   selector:
10    matchLabels:
11      app: nginx
12 template:
13   metadata:
14     labels:
15       app: nginx
16 > app: nginx
```

```
Connecting Deployment to Pods
- Pods get the label through the template blueprint
- This label is matched by the selector
  selector:
    matchLabels:
      app: nginx
```

The diagram illustrates the connection between a Deployment and a Service, with arrows indicating the flow of labels. On the left, a 'Deployment' configuration is shown with its spec section highlighted. The 'replicas' field is set to 2, and the 'selector' field contains a 'matchLabels' field with the value 'app: nginx'. On the right, a 'Service' configuration is shown with its spec section highlighted, which includes a 'selector' field with the same 'app: nginx' value. Two yellow arrows point from the 'Deployment.spec.selector.matchLabels.app' field to the 'Service.spec.selector.matchLabels.app' field, visually representing the label matching process.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
```

```
Service
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: nginx-service
5 spec:
6   selector:
7     app: nginx
8 > ports: ...
```



! nginx-service.yaml ×

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: nginx-service
5  spec:
6    selector:
7      app: nginx
8    ports:
9      - protocol: TCP
10     port: 80
11     targetPort: 8080
12

```

Service listen on port 80 and forward to pod port 8080:

Service listen on port 80:

```
[Documents]$ kubectl apply -f nginx-service.yaml
service/nginx-service created
[Documents]$ kubectl get pod
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-7d64f4b574-fklxj  1/1    Running   0          10s
nginx-deployment-7d64f4b574-v7mwj  1/1    Running   0          10s
[Documents]$ kubectl get service
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP     PORT(S)      AGE
kubernetes      ClusterIP   10.96.0.1      <none>        443/TCP     2d
nginx-service   ClusterIP   10.96.25.229   <none>        80/TCP      19s
[Documents]$
```

forward to pod port 8080:

Kubectl describe service nginx-service:

```
kubernetes      ClusterIP   10.96.0.1      1   apiVersion: v1
nginx-service   ClusterIP   10.96.25.229   2   kind: Service
[Documents]$ kubectl describe service nginx-service
Name:           nginx-service
Namespace:      default
Labels:          <none>
Annotations:    kubectl.kubernetes.io/cluster-ip: {"apiVersion": "v1", "clusterIP": "10.96.25.229", "name": "nginx-service", "namespace": "default", "selector": "app=nginx", "type": "ClusterIP"}, kubectl.kubernetes.io/service-ip: "10.96.25.229"
Selector:        app=nginx
Type:           ClusterIP
IP:             10.96.25.229
Port:           <unset>  80/TCP
TargetPort:     8080/TCP
Endpoints:      172.17.0.6:8080,172.17.0.7:8080
[Documents]$
```

How do we know that these are the correct ip addresses ? Using get pod -o wide:

```
Port:           <unset>  80/TCP
TargetPort:    8080/TCP
Endpoints:     172.17.0.6:8080,172.17.0.7:8080
Session Affinity: None
Events:        <none>
[Documents]$ kubectl get pod -o wide
NAME           NOMINATED NODE   READINESS GATES   READY   STATUS    RESTARTS   AGE   IP
nginx-deployment-7d64f4b574-fklxj  ikube       <none>      1/1    Running   0          2m17s  172.17.0.7
nginx-deployment-7d64f4b574-v7mwj  ikube       <none>      1/1    Running   0          2m17s  172.17.0.6
[Documents]$
```

Get deployment status (from etcd): -o yaml

Get deployment nginx-deployment -o yaml > nginx-deployment=result.yaml

```

status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: "2020-01-24T10:54:59Z"
    lastUpdateTime: "2020-01-24T10:54:59Z"
    message: Deployment has minimum availability.
    reason: MinimumReplicasAvailable
    status: "True"
    type: Available
  - lastTransitionTime: "2020-01-24T10:54:56Z"
    lastUpdateTime: "2020-01-24T10:54:59Z"
    message: ReplicaSet "nginx-deployment-7d64f4b574" has successfully progressed.
    reason: NewReplicaSetAvailable
    status: "True"
    type: Progressing
  observedGeneration: 1
  readyReplicas: 2
  replicas: 2
  updatedReplicas: 2
[Documents]$ kubectl get deployment nginx-deployment -o yaml > nginx-deployment-result.yaml

```

```

! nginx-deployment.yaml ✘ ...
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4  |   name: nginx-deployment
5  |   labels:
6  |     app: nginx
7  spec:
8  |   replicas: 2
9  |   selector:
10 |     matchLabels:
11 |       app: nginx
12 |     template:
13 |       metadata:
14 |         labels:
15 |           app: nginx
16 |         spec:
17 |           containers:
18 |             - name: nginx
19 |               image: nginx:1.16
20 |             ports:
21 |               - containerPort: 8080
22
! nginx-deployment-result.yaml ✘ ...
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata: ...
17  spec: ...
50  status:
51  |   availableReplicas: 2
52  |   conditions:
53  |   - lastTransitionTime: "2020-01-24T10:54:59Z"
54  |     lastUpdateTime: "2020-01-24T10:54:59Z"
55  |     message: Deployment has minimum availability.
56  |     reason: MinimumReplicasAvailable
57  |     status: "True"
58  |     type: Available
59  - lastTransitionTime: "2020-01-24T10:54:56Z"
60  |     lastUpdateTime: "2020-01-24T10:54:59Z"
61  |     message: ReplicaSet "nginx-deployment-7d64f4b574" has successfully progressed.
62  |     reason: NewReplicaSetAvailable
63  |     status: "True"
64  |     type: Progressing
65  |   observedGeneration: 1
66  |   readyReplicas: 2
67  |   replicas: 2

```

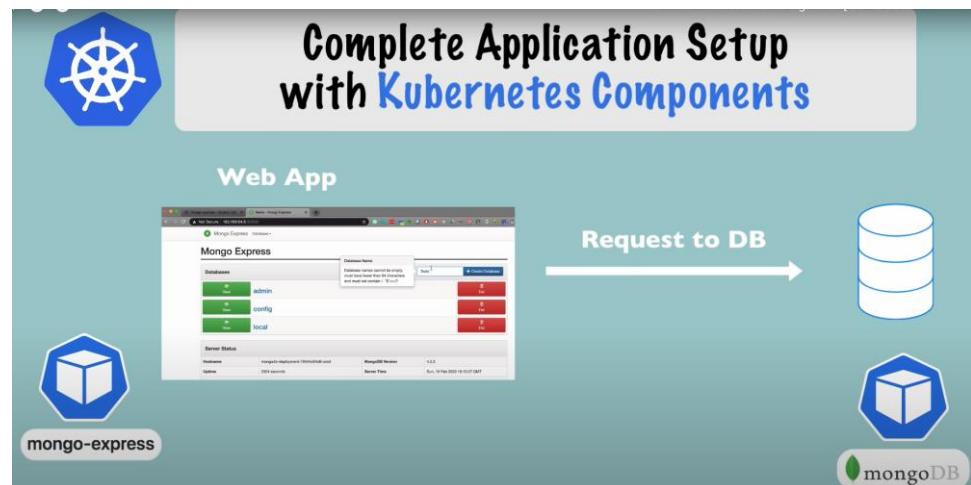
Delete and edit configuration files:

```

[Documents]$ kubectl get deployment nginx-deployment -o yaml > nginx-deployment.yaml
[Documents]$ kubectl delete -f nginx-deployment.yaml
deployment.apps "nginx-deployment" deleted
[Documents]$ kubectl delete service mongo

```

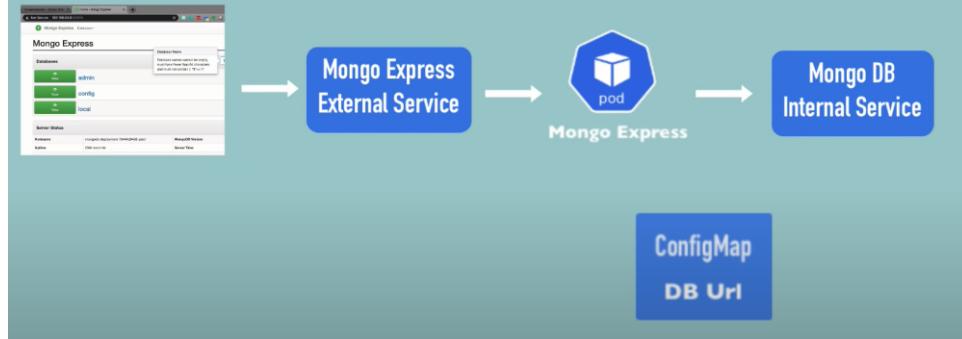
Demo on mongod



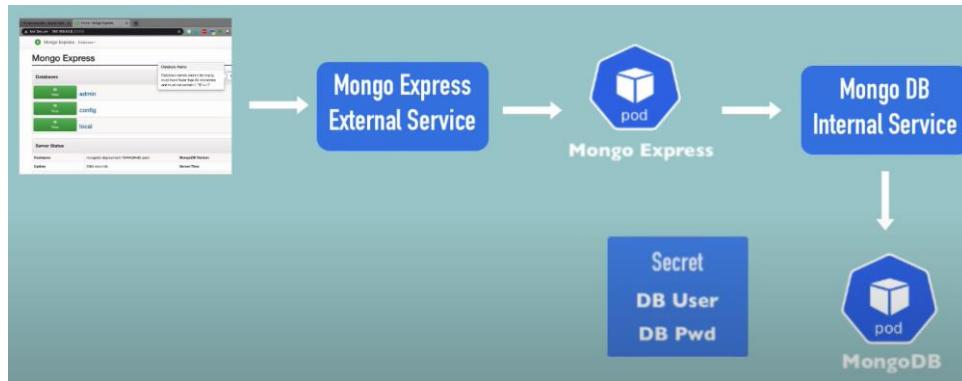
Architecture



Browser Request Flow through the K8s components



Authentication:



mongo.yaml deployment file;

Good reference: <https://devopscube.com/deploy-mongodb-kubernetes/>

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
  labels:
    app: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - name: mongodb
          image: mongo

```

Pod Blueprint:

- name: mongodb
- image: mongo

Getting ports and env variables from docker hub mongo image and adding to template pod:

```

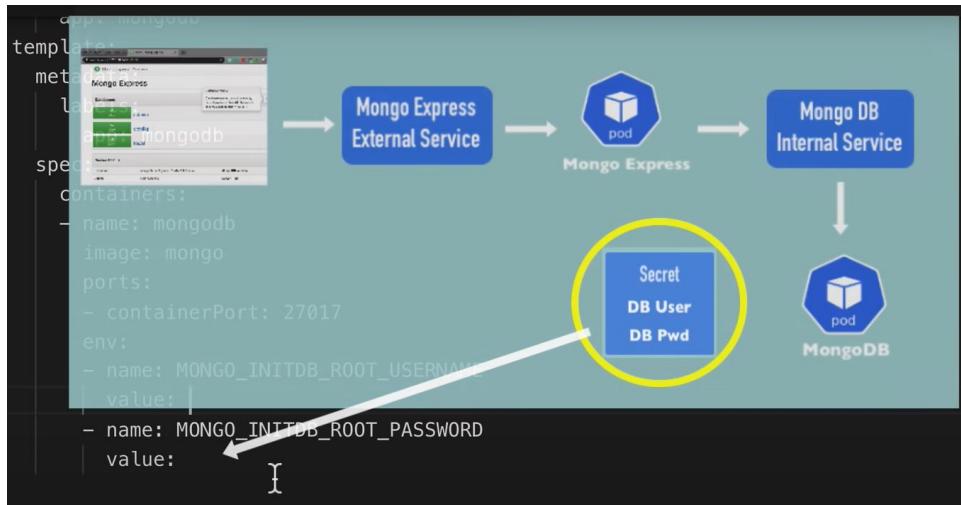
template:
  metadata:
    labels:
      app: mongodb
  spec:
    containers:
      - name: mongodb
        image: mongo
        ports:
          - containerPort: 27017
        env:
          - name: MONGO_INITDB_ROOT_USERNAME
            value: |
          - name: MONGO_INITDB_ROOT_PASSWORD
            value:

```

Deployment Config File is checked into repository.

Username and Password should not go here!

Creating secrets:



mongo-secret.yaml deployment file:

The screenshot shows the mongo-secret.yaml file on the left and a diagram titled "Secret Configuration File" on the right.

mongo-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
|   name: mongodb-secret
type: Opaque
data:
|   username: I
|   password: I
```

Secret Configuration File

- kind: "Secret"
- metadata / name: a random name
- type: "Opaque" - default for arbitrary key-value pairs
- data: the actual contents - in key-value pairs

Adding value as base64 :

The terminal output shows the base64 encoding of 'username' and 'password'.

```
[~]$ echo -n 'username' | base64
dXNlcm5hbWU=
[~]$ echo -n 'password' | base64
cGFzc3dvcmQ=
```

mongo-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
|   name: mongodb-secret
type: Opaque
data:
|   mongo-root-username: I
|   mongo-root-password: I
```

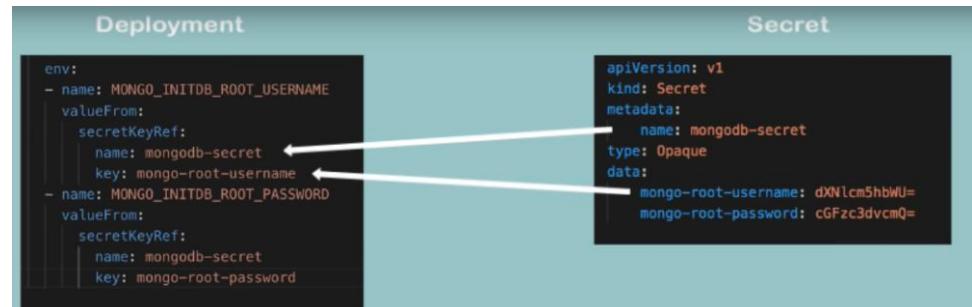
Warning: Storing the data in a Secret component doesn't automatically make it secure.

There are built-in mechanism (like encryption) for basic security, which are not enabled by default.

```
! mongo.yaml      ! mongo-secret.yaml ×
1  apiVersion: v1
2  kind: Secret
3  metadata:
4    name: mongodb-secret
5  type: Opaque
6  data:
7    mongo-root-username: dXNlcj5hbWU=
8    mongo-root-password: cGFzc3dvcmQ=
9
```

```
[~]$ cd k8s-configuration/
[k8s-configuration]$ ls
mongo-secret.yaml      mongo.yaml
[k8s-configuration]$ kubectl apply -f mongo-secret.yaml
secret/mongodb-secret created
[k8s-configuration]$ kubectl get secret
NAME              TYPE          DATA   AGE
default-token-4clzs  kubernetes.io/service-account-token  3       22m
mongodb-secret     Opaque        2       68s
[k8s-configuration]$
```

Secret can be referenced now in Deployment



```

! mongo.yaml ! mongo-secret.yaml

12   template:
13     metadata:
14       labels:
15         app: mongodb
16     spec:
17       containers:
18         - name: mongodb
19           image: mongo
20           ports:
21             - containerPort: 27017
22           env:
23             - name: MONGO_INITDB_ROOT_USERNAME
24               valueFrom:
25                 secretKeyRef:
26                   name: mongodb-secret
27                   key: mongo-root-username
28             - name: MONGO_INITDB_ROOT_PASSWORD
29               valueFrom:
30                 secretKeyRef:
31                   name: mongodb-secret
32                   key: mongo-root-password

```

Mongo db Deployment +

Get deployment, replicaset and pod:

```

[k8s-configuration]$ kubectl get secret
NAME          TYPE      DATA  AGE
default-token-4clzs  kubernetes.io/service-account-token  3    22m
mongodb-secret  Opaque      2    68s
[k8s-configuration]$ kubectl apply -f mongo.yaml
deployment.apps/mongodb-deployment created
[k8s-configuration]$ kubectl get all
NAME                           READY   STATUS            RESTARTS
AGE
pod/mongodb-deployment-78444d94d6-zsrc1  0/1    ContainerCreating   0
7s

NAME          TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/kubernetes  ClusterIP  10.96.0.1    <none>        443/TCP    25m

NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mongodb-deployment  0/1     1           0          7s

NAME          DESIRED  CURRENT  READY   AGE
replicaset.apps/mongodb-deployment-78444d94d6  1        1       0      7s

```

Mongo db internal service inside same deployment file:



mongo.yaml ● mongo-secret.yaml Kubernetes Tutorial for Beginners [FULL COURSE]

```
19     image: mongo
20     ports:
21       - containerPort: 27017
22     env:
23       - name: MONGO_INITDB_ROOT_USERNAME
24         valueFrom:
25           secretKeyRef:
26             name: mongodb-secret
27             key: mongo-root-username
28       - name: MONGO_INITDB_ROOT_PASSWORD
29         valueFrom:
30           secretKeyRef:
31             name: mongodb-secret
32             key: mongo-root-password
33   --- Deployment separation
34
```

Deployment & Service in 1 file, because they belong together.

mongo.yaml ● mongo-secret.yaml Kubernetes Tutorial for Beginners [FULL COURSE]

```
27     key: mongo-root-username
28   - name: MONGO_INITDB_ROOT_USERNAME
29     valueFrom:
30       secretKeyRef:
31         name: mongodb-secret
32         key: mongo-root-username
33   ---
34   apiVersion: v1
35   kind: Service
36   metadata:
37     name: mongodb-service
38   spec:
39     selector:
40       app: mongodb
41     ports:
42       - protocol: TCP
43         port: 27017
44         targetPort: 27017
45 }
```

Service Configuration File

- kind: "Service"
- metadata / name: a random name
- selector: to connect to Pod through label
- ports:
 - port: Service port
 - targetPort: containerPort of Deployment

```
[k8s-configuration]$ kubectl apply -f mongo.yaml
deployment.apps/mongodb-deployment unchanged
service/mongodb-service created
```

Service is listening on 27017 (mongo port):

```
[k8s-configuration]$ kubectl get service
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)    AGE
kubernetes     ClusterIP 10.96.0.1    <none>       443/TCP   36m
mongodb-service ClusterIP 10.96.86.105 <none>       27017/TCP  51s
```

Make sure service is connecting to the pods (using describe on the service and get pod -o wide):

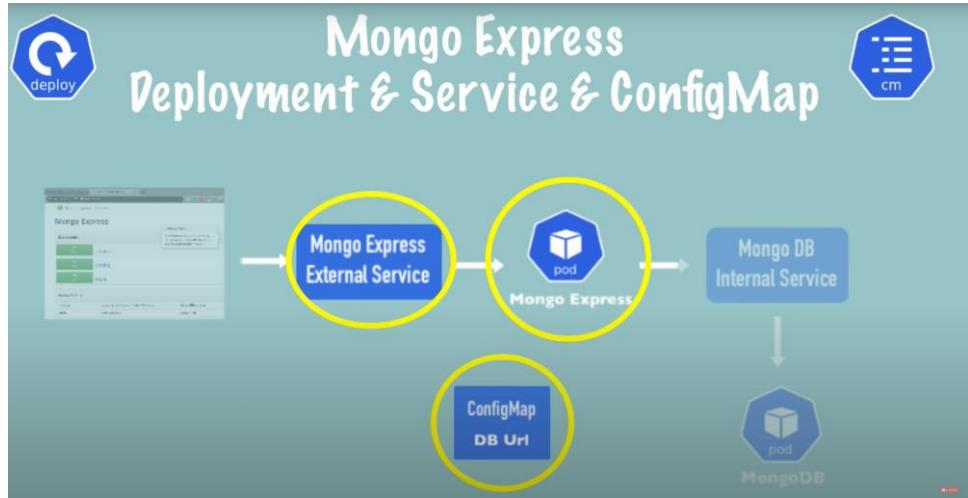
```
[k8s-configuration]$ kubectl get service
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)    AGE
kubernetes     ClusterIP 10.96.0.1    <none>       443/TCP   36m
mongodb-service ClusterIP 10.96.86.105 <none>       27017/TCP  51s
[k8s-configuration]$ kubectl describe service mongodb-service
Name:         mongodb-service
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
              {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"mongodb-service","namespace":"default"},"spec":{"ports":[{"port":...
Selector:    app=mongodb
Type:        ClusterIP
IP:          10.96.86.105
Port:        <unset>  27017/TCP
TargetPort:  27017/TCP
Endpoints:   172.17.0.6:27017
Session Affinity: None
Events:      <none>

Selector:    app=mongodb
Type:        ClusterIP
IP:          10.96.86.105
Port:        <unset>  27017/TCP
TargetPort:  27017/TCP
Endpoints:   172.17.0.6:27017
Session Affinity: None
Events:      <none>
[k8s-configuration]$ kubectl get pod -o wide
NAME                           READY   STATUS    RESTARTS   AGE   IP
mongodb-deployment-78444d94d6-zsrc1  1/1    Running   0        12m   172.17
.0.6   minikube   <none>   <none>
```

Summarize the mongo service / pod deployment / replicaset:

```
[k8s-configuration]$ kubectl get all | grep mongo
pod/mongodb-deployment-78444d94d6-zsrc1  1/1    Running   0        13m
service/mongodb-service   ClusterIP  10.96.86.105 <none>       27017/TCP
2m52s
deployment.apps/mongodb-deployment  1/1    1        1        13m
replicaset.apps/mongodb-deployment-78444d94d6  1        1        1        13m
```

mongo express deployment , service and configMap:



Environment vars for mongo-express (from docker-hub):

Name	Default
ME_CONFIG_BASICAUTH_USERNAME	''
ME_CONFIG_BASICAUTH_PASSWORD	''
ME_CONFIG_MONGODB_ENABLE_ADMIN	'true'
ME_CONFIG_MONGODB_ADMINUSERNAME	''
ME_CONFIG_MONGODB_ADMINPASSWORD	''
ME_CONFIG_MONGODB_PORT	27017
ME_CONFIG_MONGODB_SERVER	'mongo'
ME_CONFIG_OPTIONS_EDITORTHEME	'default'
ME_CONFIG_REQUEST_SIZE	'100kb'
ME_CONFIG_SITE_BASEURL	'/'
ME_CONFIG_SITE_COOKIESECRET	'cookiesecr...
ME_CONFIG_SITE_SESSIONSECRET	'sessionsecr...
ME_CONFIG_SITE_SSL_ENABLED	'false'
ME_CONFIG_SITE_SSL_CRT_PATH	''
ME_CONFIG_SITE_SSL_KEY_PATH	''

Which database to connect?

MongoDB Address / Internal Service
...MONGODB_SERVER

Which credentials to authenticate?

...ADMINUSERNAME
...ADMINPASSWORD

Port: 8081

Mongo-express.yaml config file:

Secrets should be same secrets as we use in mongo.yaml file.

ConfigMap is being shared between several applications:

Kubernetes Tutorial for Beginners [FULL COURSE in 4 Hours]

ConfigMap

- external configuration
- centralized
- other components can use it



```
16 spec:  
17   containers:  
18     - name: mongo-express  
19       image: mongo-express  
20       ports:  
21         - containerPort: 8081  
22     env:  
23       - name: ME_CONFIG_MONGO  
24         valueFrom:  
25           secretKeyRef:  
26             name: mongodb-secret  
27             key: mongo-root-u  
28       - name: ME_CONFIG_MONGO  
29         valueFrom:  
30           secretKeyRef:  
31             name: mongodb-secret  
32             key: mongo-root-p  
33       - name: ME_CONFIG_MONGO  
34         value: 1
```

mongo.yaml

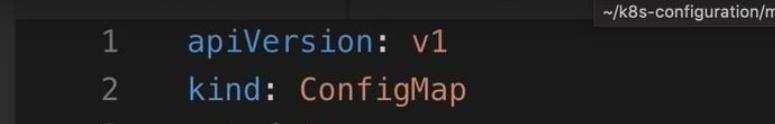
```
1 apiVersion: apps/v1  
2 kind: Deployment  
3 metadata:  
4   name: mongo-express  
5   labels:  
6     app: mongo-express  
7 spec:  
8   replicas: 1  
9   selector:  
10    matchLabels:  
11      app: mongo-express  
12   template:  
13     metadata:  
14       labels:  
15         app: mongo-express
```

```
15 |     app: mongo-express
16 |   spec:
17 |     containers:
18 |       - name: mongo-express
19 |         image: mongo-express
20 |         ports:
21 |           - containerPort: 8081
22 |         env:
23 |           - name: ME_CONFIG_MONGODB_ADMINUSERNAME
24 |             valueFrom:
25 |               secretKeyRef:
26 |                 name: mongodb-secret
27 |                 key: mongo-root-username
28 |           - name: ME_CONFIG_MONGODB_ADMINPASSWORD
29 |             valueFrom:
30 |               secretKeyRef:
31 |                 name: mongodb-secret
32 |                 key: mongo-root-password
33 |           - name: ME_CONFIG_MONGODB_SERVER
34 |             value:
```

Mongo-configmap.yml – does not have a type as we have just one config map:

ConfigMap Configuration File
<pre>apiVersion: v1 kind: ConfigMap metadata: name: mongodb-configmap data: database_url:</pre>
<ul style="list-style-type: none">- kind: "ConfigMap"
<ul style="list-style-type: none">- metadata / name: a random name
<ul style="list-style-type: none">- data: the actual contents - in key-value pairs

Database-url is the name of the service !!



The screenshot shows a code editor with two tabs open. The left tab is titled 'mongo.yaml' and contains the following YAML configuration:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mongodb-configmap
data:
  database_url: mongodb-service
```

The right tab is titled 'mongo-express.yaml' and contains the following YAML configuration:

```
! mongo-express.yaml
```

Both tabs have a status bar at the bottom indicating the path: ~k8s-configuration/mongo-express.

mongo-express.yaml with configmap:

```
 mongo.yaml      ! mongo-express.yaml ×      ! mongo-confi
17   containers:
18     - name: mongo-express
19       image: mongo-express
20       ports:
21         - containerPort: 8081
22       env:
23         - name: ME_CONFIG_MONGODB_ADMINUSERNAME
24           valueFrom:
25             secretKeyRef:
26               name: mongodb-secret
27               key: mongo-root-username
28         - name: ME_CONFIG_MONGODB_ADMINPASSWORD
29           valueFrom:
30             secretKeyRef:
31               name: mongodb-secret
32               key: mongo-root-password
33         - name: ME_CONFIG_MONGODB_SERVER
34           valueFrom:
35             configMapKeyRef:
36               name: mongodb-configmap
37               key: database_url
```

Create config-map first, and mongo-express

```
[k8s-configuration]$ kubectl apply -f mongo-configmap.yaml
configmap/mongodb-configmap created
[k8s-configuration]$ kubectl apply -f mongo-express.yaml
deployment.apps/mongo-express created
[k8s-configuration]$ kubectl get pod
NAME                           READY   STATUS        RESTARTS   AGE
mongo-express-797845bd97-p9grri  0/1    ContainerCreating  0          5s
mongodb-deployment-78444d94d6-zsrc1 1/1    Running       0          23m
[k8s-configuration]$
```

Make sure mongo-express is up and running":

```
[k8s-configuration]$ kubectl get pod
NAME                               READY   STATUS    RESTARTS   AGE
mongo-express-797845bd97-p9grr     1/1    Running   0          7m30s
mongodb-deployment-78444d94d6-zsrc1 1/1    Running   0          31m
[k8s-configuration]$ kubectl logs mongo-express-797845bd97-p9grr
Waiting for mongodb-service:27017...
Welcome to mongo-express
-----
Mongo Express server listening at http://0.0.0.0:8081
Server is open to allow connections from anyone (0.0.0.0)
basicAuth credentials are "admin:pass", it is recommended you change this in your config.js!
Database connected
Admin Database connected
[k8s-configuration]$
```

Mongo express external service



Mongo-express additional deployment:

```
32      key: mongo-root-password
33      - name: ME_CONFIG_MONGODB_SERV
34      valueFrom:
35      configMapKeyRef:
36      | name: mongodb-configmap
37      | key: database_url
38  ---
39  apiVersion: v1
40  kind: Service
41  metadata:
42  | name: mongo-express-service
43  spec:
44  | selector:
45  | | app: mongo-express
46  | type: LoadBalancer
47  | ports:
48  | | protocol: TCP
49  | | port: 8081
50  | | targetPort: 8081
51  | | nodePort: 30000|
```

How to make it an External Service?

- **type:** "Loadbalancer"
..assigns service an external IP address and so accepts external requests
- **nodePort:** must be between 30000-32767

Port for external IP address

```
[k8s-configuration]$ kubectl apply -f mongo-express.yaml
deployment.apps/mongo-express unchanged
service/mongo-express-service created
[k8s-configuration]$ kubectl get service
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)
AGE
kubernetes     ClusterIP  10.96.0.1    <none>       443/TCP
62m
mongo-express-service LoadBalancer  10.96.178.16  <pending>   8081:30000/
TCP 6s
mongodb-service ClusterIP  10.96.86.105  <none>       27017/TCP
26m
[k8s-configuration]$
```

In actual k8s cluster , we will get external ip address instead of <pending>.

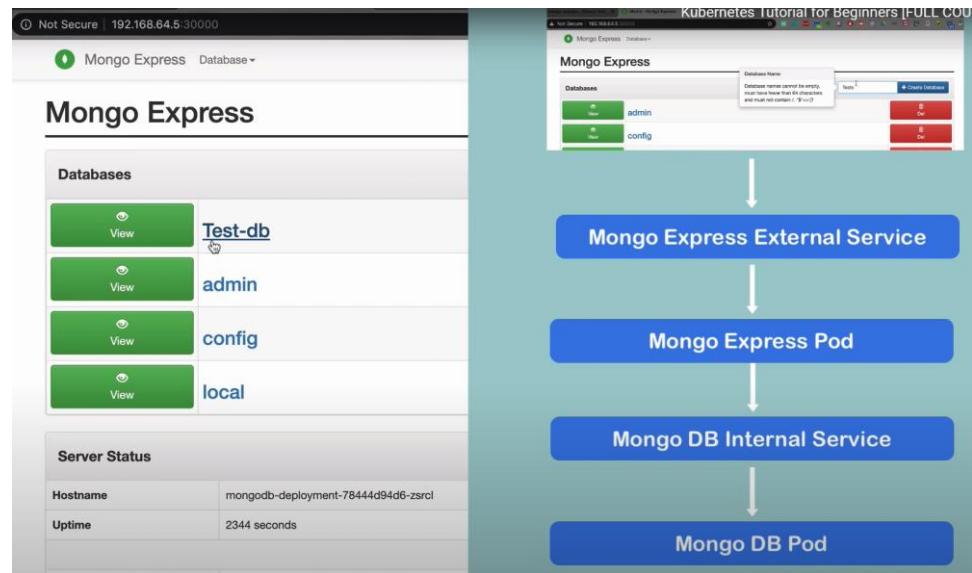
In minikube, we will need to set it by ourselves:

```
[k8s-configuration]$ kubectl get service
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)
AGE
kubernetes     ClusterIP  10.96.0.1    <none>       443/TCP
67m
mongo-express-service LoadBalancer  10.96.178.16  <pending>   8081:30000/
TCP 5m7s
mongodb-service ClusterIP  10.96.86.105  <none>       27017/TCP
32m
[k8s-configuration]$ minikube service mongo-express-service
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | mongo-express-service | | http://192.168.64.5:30000 |
|-----|-----|-----|-----|
💡 Opening service default/mongo-express-service in default browser...
[k8s-configuration]$
```

Databases		Database Name	+ Create Database
	Test-db		
	admin		
	config		
	local		

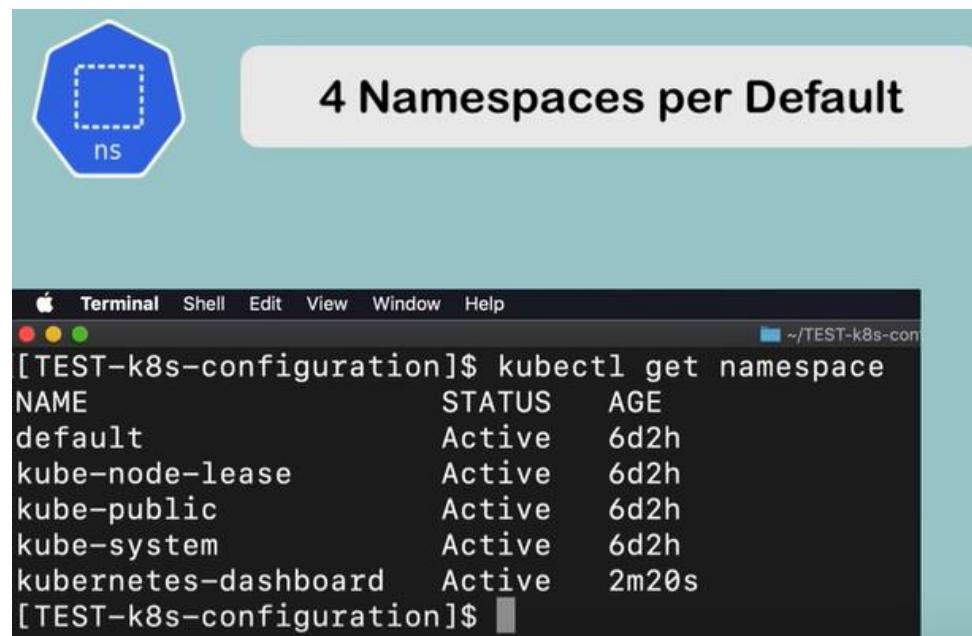
Server Status			
Hostname	mongodb-deployment-78444d94d6-zsrc1	MongoDB Version	4.2.3
Uptime	2344 seconds	Server Time	Sun, 16 Feb 2020 16:14:07 GMT
Current Connections	4	Available Connections	838856

Execution flow



Namespaces

Its like virtual cluster inside cluster



Kubernetes-dashboard – ONLY in mimikube

Kube-system: for master and kubectl / etc – DO NOT use it

Kube-public: configMap and other public data

- publicly accessible data
- A configmap, which contains cluster information

```
[TEST-k8s-configuration]$ kubectl cluster-info
Kubernetes master is running at https://192.168.64.5:8443
KubeDNS is running at https://192.168.64.5:8443/api/v1/namespaces/kube-system/services/kube-dns:dns
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
[TEST-k8s-configuration]$
```

Kube-node-lease: heartbeat of all nodes

- heartbeats of nodes
- each node has associated lease object in namespace
- determines the availability of a node

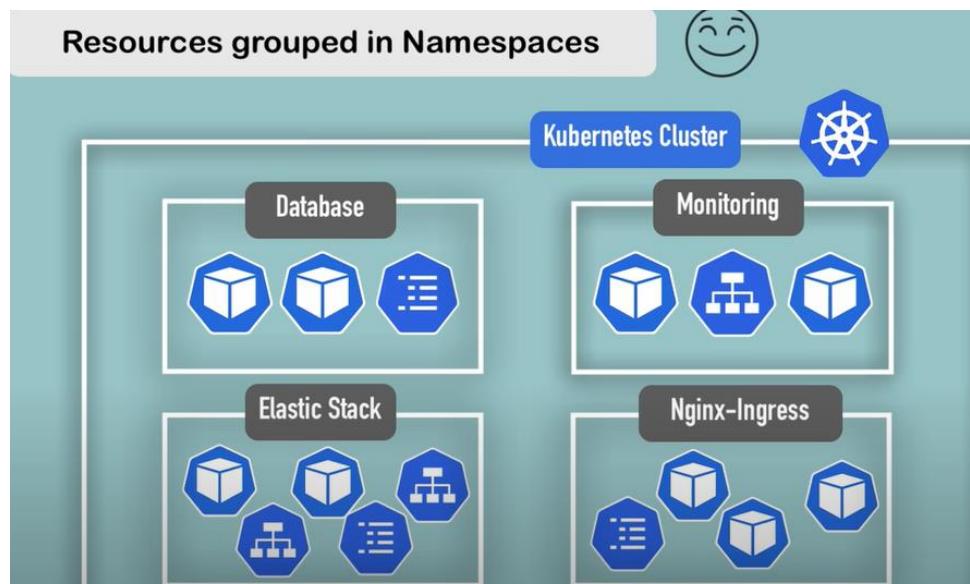
Default: for my resources

To create namesapce:

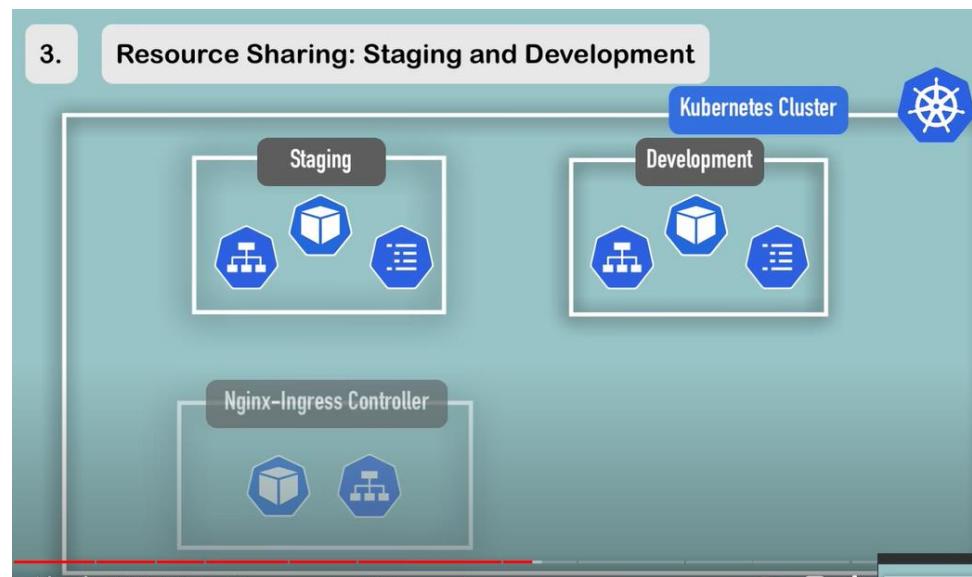
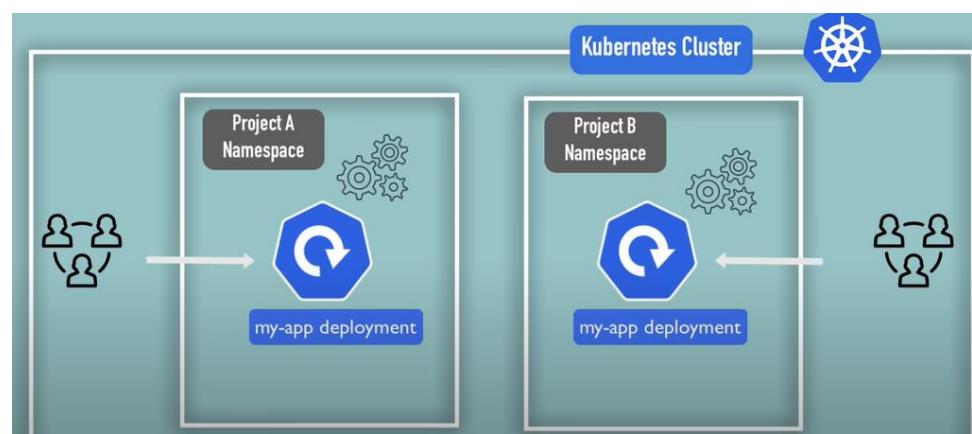
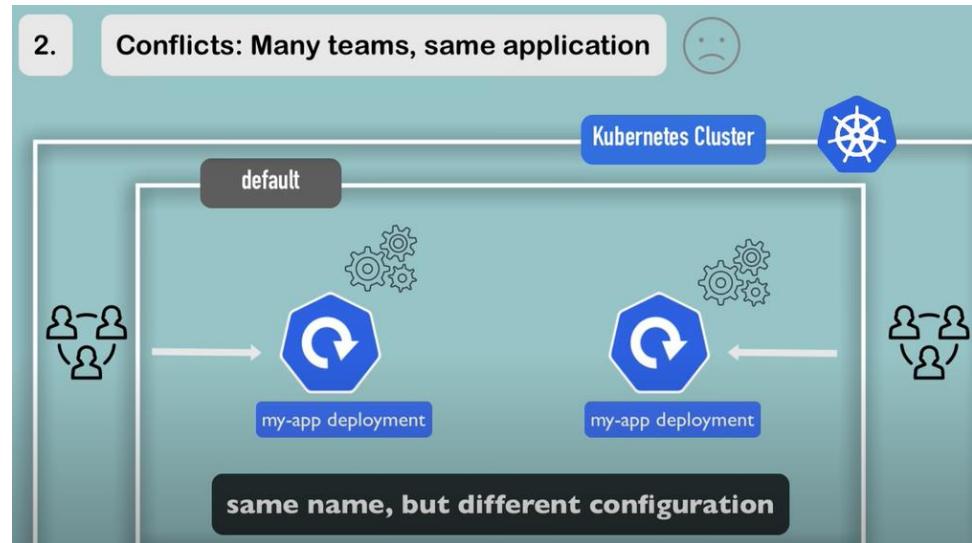
Kubectl create namespace my-namespace

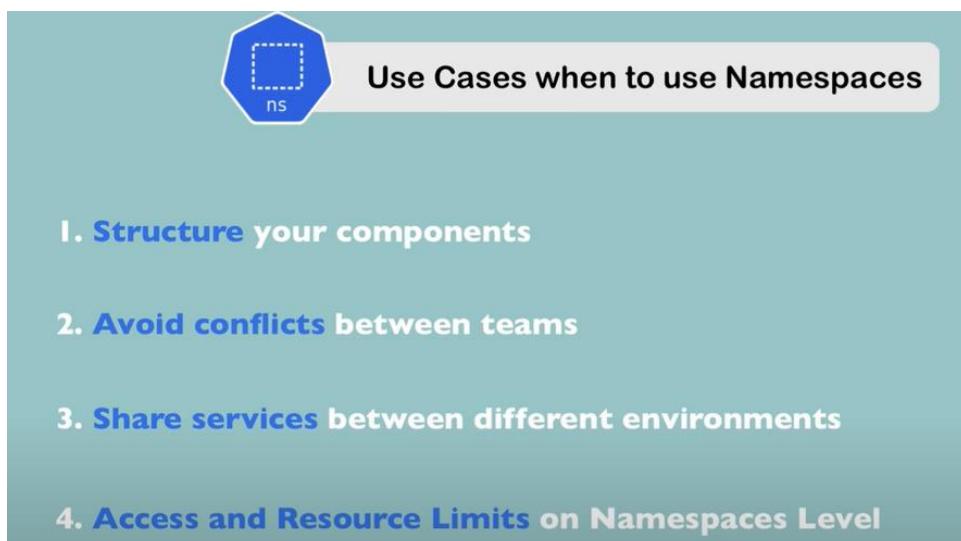
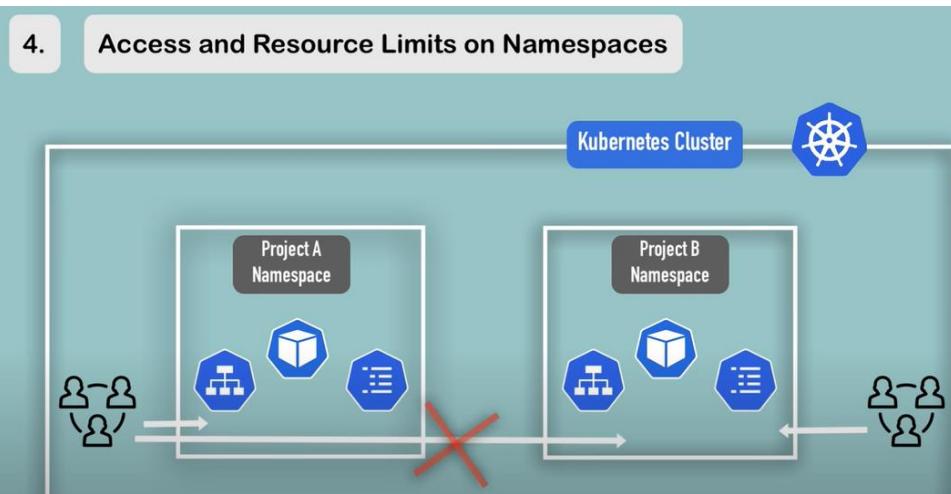
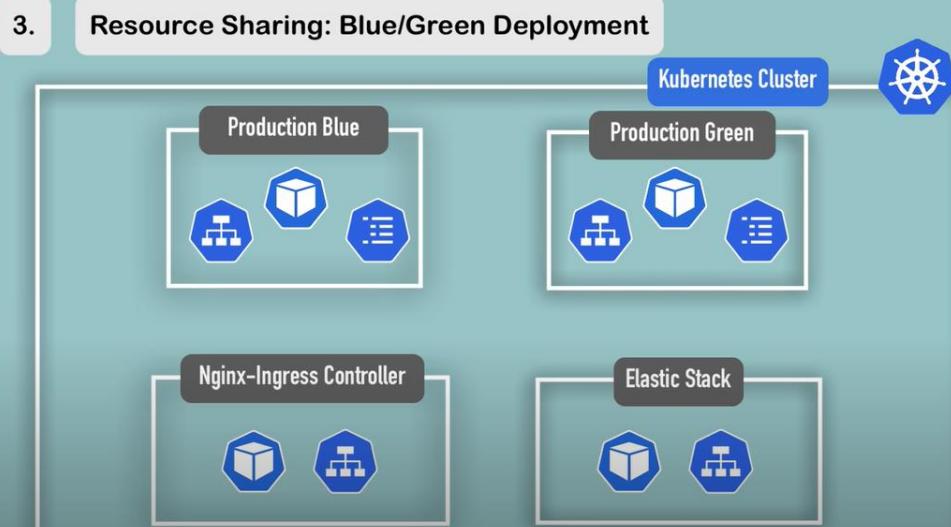
Create a namespace with a configuration file

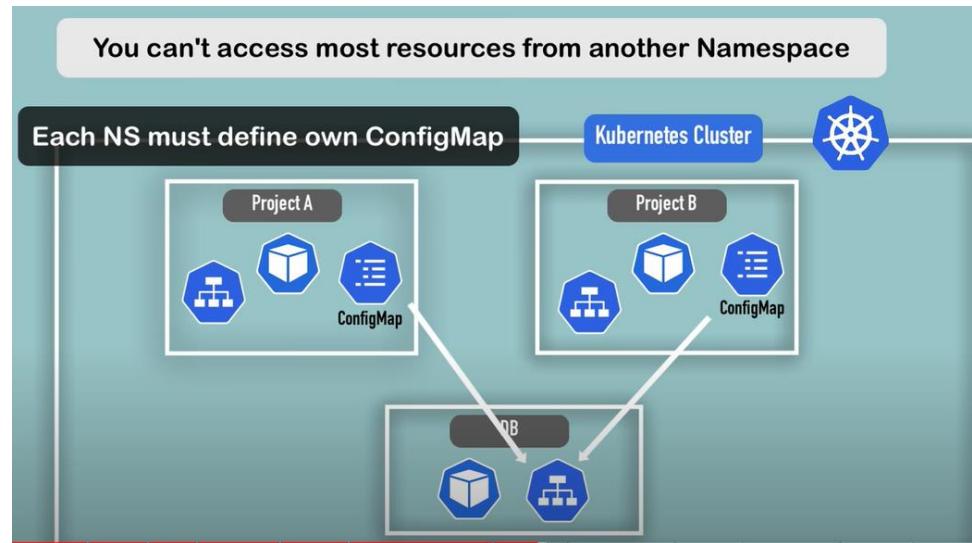
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-configmap
  namespace: my-namespace
data:
  db_url: mysql-service.database
```



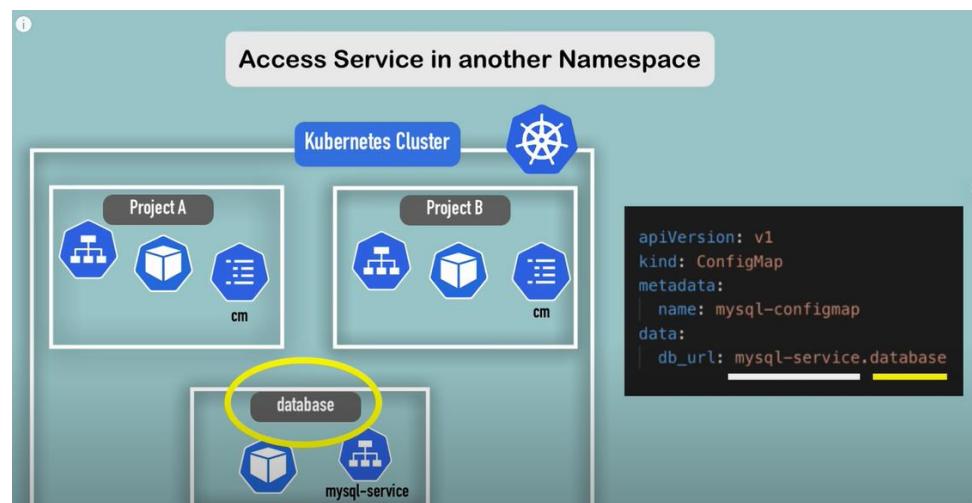
Putting same name application in different namespace:







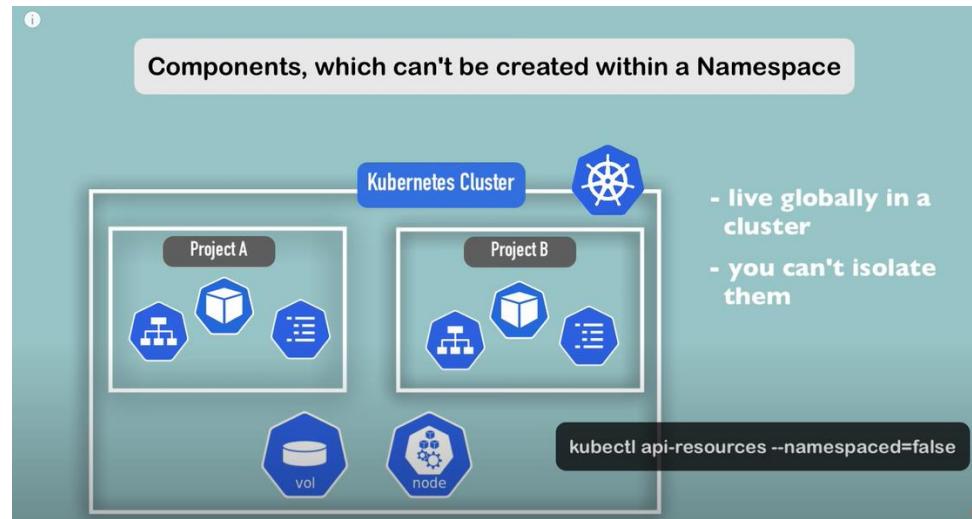
And secret file



Node and vol components cannot live inside namespace

To get the global components:

Kubectl api-resources --namecpased=false



Create component inside namespace:

Kubectl apply -f mysql-configmap.yaml --namespace=my-namespace

Inside yaml file: (better documented and in automatic way, will be deployed correctly!!)

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-configmap
  namespace: my-namespace
data:
  db_url: mysql-service.database
```

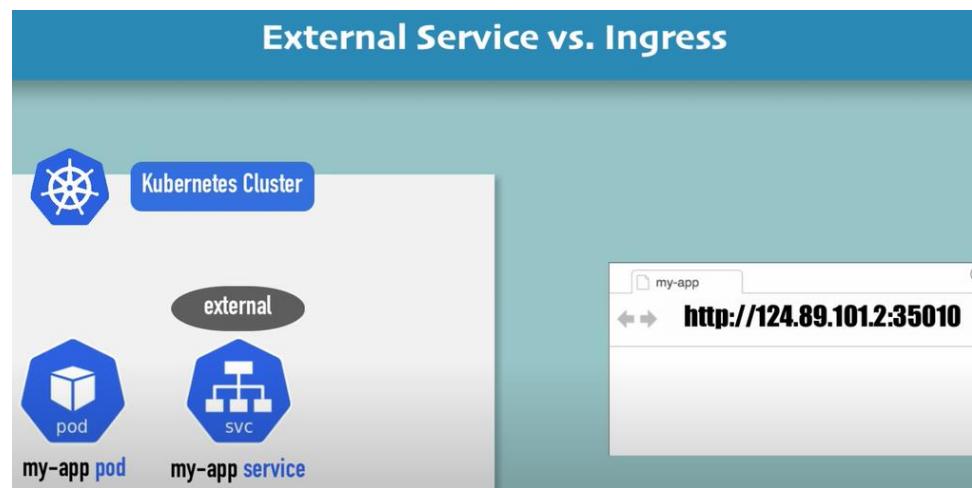
Kubectx – 3rd party tool to change active namespace from default to different one

Inside it there is executables: kubens to change active namespace

```
[~]$ kubens
default
kube-node-lease
kube-public
kube-system
kubernetes-dashboard
my-namespace
[~]$ kubens my-namespace
Context "minikube" modified.
Active namespace is "my-namespace"
[~]$ kubens
default
kube-node-lease
kube-public
kube-system
kubernetes-dashboard
my-namespace
```

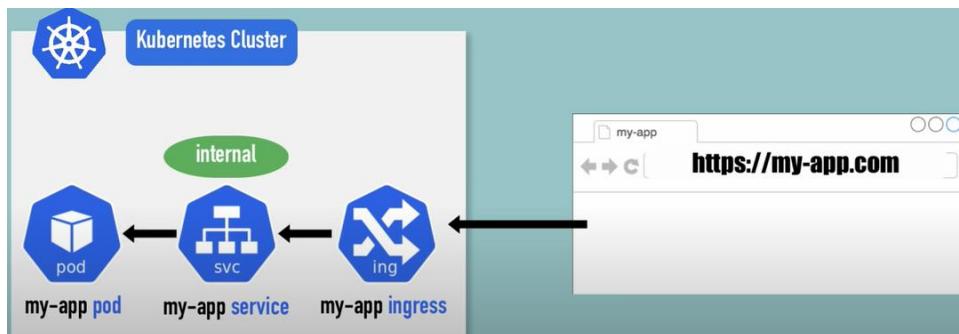
Ingress

External service good for tests / quick deployment



This is the target:

<Secure protocol>://Domain-name



Example YAML File: External Service

```
apiVersion: v1
kind: Service
metadata:
  name: myapp-external-service
spec:
  selector:
    app: myapp
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
      nodePort: 35010
```

Assign external IP address to service

Example YAML File: Ingress

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: myapp-ingress
spec:
  rules:
    - host: myapp.com
      http:
        paths:
          - backend:
              serviceName: myapp-internal-service
              servicePort: 8080
```

kind: Ingress

Routing rules:



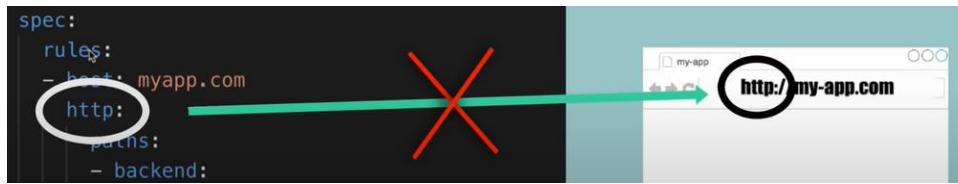
Forward request to the internal service.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: myapp-ingress
spec:
  rules:
    - host: myapp.com
      http:
        paths:
          - backend:
              serviceName: myapp-internal-service
              servicePort: 8080
```

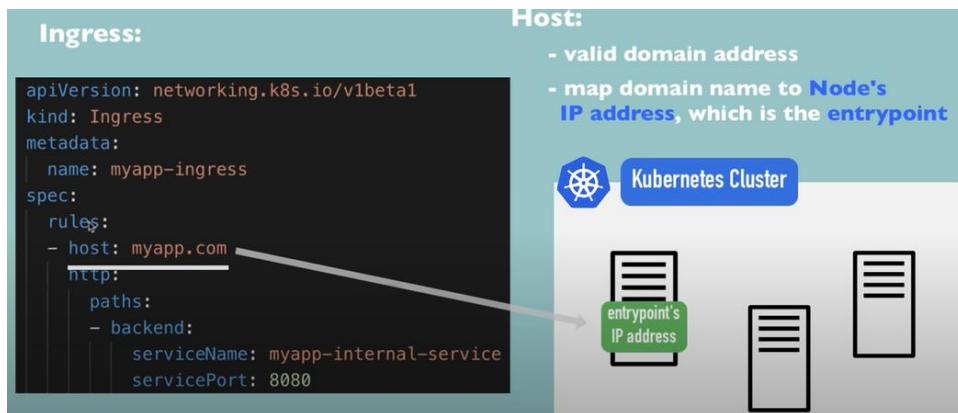
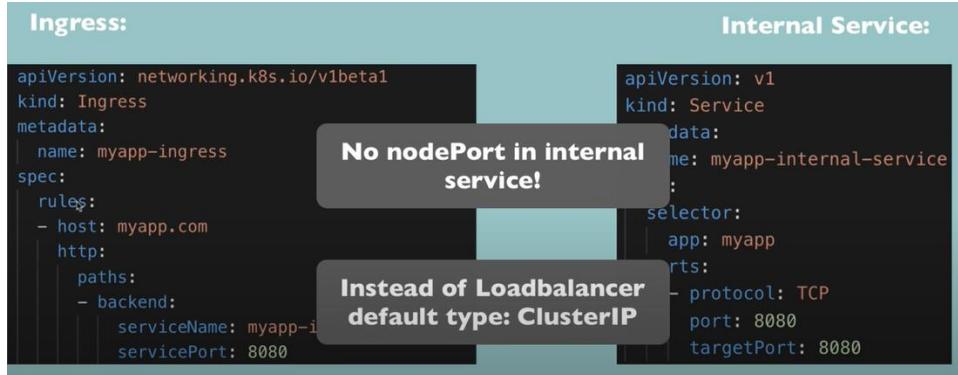
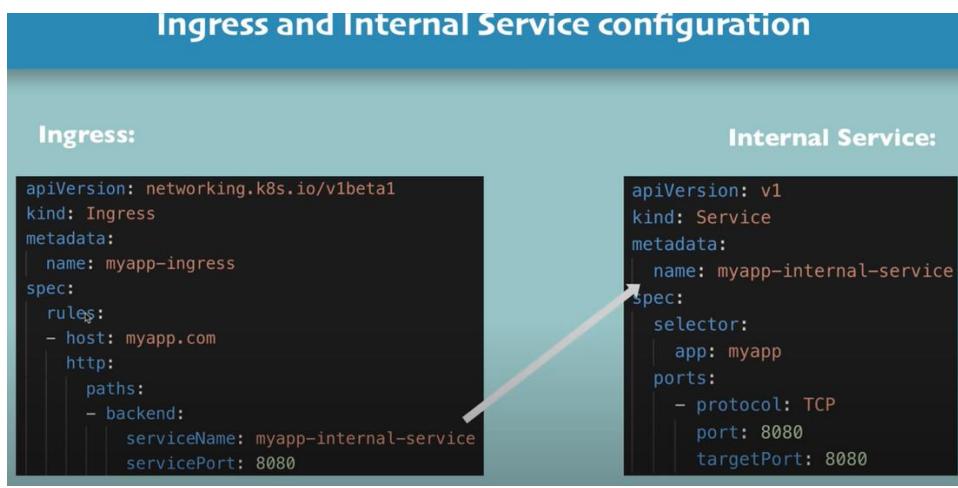
Routing rules:

Forward request to the internal service.

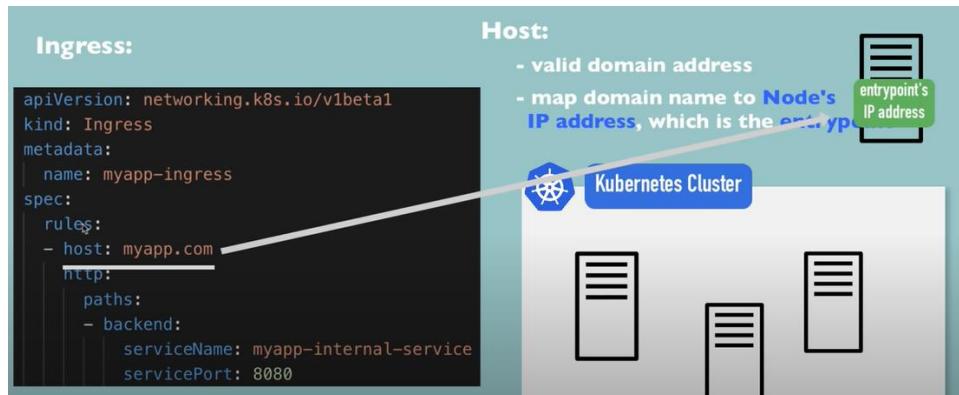




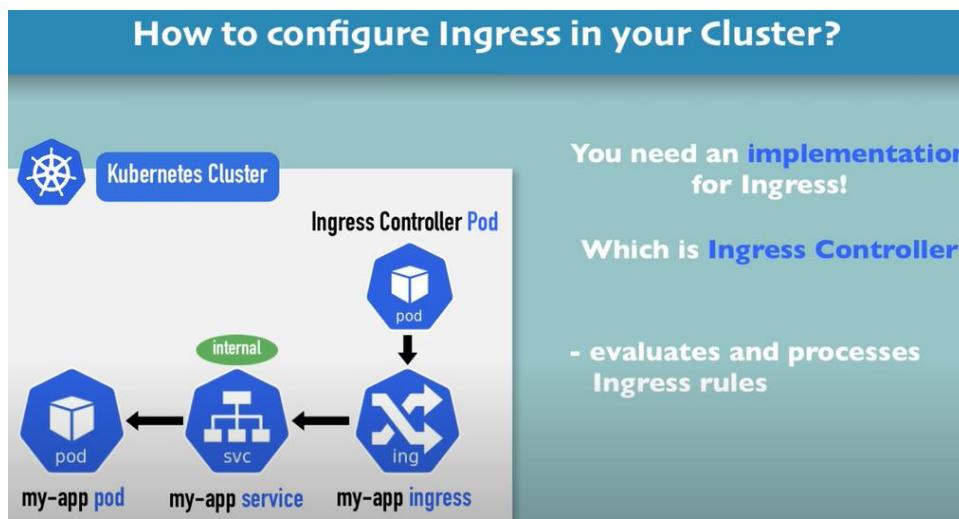
- host: myapp.com
 http: **= 2. Step: Incoming Request gets forwarded to internal service**
 paths:
 - backend:



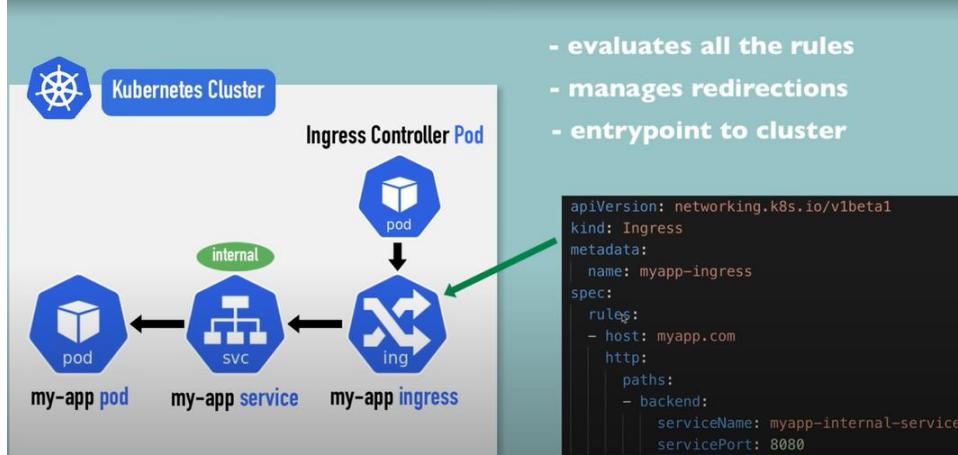
Additional option:



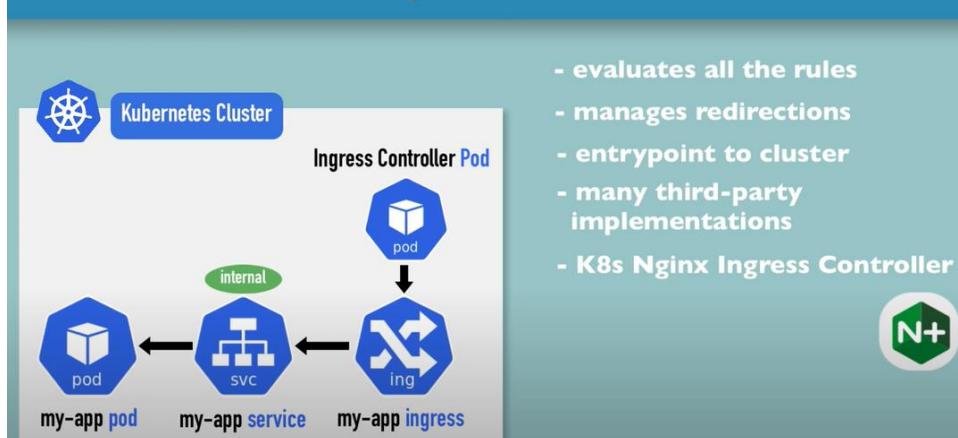
Ingress Controller



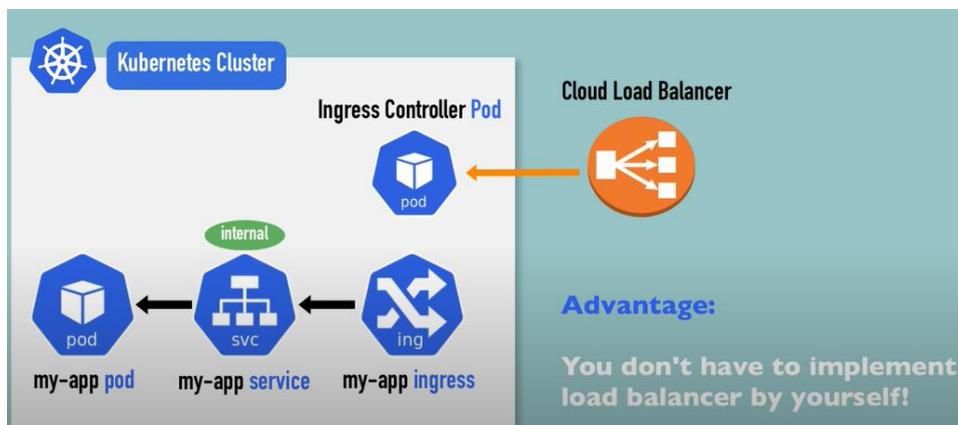
What is Ingress Controller? 🤔

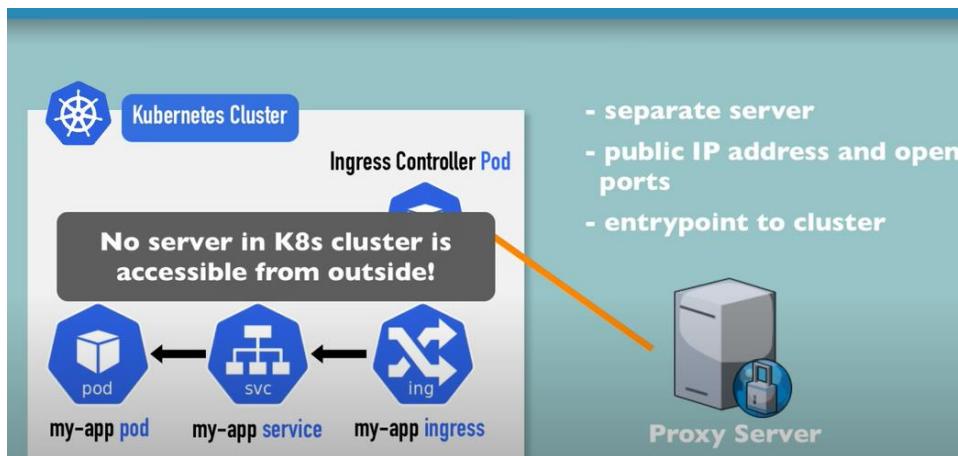
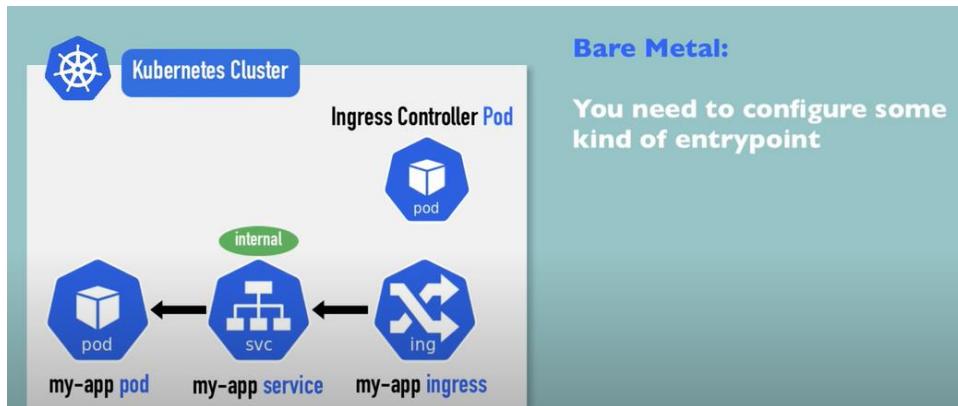


What is Ingress Controller? 🤔



K8s on cloud:





Implementation

Install Ingress Controller in Minikube

```
[~]$ minikube addons enable ingress
✓ ingress was successfully enabled
```

Automatically starts the K8s Nginx implementation of Ingress Controller

Install Ingress Controller in Minikube

```
[~]$ minikube addons enable ingress
✓ ingress was successfully enabled
```

```
[~]$ kubectl get pod -n kube-system
NAME                                         READY   STATUS    RESTARTS   AGE
coredns-6955765f44-bdrxd                   1/1    Running   0          5d1h
coredns-6955765f44-x94rx                   1/1    Running   0          5d1h
etcd-minikube                                1/1    Running   3          22d
kube-addon-manager-minikube                  1/1    Running   3          22d
kube-apiserver-minikube                     1/1    Running   3          22d
kube-controller-manager-minikube             1/1    Running   50         22d
kube-proxy-6g8k4                            1/1    Running   3          22d
kube-scheduler-minikube                     1/1    Running   49         22d
nginx-ingress-controller-6fc5bcc8c9-wd4g9   1/1    Running   0          30h
storage-provisioner                          1/1    Running   0          32h
```

Ingress config file:

```
! dashboard-ingress.yaml ●

1  apiVersion: networking.k8s.io/v1beta1
2  kind: Ingress
3  metadata:
4    name: dashboard-ingress
5    namespace: kubernetes-dashboard
6  spec:
7    rules:
8      - host: dashboard.com
9        http:
10          paths:
11            - backend:
12              serviceName: kubernetes-dashboard
13              servicePort: 80|
```

It will forward to the internal service (ClusterIP TYPE)

```
[~]$ kubectl get all -n kubernetes-dashboard
NAME                                         READY   STATUS    RESTARTS   AGE
pod/dashboard-metrics-scraper-7b64584c5c-9729k   1/1    Running   0          6d17h
pod/kubernetes-dashboard-79d9cd965-fvrbt        1/1    Running   0          6d17h

NAME           TYPE      CLUSTER-IP     EXTERNAL-IP   PORT(S)
AGE
service/dashboard-metrics-scraper   ClusterIP  10.96.188.182 <none>       8000/TCP
17d
service/kubernetes-dashboard        ClusterIP  10.96.220.185 <none>       80/TCP
17d

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/dashboard-metrics-scraper   1/1     1           1           17d
deployment.apps/kubernetes-dashboard        1/1     1           1           17d

NAME           DESIRED  CURRENT    READY   AGE
replicaset.apps/dashboard-metrics-scraper-7b64584c5c 1        1           1           17d
replicaset.apps/kubernetes-dashboard-79d9cd965      1        1           1           17d
```

How to resolve ip address to domain name (dashboard.com) ?

K8s will assign ip address to ingress:

```
[~]$ kubectl apply -f dashboard-ingress.yaml
ingress.networking.k8s.io/dashboard-ingress created
[~]$ kubectl get ingress -n kubernetes-dashboard
NAME          HOSTS      ADDRESS   PORTS   AGE
dashboard-ingress  dashboard.com        80      14s
[~]$ kubectl get ingress -n kubernetes-dashboard --watch
NAME          HOSTS      ADDRESS   PORTS   AGE
dashboard-ingress  dashboard.com  192.168.64.5  80      42s
c^C[~]$
```

In /etc/hosts file on the PC/ laptop add the domain name:

```
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1      localhost
255.255.255.255 broadcasthost
::1            localhost
192.168.64.5    dashboard.com
```

Ingress backend

```
[~]$ kubectl describe ingress dashboard-ingress -n kubernetes-dashboard
Name:           dashboard-ingress
Namespace:      kubernetes-dashboard
Address:        192.168.64.5
Default backend: default-http-backend:80 (<none>)
Rules:
  Host          Path  Backends
  ----          ---   -----
  dashboard.com      kubernetes-dashboard:80 (172.17.0.3:9090)
Annotations:
  kubectl.kubernetes.io/last-applied-configuration: {"apiVersion":"networking.k8s.io/v1beta1","kind":"Ingress","metadata":{"annotations":{},"name":"dashboard-ingress","namespace":"kubernetes-dashboard"},"spec":{"rules":[{"host":"dashboard.com","http":{"paths":[{"backend":{"serviceName":"kubernetes-dashboard","servicePort":80}}]}]}}
```

```
Events:
  Type  Reason  Age   From           Message
  ----  -----  --   --   Ingress kubernetes-dashboard/dashboard-ingress
  Normal CREATE  12m   nginx-ingress-controller  Ingress kubernetes-dashboard/dashboard-ingress
  Normal UPDATE  12m   nginx-ingress-controller  Ingress kubernetes-dashboard/dashboard-ingress
```

```
Default backend: default-http-backend:80 (<none>)
```

If u call a service that is not mapped to any backend – this will be the default

Configure Default Backend in Ingress

```
[~]$ kubectl describe ingress myapp-ingress
Name:           myapp-ingress
Namespace:      default
Address:
Default backend: default-http-backend:80 (<none>)
Rules:
  Host    Path  Backends
  ----  ----  -----
  myapp.com        myapp-internal-service:8080 (<none>)
```

```
apiVersion: v1
kind: Service
metadata:
  name: default-http-backend
spec:
  selector:
    app: default-response-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

+ to create a pod to return meaningful message

Additional use cases for ingress

Many services on same domain

Multiple paths for same host

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - path: /analytics
        backend:
          serviceName: analytics-service
          servicePort: 3000
      - path: /shopping
        backend:
          serviceName: shopping-service
          servicePort: 8080
```

- Example Google

- One domain but many services

Multiple paths for same host

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - path: /analytics
        backend:
          serviceName: analytics-service
          servicePort: 3000
      - path: /shopping
        backend:
          serviceName: shopping-service
```

Multiple sub-domains or domains

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: analytics.myapp.com
    http:
      paths:
        backend:
          serviceName: analytics-service
          servicePort: 3000
  - host: shopping.myapp.com
    http:
      paths:
        backend:
          serviceName: shopping-service
```

Instead of:

<http://myapp.com/analytics>

<http://analytics.myapp.com>

Multiple sub-domains or domains

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: analytics.myapp.com
    http:
      paths:
        backend:
          serviceName: analytics-service
          servicePort: 3000
  - host: shopping.myapp.com
    http:
      paths:
        backend:
          serviceName: shopping-service
```

```
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - path: /analytics
        backend:
          serviceName: analytics-service
          servicePort: 3000
      - path: /shopping
        backend:
          serviceName: shopping-service
          servicePort: 8080
```

Instead of **1 host and multiple path**

You have now **multiple hosts with 1 path**. Each host represents a subdomain.

SSL configuration

Installation to get crt key/value:

<https://www.stechies.com/installing-openssl-windows-10-11/>

Configuring TLS Certificate - https://

The diagram illustrates the flow of traffic from the outside world through an Ingress controller to a service. It shows a blue hexagon labeled 'Kubernetes Cluster' containing a blue icon of a ship's wheel. An arrow points from this cluster to a blue hexagon labeled 'Ingress' containing a white 'X'. Another arrow points from the Ingress to a blue hexagon labeled 'secret' containing a lock icon.

```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
| name: tls-example-ingress
spec:
  tls:
    - hosts:
      - myapp.com
      secretName: myapp-secret-tls
    rules:
      - host: myapp.com
        http:
          paths:
            - path: /
              backend:
                serviceName: myapp-internal-service
                servicePort: 8080
  
```

- **tls**
- **secretName**

Configuring TLS Certificate - https://

A white arrow points from the 'secretName' field in the Ingress configuration to the 'name' field in the Secret resource. This indicates that the Ingress is referencing the Secret.

```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
| name: tls-example-ingress
spec:
  tls:
    - hosts:
      - myapp.com
      secretName: myapp-secret-tls
    rules:
      - host: myapp.com
        http:
          paths:
            - path: /
              backend:
                serviceName: myapp-internal-service
                servicePort: 8080
  
```

```

apiVersion: v1
kind: Secret
metadata:
| name: myapp-secret-tls
| namespace: default
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
  type: kubernetes.io/tls
  
```

Tls config file:

Configuring TLS Certificate - https://

The diagram highlights specific fields in the Secret resource configuration:

- I) Data keys need to be "tls.crt" and "tls.key"**
- 2) Values are file contents
NOT file paths/locations**
- 3) Secret component must be in the same namespace as the Ingress component**

```

apiVersion: v1
kind: Secret
metadata:
| name: myapp-secret-tls
| namespace: default
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
  type: kubernetes.io/tls
  
```

Configuring TLS Certificate - https://

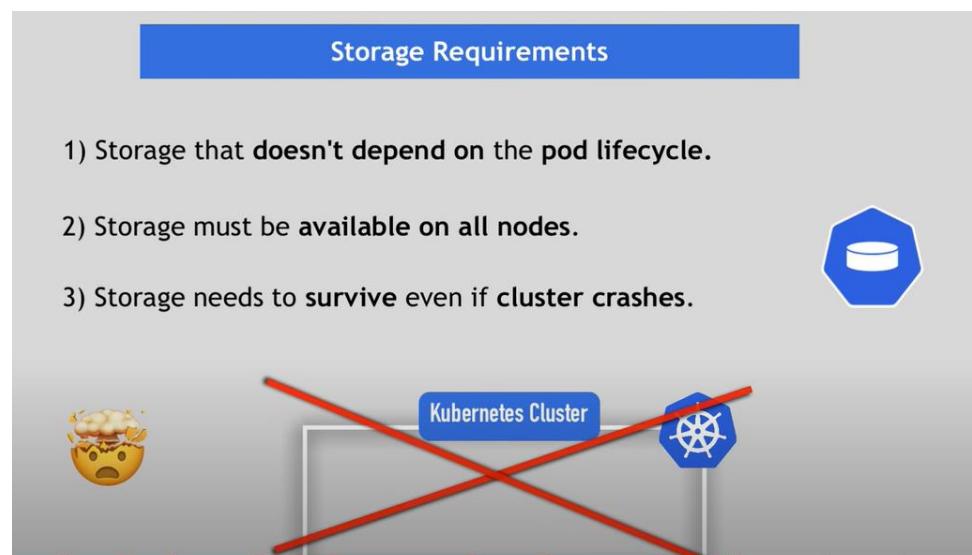
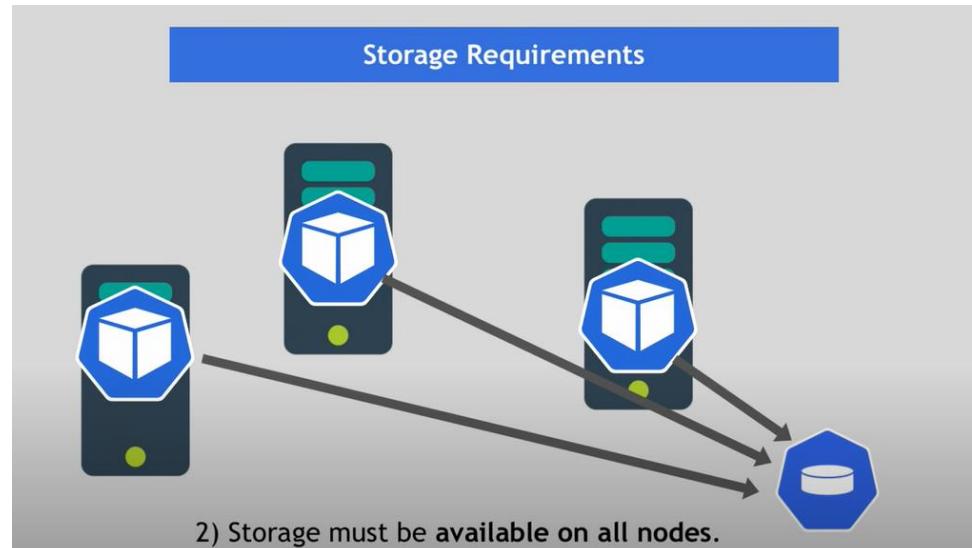


```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: tls-example-ingress
spec:
  tls:
    - hosts:
      - myapp.com
      secretName: myapp-secret-tls
  rules:
    - host: myapp.com
      http:
        paths:
          - path: /
            backend:
              serviceName: myapp-internal-service
              servicePort: 8080

```

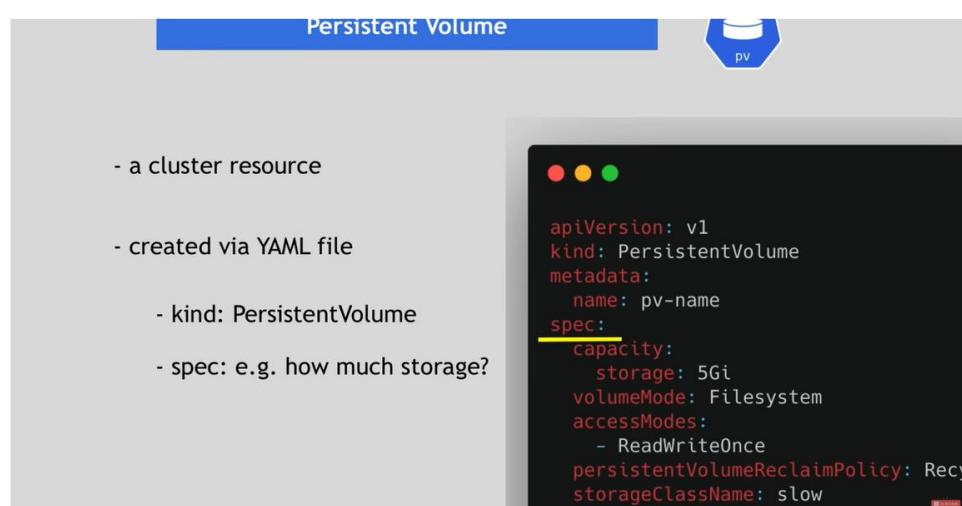
Persistent Data using VOLUMES

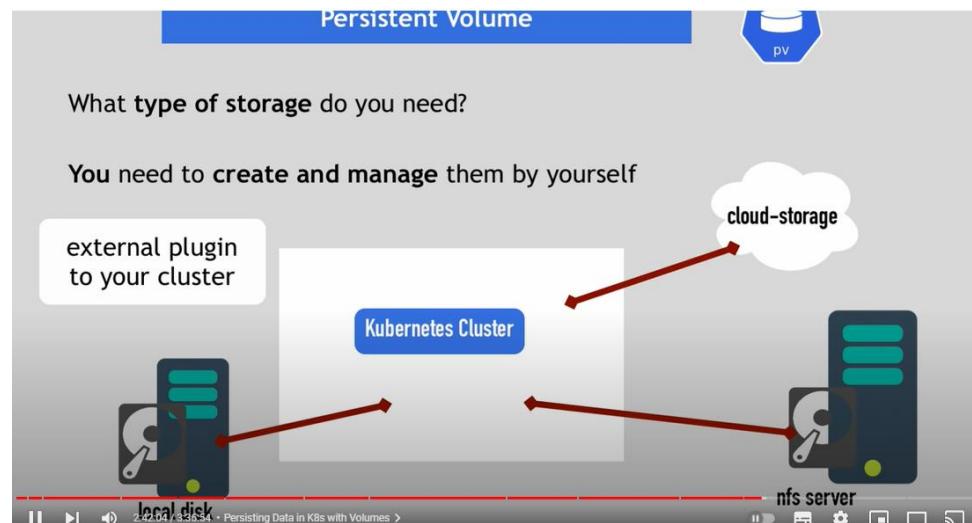


Another Use Case for persistent storage



Yaml file:





Persistent Volume YAML Example

NFS Storage

Use that physical storages in the **spec** section

How much:

Additional params, like access:

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-name
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.0
  nfs:
    path: /dir/path/on/nfs/server
    server: nfs-server-ip-address

```

Persistent Volume YAML Example

Google Cloud

How much:

Google Cloud parameters:

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: test-volume
  labels:
    failure-domain.beta.kubernetes.io/zone: us-central1-a__us-central1-b
spec:
  capacity:
    storage: 400Gi
  accessModes:
    - ReadWriteOnce
  gcePersistentDisk:
    pdName: my-data-disk
    fsType: ext4

```

Persistent Volume YAML Example

Local storage

Depending on storage type, spec attributes differ

Node Affinity: 

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: example-pv
spec:
  capacity:
    storage: 100Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage
  local:
    path: /mnt/disks/ssd1
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/hostname
            operator: In
            values:
              - example-node

```

Local vs. Remote Volume Types

Each volume type has its own use case!

Local volume types violate 2. and 3. requirement for data persistence:

- ✖ Being tied to 1 specific node
- ✖ surviving cluster crashes

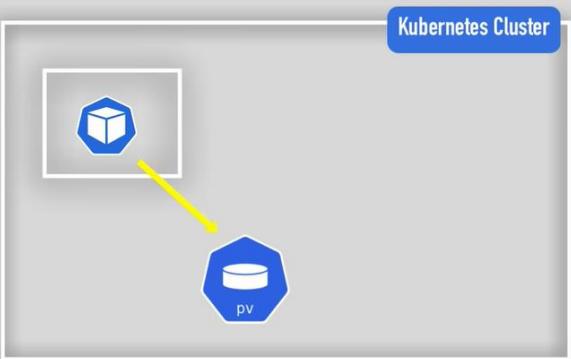
For DB persistence use remote storage!

K8s Administrator and K8s User

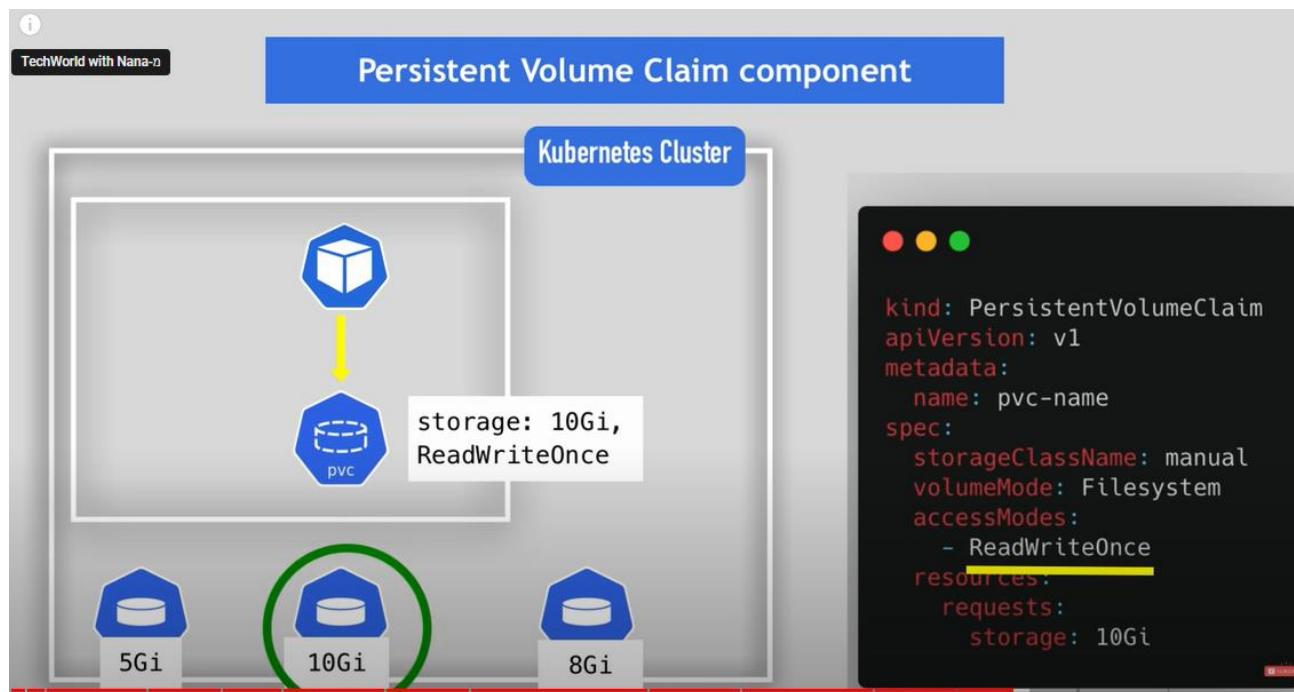
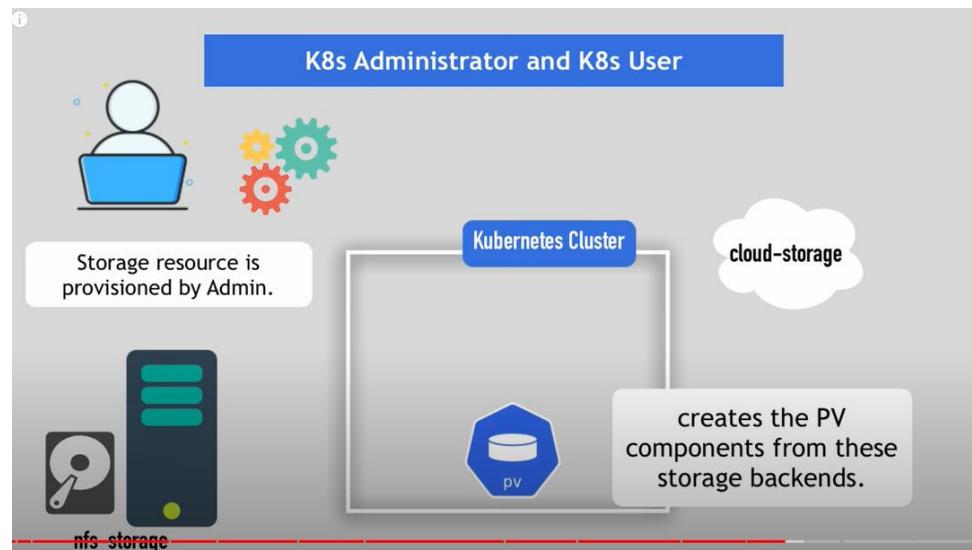
Who creates the Persistent Volumes and when? 🤔

..the Pod that **depends on** it is created

PV are resources that need to be there **BEFORE**..

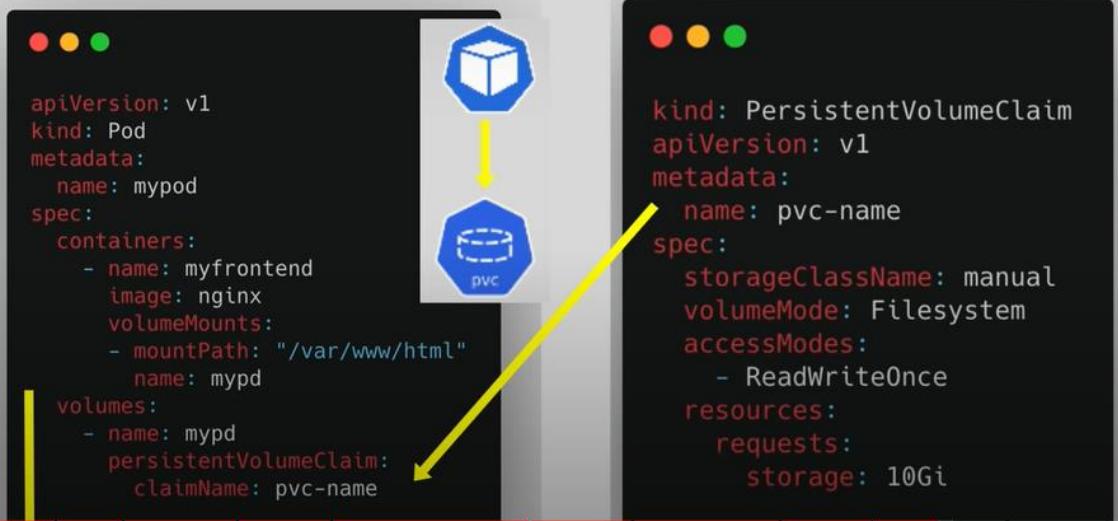


Kubernetes Cluster

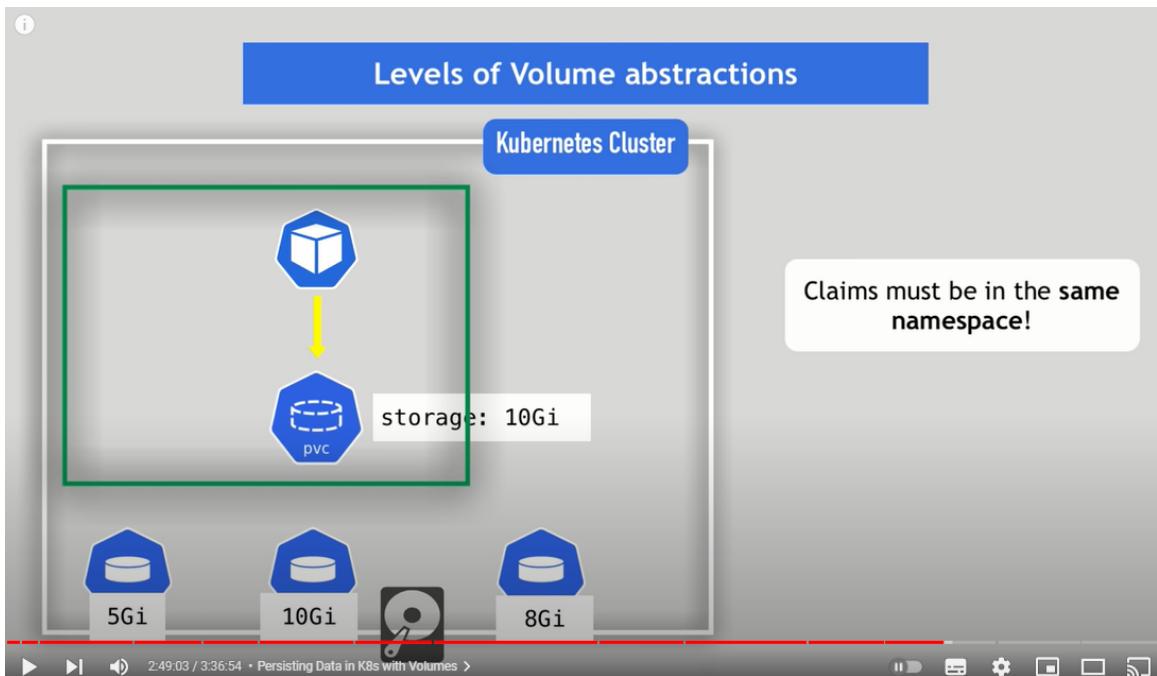


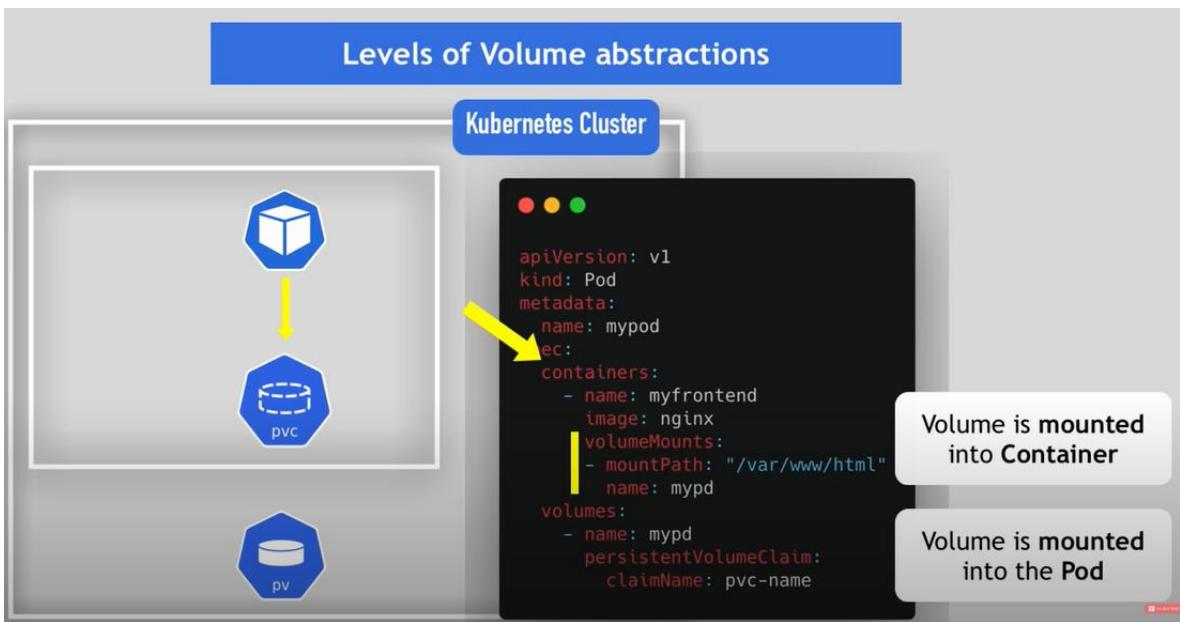
PersistentVolumeClaim component

Use that PVC in Pods configuration



Levels of Volume abstractions





DIFFERENT VOLUME TYPES

Different volume types

```

apiVersion: v1
kind: Pod
metadata:
  name: elastic
spec:
  selector:
    matchLabels:
      app: elastic
  template:
    metadata:
      labels:
        app: elastic
    spec:
      containers:
        - image: elastic/latest
          name: elastic-container
          ports:
            - containerPort: 9200
          volumeMounts:
            - name: es-persistent-storage
              mountPath: /var/lib/data
            - name: es-secret-dir
              mountPath: /var/lib/secret
            - name: es-config-dir
              mountPath: /var/lib/config
      volumes:
        - name: es-persistent-storage
          persistentVolumeClaim:
            claimName: es-pv-claim
        - name: es-secret-dir
        secret:
  
```

elastic-app

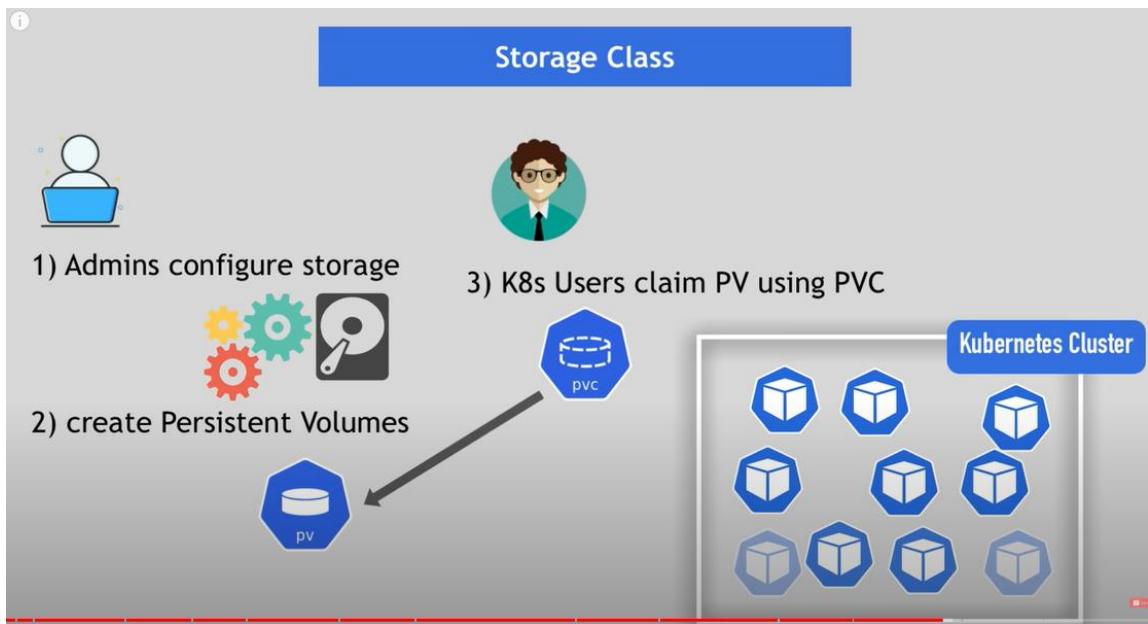
awsElasticBlockStore

pvc

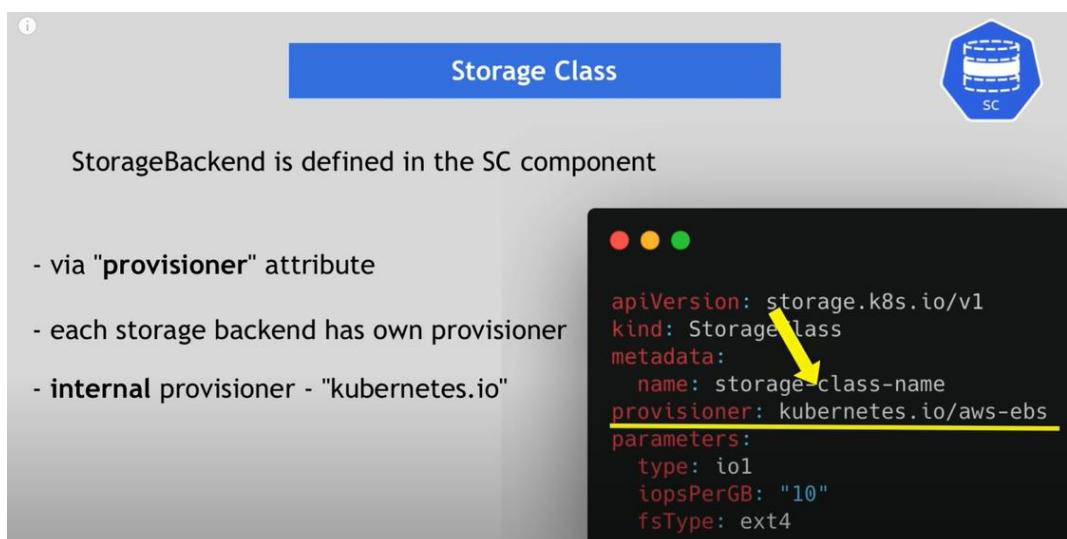
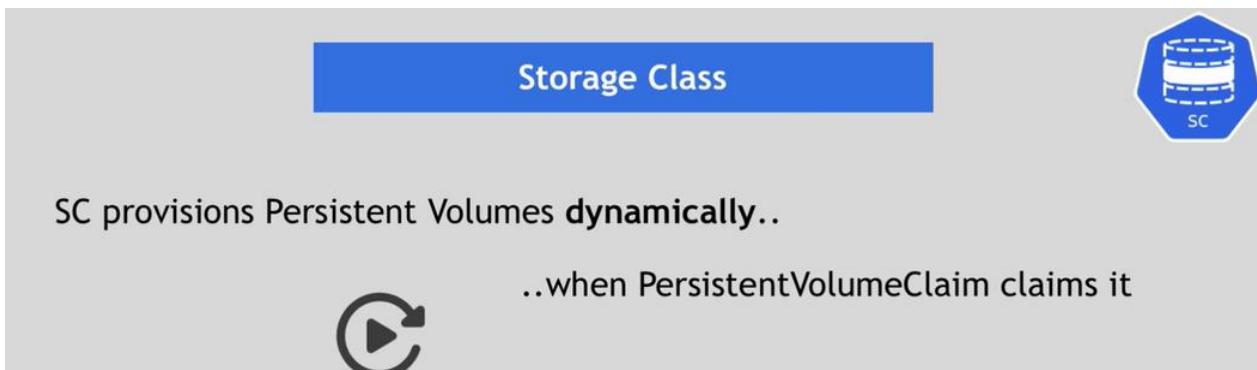
secret

cm

2:54:22 / 3:36:54 • Persisting Data in K8s with Volumes >



Storage class – 3rd party to provision PV (Persistent Volume) when PVC (Persistent Volume claim) claims it



also, there might be external provisioner !!

Storage Class



StorageBackend is defined in the SC component

- via "provisioner" attribute
- each storage backend has own provisioner
- **internal** provisioner - "kubernetes.io"
- **external** provisioner
- configure **parameters** for storage we want to request for PV

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: storage-class-name
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  iopsPerGB: "10"
  fsType: ext4
```

Storage Class



Another abstraction level

- abstracts underlying storage provider
- parameters for that storage

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: storage-class-name
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  iopsPerGB: "10"
  fsType: ext4
```

Storage Class usage



Requested by PersistentVolumeClaim



PVC Config

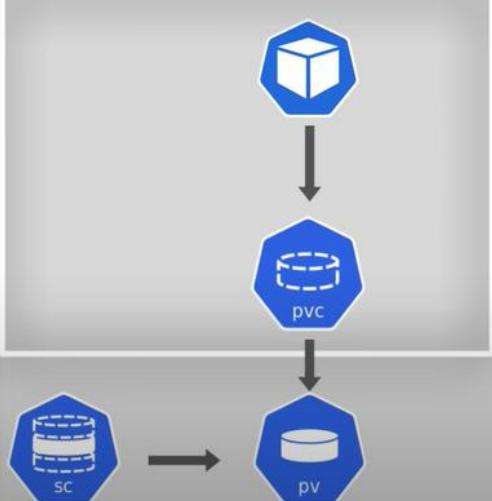
Storage Class Config

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: storage-class-name
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  iopsPerGB: "10"
  fsType: ext4
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: storage-class-name
```

Storage Class usage

Kubernetes Cluster



- 1) Pod claims storage via PVC
- 2) PVC requests storage from SC
- 3) SC creates PV that meets the needs of the Claim

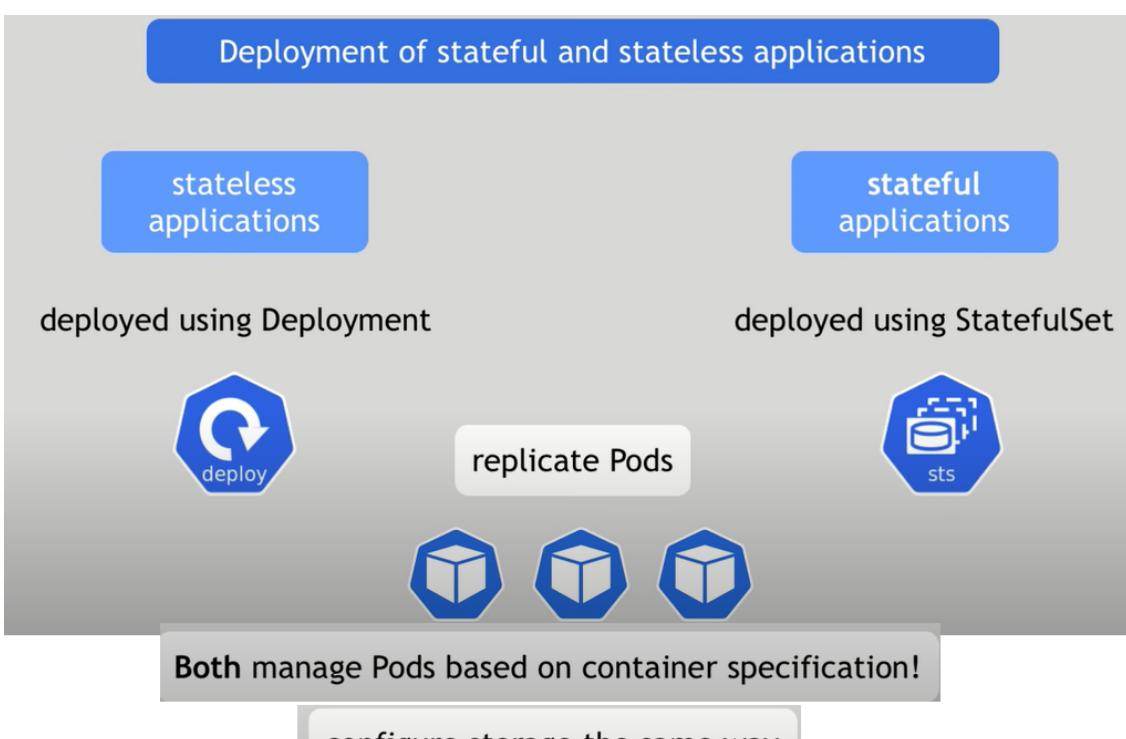
StatefulSet

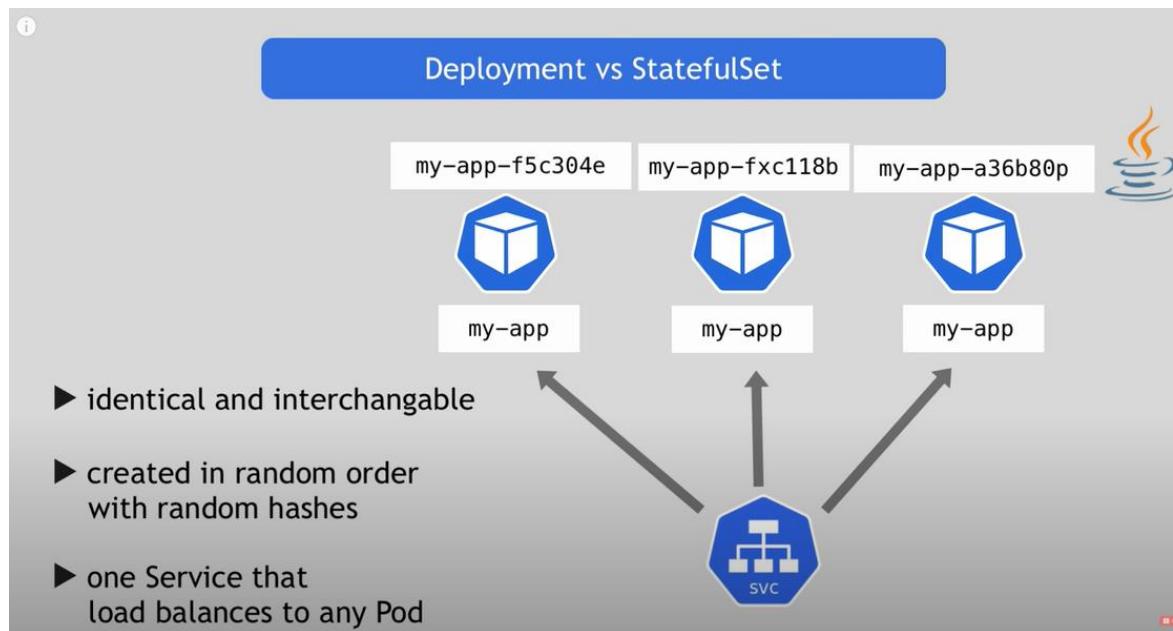
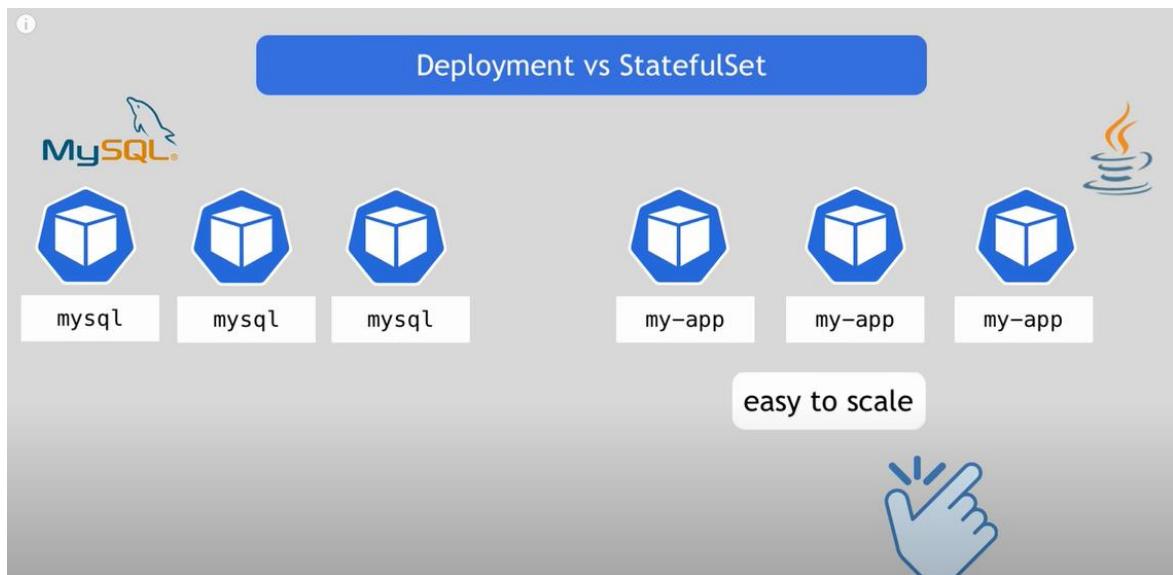
Kubernetes StatefulSet explained



- What is StatefulSet?
- Why StatefulSet is used?
- How StatefulSet works and how it's different from Deployment?

Usage:





Deployment vs StatefulSet



mysql



mysql



mysql

more difficult

- ▶ can't be created/deleted at same time
- ▶ can't be randomly addressed
- ▶ replica Pods are not identical
 - Pod Identity

Pod Identity

- sticky identity for each pod



ID-0

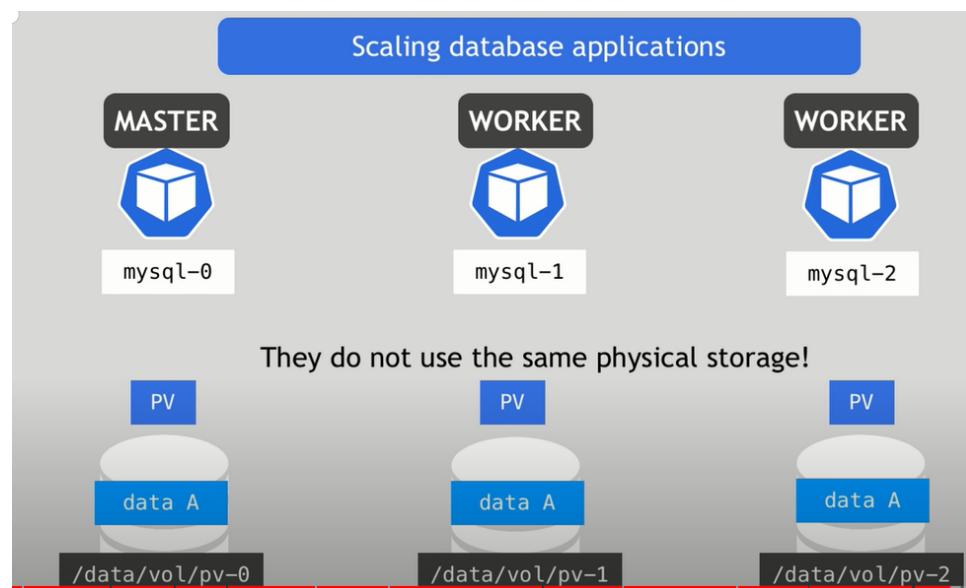
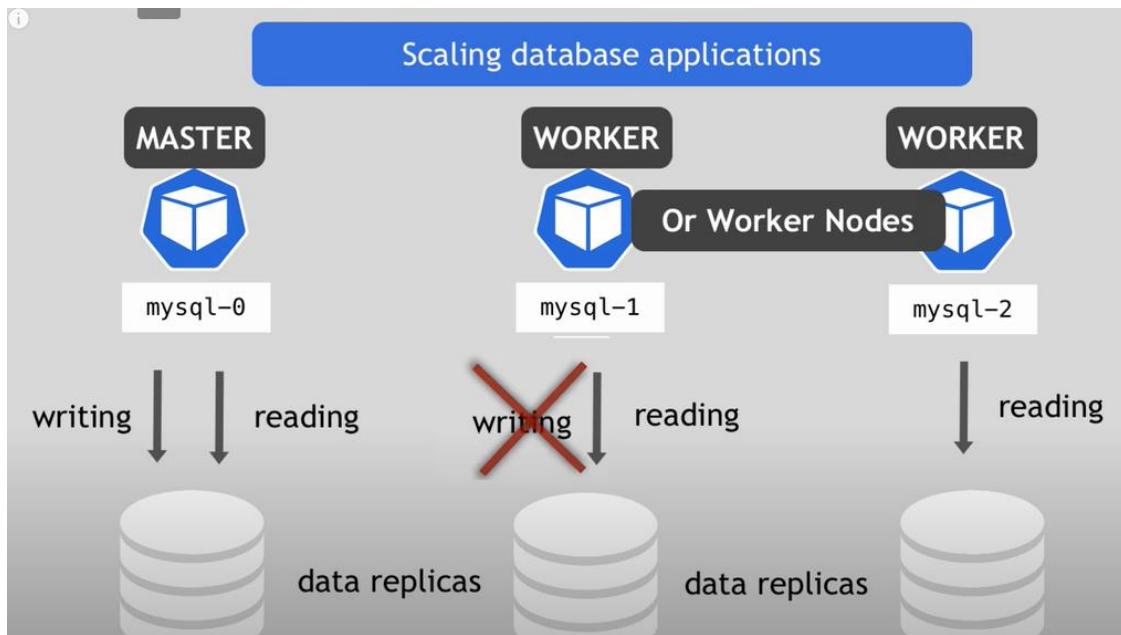


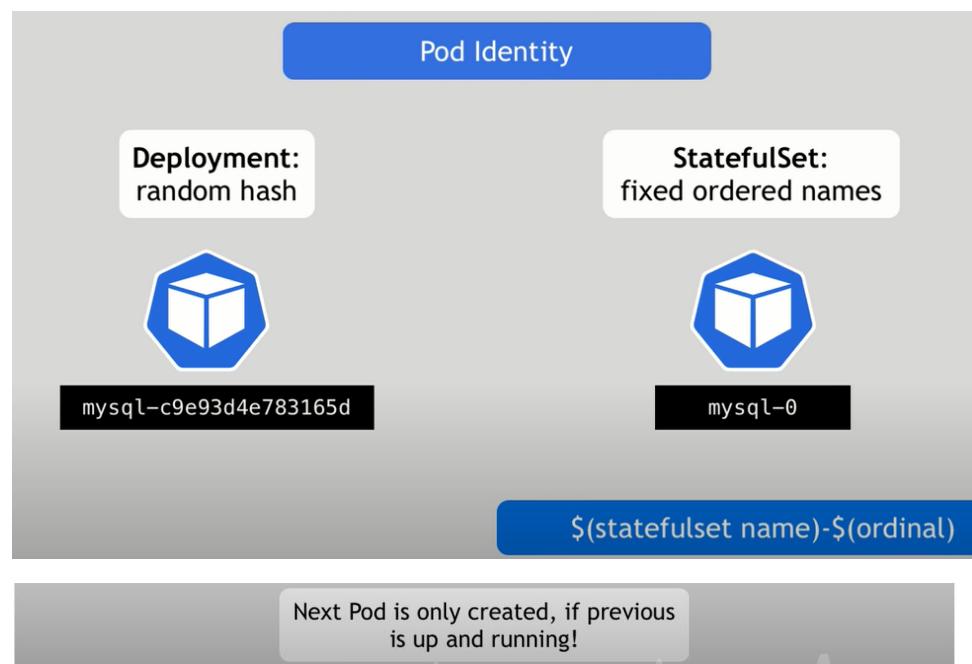
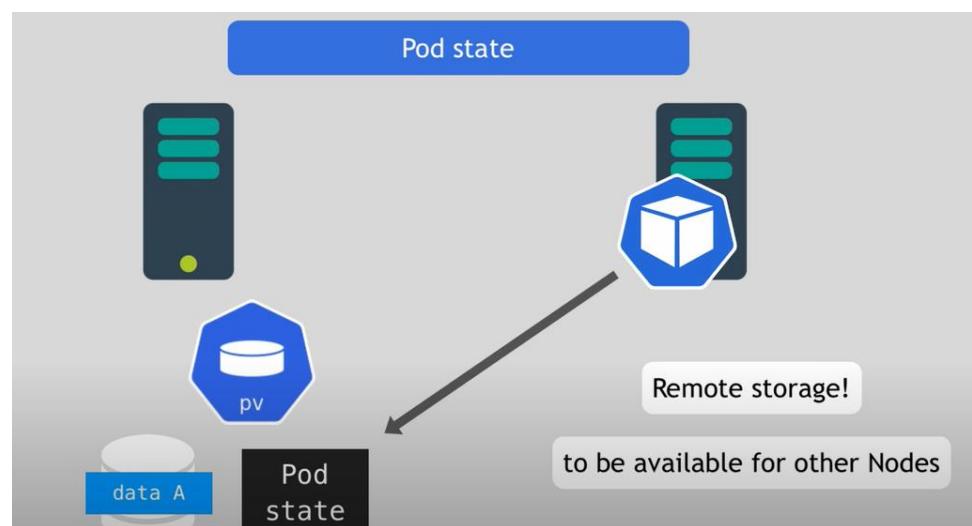
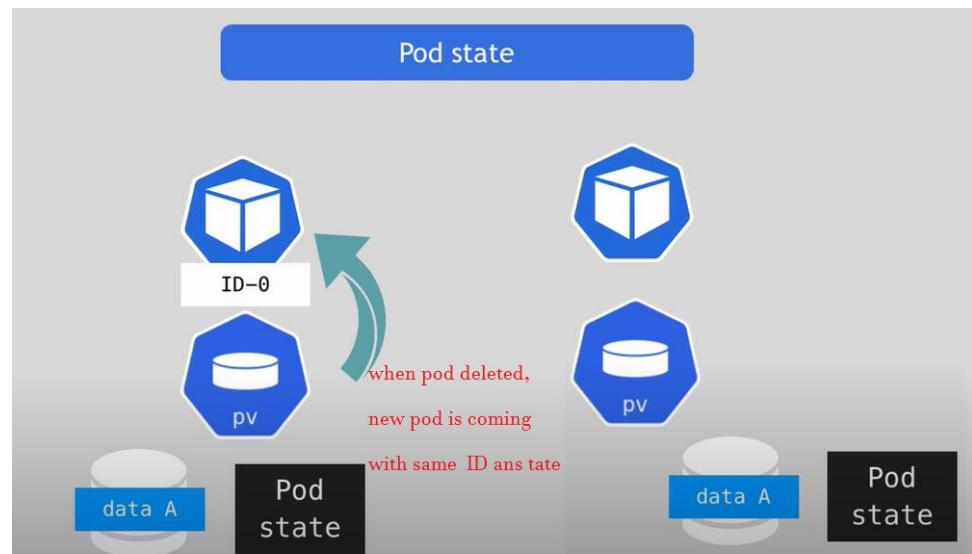
ID-1



ID-2

- created from **same specification**, but **not interchangeable!**
- persistent identifier across any re-scheduling





Pod Identity

Delete StatefulSet or scale down to 1 replica



mysql-0



mysql-1



mysql-2

Deletion in reverse order, starting from the last one

2 Pod endpoints

2) individual service name

mysql-0.svc2

mysql-1.svc2

mysql-2.svc2

1) loadbalancer service

Same as Deployment!

SVC1



2 characteristics

1) predictable pod name

mysql-0

2) fixed individual DNS name

mysql-0.svc2

When Pod restarts:

👉 IP address changes

👉 name and endpoint stays same

2 characteristics

Sticky identity



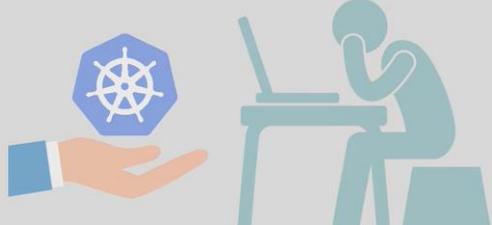
retain state

retain role



Replicating stateful apps

- ▶ it's complex
- ▶ Kubernetes helps you
- ▶ You still need to do a lot:
 - Configuring the cloning and data synchronization
 - Make remote storage available
 - Managing and back-up



Services

Types are :

ClusterIP service (default)

Headless Service

Kubernetes Services



- ▶ What is a Kubernetes Service and when we need it?
- ▶ Different **Service types** explained
 - ✓ ClusterIP Services
 - ✓ Headless Services
 - ✓ NodePort Services
 - ✓ LoadBalancer Services
- ▶ Differences between them and when to use which one

What is a Service and when we need it? 🤔

- ▶ Each Pod has its own IP address
 - ✖ Pods are ephemeral - are destroyed frequently!
- ▶ Service:
 - stable IP address
 - loadbalancing
 - loose coupling
 - within & outside cluster

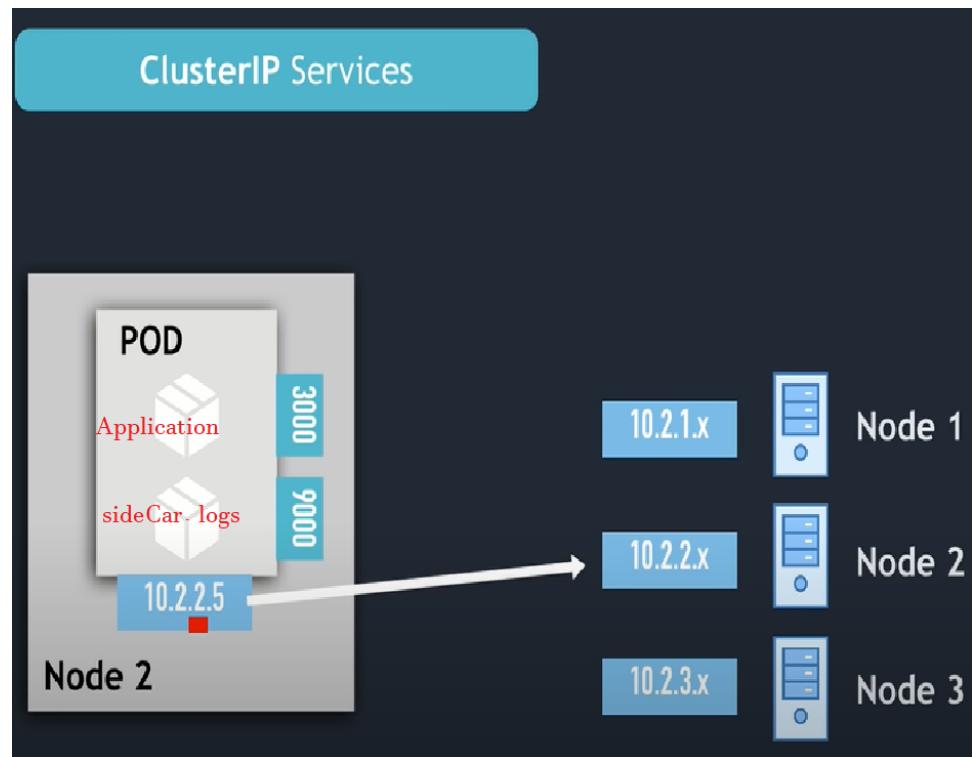
ClusterIP services

ClusterIP Services

- ▶ default type

No type specified here:

```
! service-clusterIP.yaml ×
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: microservice-one-service
5  spec:
6    selector:
7      app: microservice-one
8    ports:
9      - protocol: TCP
10     port: 3200
11     targetPort: 3000
```



```
kubectl get pod -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP
ATED NODE	READY	STATUS	RESTARTS	AGE	IP
mongodb-deployment-55577c4cbc-gkz8b	1/1	Running	0	29s	10.2.2.2
> <none>					
mongodb-deployment-55577c4cbc-z828l	1/1	Running	0	29s	10.2.1.4
> <none>					
mongodb-deployment-55577c4cbc-zv9h8	1/1	Running	0	29s	10.2.2.3
> <none>					

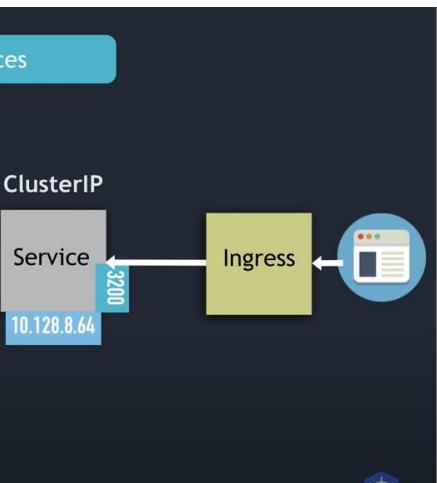
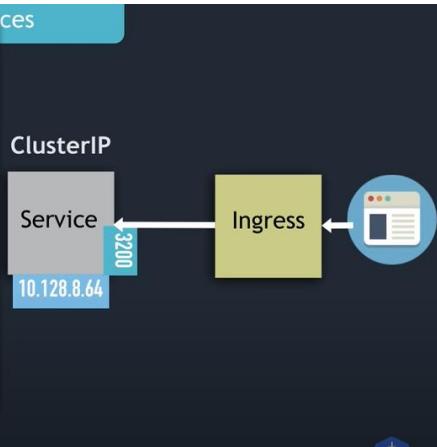
```
kubectl get pod -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP
ATED NODE	READY	STATUS	RESTARTS	AGE	IP
mongodb-deployment-55577c4cbc-gkz8b	1/1	Running	0	Node 2	10.2.2.2
> <none>					
mongodb-deployment-55577c4cbc-z828l	1/1	Running	0	Node 1	10.2.1.4
> <none>					
mongodb-deployment-55577c4cbc-zv9h8	1/1	Running	0	Node 2	10.2.2.3
> <none>					

Setting ingress – forward the request to the service

```
ess.yaml x 1     apiVersion: v1
            2     kind: Service
apiVersion: networking.k8s.io/v1beta1
kind: Ingress      name: microservice-one-service
metadata: 5       spec:
  name: ms-one-ingress
  annotations: app: microservice-one
    kubernetes.io/ingress.class: "nginx"
spec: 9           - protocol: TCP
rules: 10          port: 3200
  - host:1 microservice-one.com 3000
    http:12
      paths:
        - path: /
          backend:
            serviceName: microservice-one-service
            servicePort: 3200

```



Service Communication: selector

which Pods to forward the request to?

"selector"

- ▶ Pods are identified via **selectors**
 - ▶ key value pairs
 - ▶ **labels** of pods

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: microservice-one-service
5 spec:
6   selector:
7     app: microservice-one
8 ...
9
```

Service Communication: selector

```
which deployment has replicas? 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
478
479
480
481
482
483
484
485
486
487
488
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
578
579
579
580
581
582
583
584
585
586
587
588
588
589
589
590
591
592
593
594
595
596
597
598
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
678
679
679
680
681
682
683
684
685
686
687
688
688
689
689
690
691
692
693
694
695
696
697
697
698
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1957
1958
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2057
2058
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2067
2068
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2096
2097
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2127
2128
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2137
2138
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2157
2158
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2167
2168
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2177
2178
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2196
2197
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2227
2228
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2237
2238
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2247
2248
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2257
2258
2258
2259
2259
2260
2261
2262
2263
2264
2265

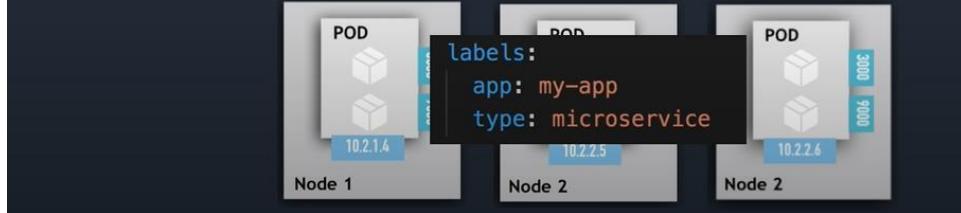
```

- ▶ Pods are in the `microservice-one` namespace
 - ▶ key value pairs in the `metadata` field
 - ▶ labels of pods

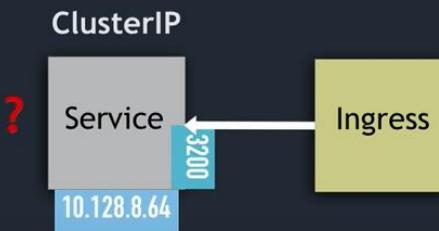
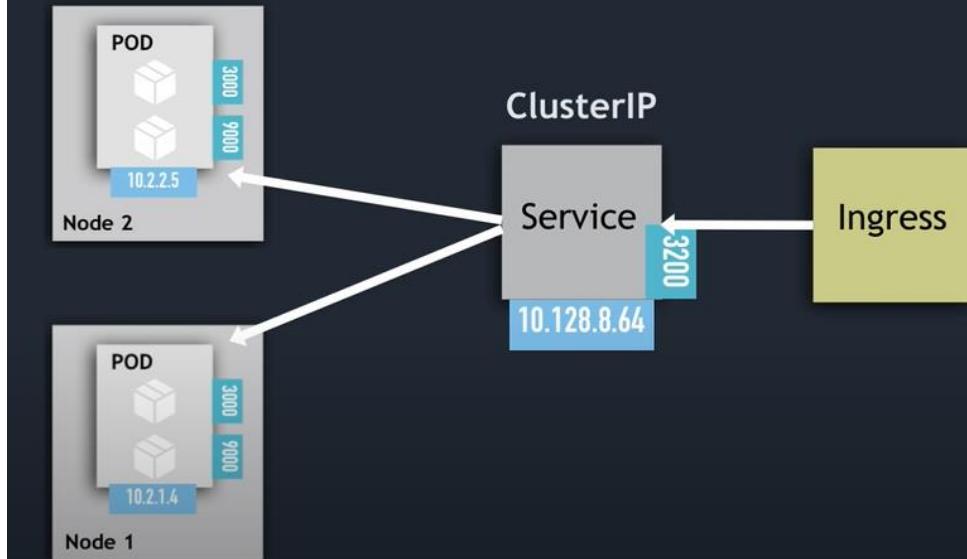
Service Communication: selector

- ▶ Svc matches all 3 replica
 - ▶ registers as Endpoints
 - ▶ must match ALL the selectors

```
  selector:  
    Service | app: my-app  
             | type: microservice
```

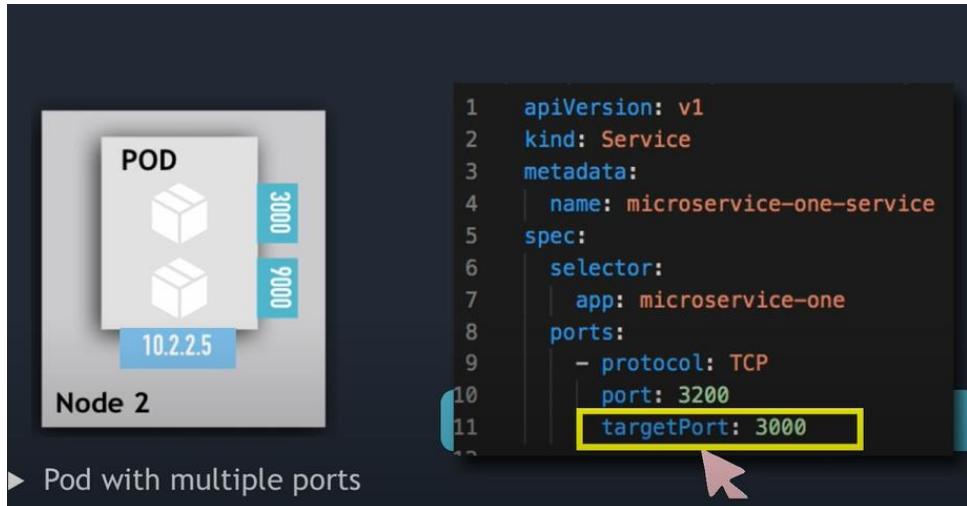


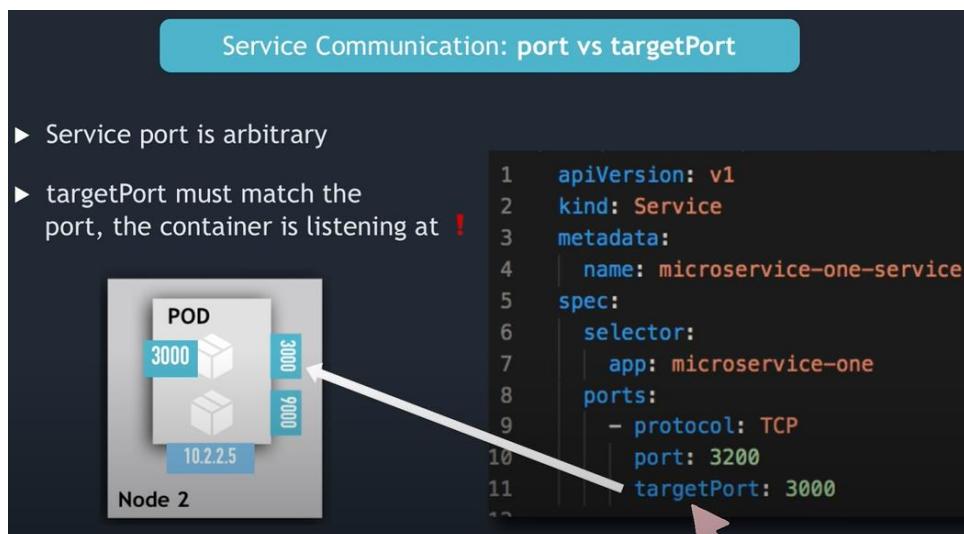
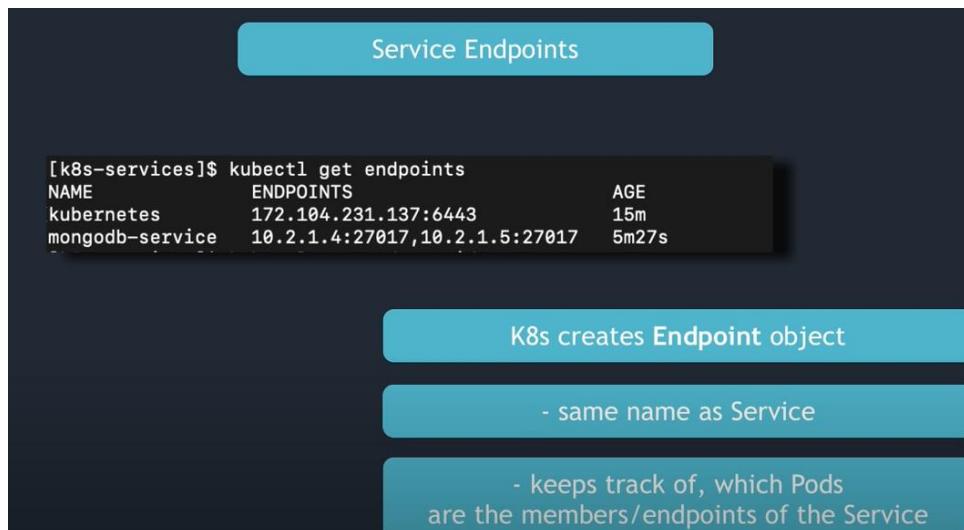
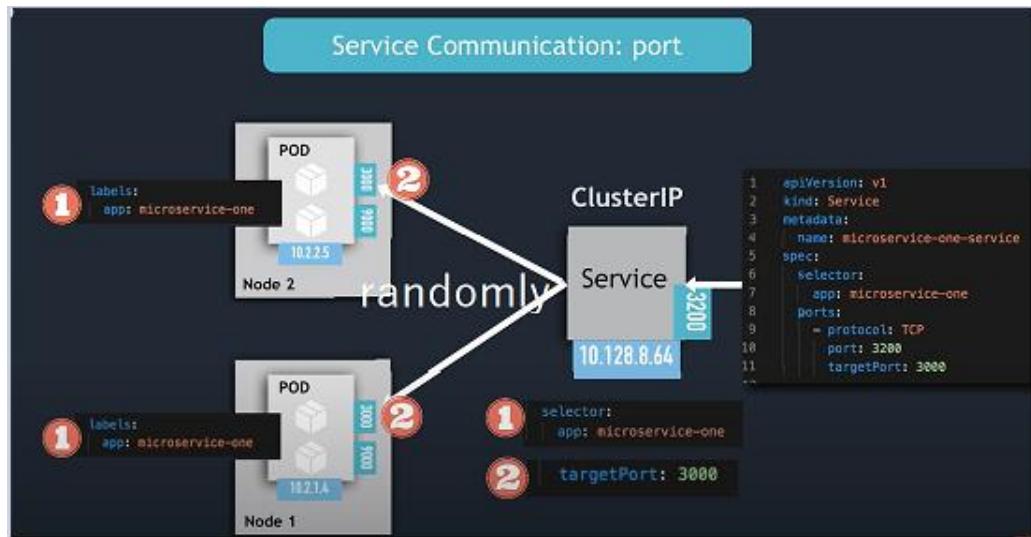
Service Communication: selector

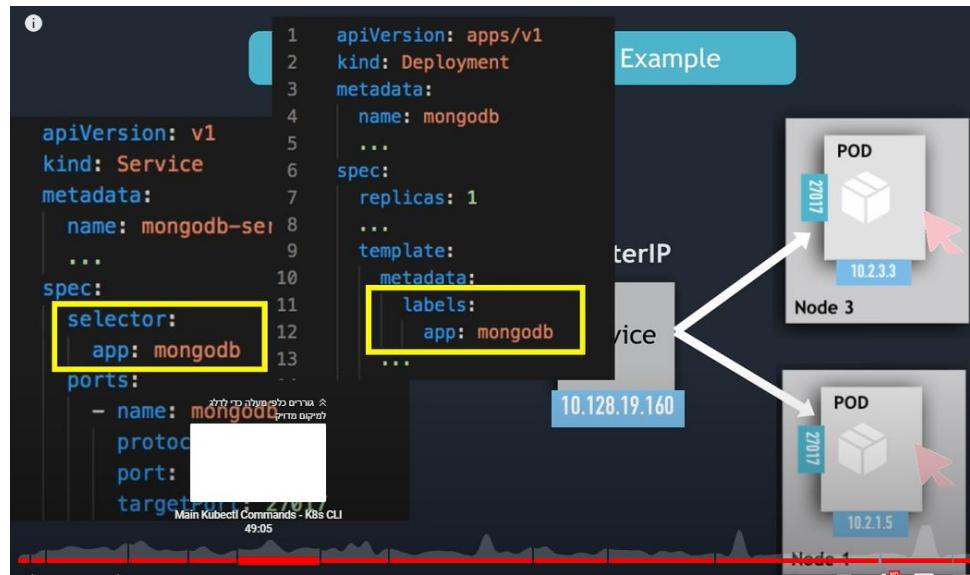
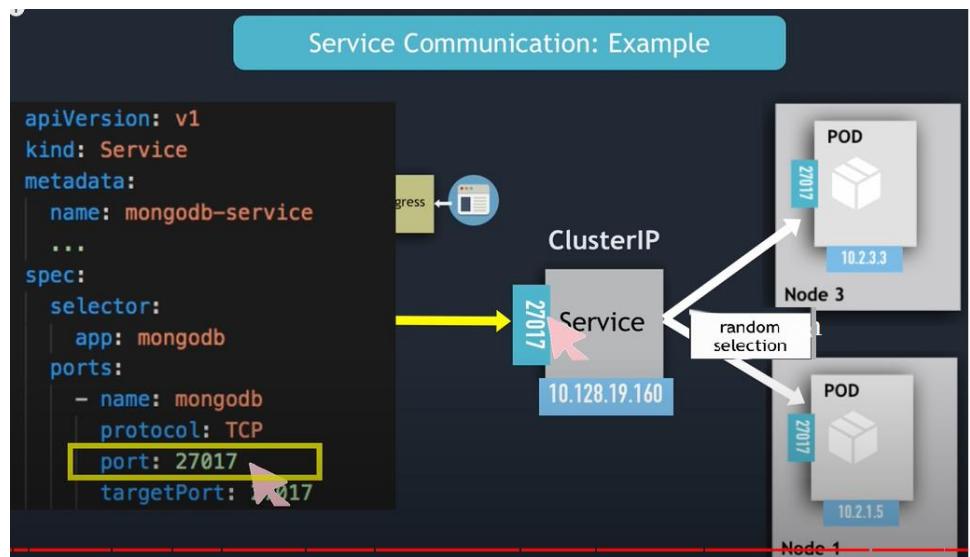
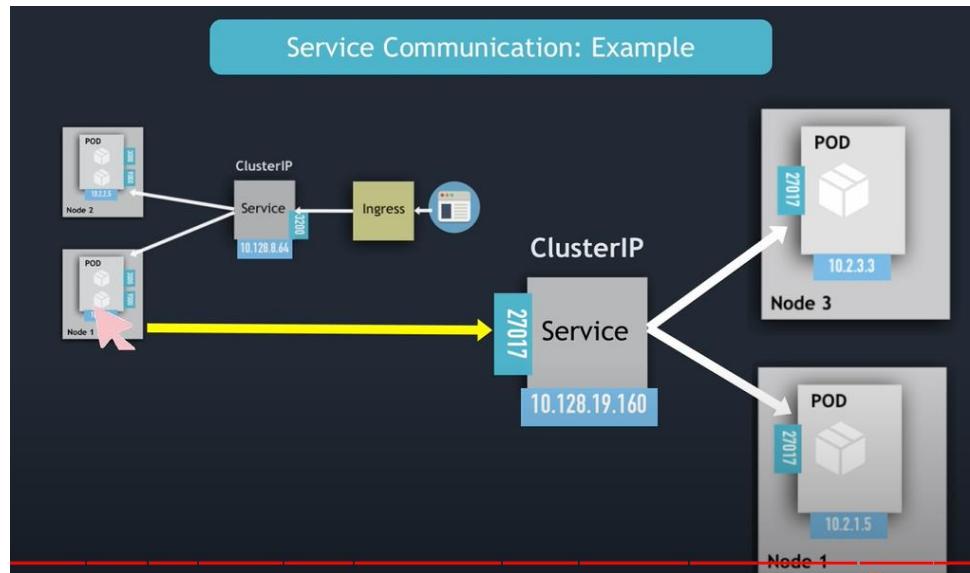


Which port to forward request to?

Pod with multiple ports

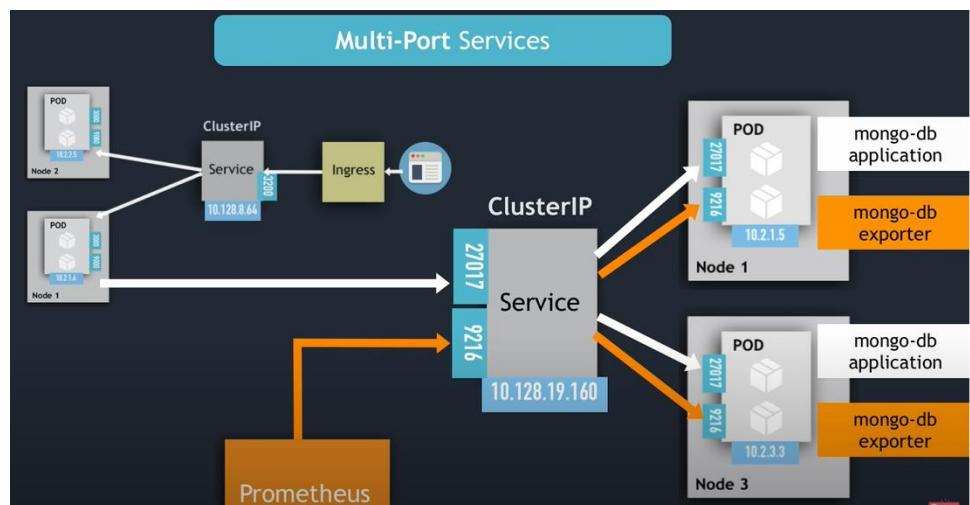
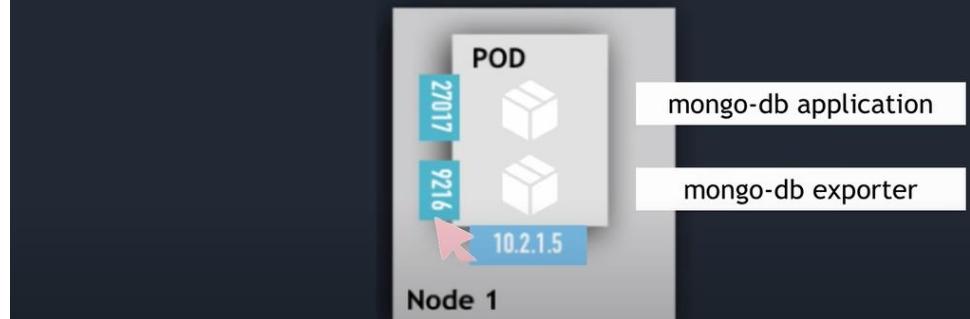




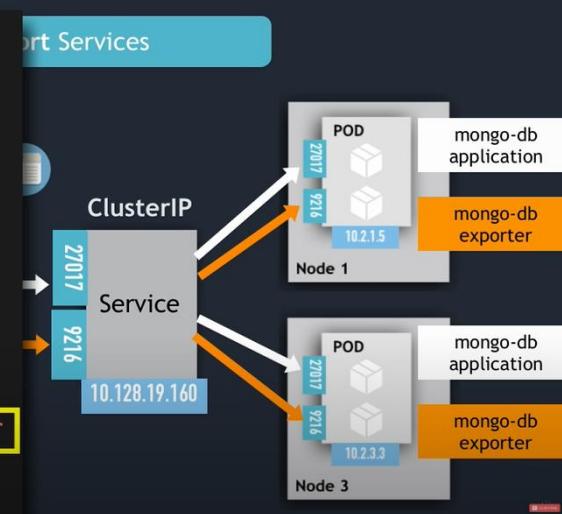


Multi-Port Services

- ▶ Second container running for monitoring metrics



```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: mongodb-service
5 ...
6 spec:
7   selector:
8     app: mongodb
9   ports:
10    - name: mongodb
11      protocol: TCP
12      port: 27017
13      targetPort: 27017
14    - name: mongodb-exporter
15      protocol: TCP
16      port: 9216
17      targetPort: 9216
```



Headless Service

Use case:

1. Client need to connect directly to the pod (DB worker pod)
2. DB Worker pods to master pod (for synchronization)
3. Master pod to perform writecommands to DB

Headless Services

- Client wants to communicate with 1 specific Pod directly
- Pods want to talk directly with specific Pod
- So, not randomly selected

► Use Case: Stateful applications, like databases

- mysql
- mongodb
- elasticsearch

Headless Services

- Client wants to communicate with 1 specific Pod directly
- Pods want to talk directly with specific Pod
- So, not randomly selected

► Use Case: Stateful applications, like

✗ Pod replicas are not identical

Only Master is allowed to write to DB

Headless Services

- ▶ Client wants to communicate with 1 specific Pod directly

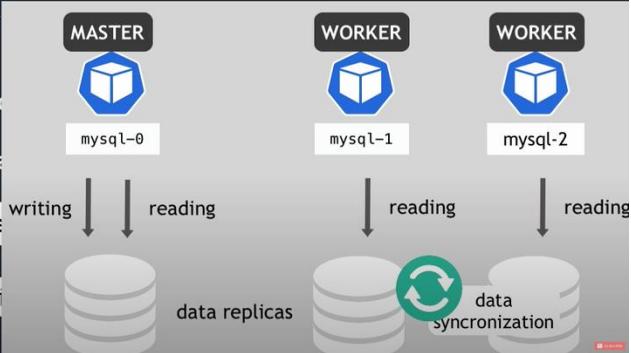
- ▶ Pods want to talk directly with specific Pod

- ▶ So, not randomly selected

- ▶ Use Case: Stateful applications

✗ Pod replicas are not identical

Only Master is allowed to write



Headless Services

Client needs to figure out IP addresses of each Pod

Option 1 - API call to K8s API Server ?

- ✗ makes app too tied to K8s API
- ✗ inefficient

Option 2 - DNS Lookup💡

- ▶ DNS Lookup for Service - returns single IP address (ClusterIP)

Headless Services

Client needs to figure out IP addresses

Option 1 - API call to K8s API Server

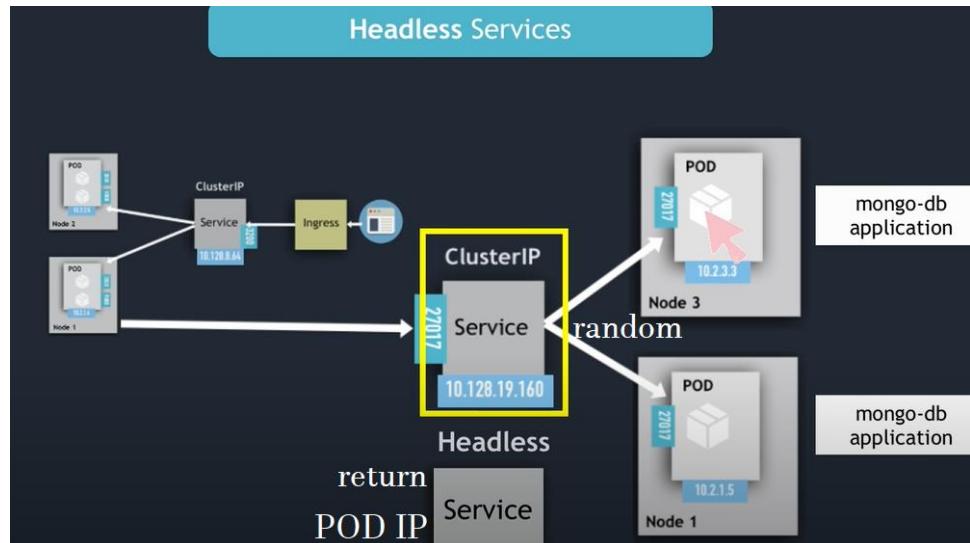
- ✗ makes app too tied to K8s API
- ✗ inefficient

Option 2 - DNS Lookup💡

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: mongodb-service-headless
5  spec:
6    clusterIP: None
7    selector:
8      app: mongodb
9    ports:
10   - protocol: TCP
11     port: 27017
12     targetPort: 27017
```

- ▶ DNS Lookup for Service - returns single IP address (ClusterIP)

- ▶ Set ClusterIP to "None" - returns Pod IP address instead ✓

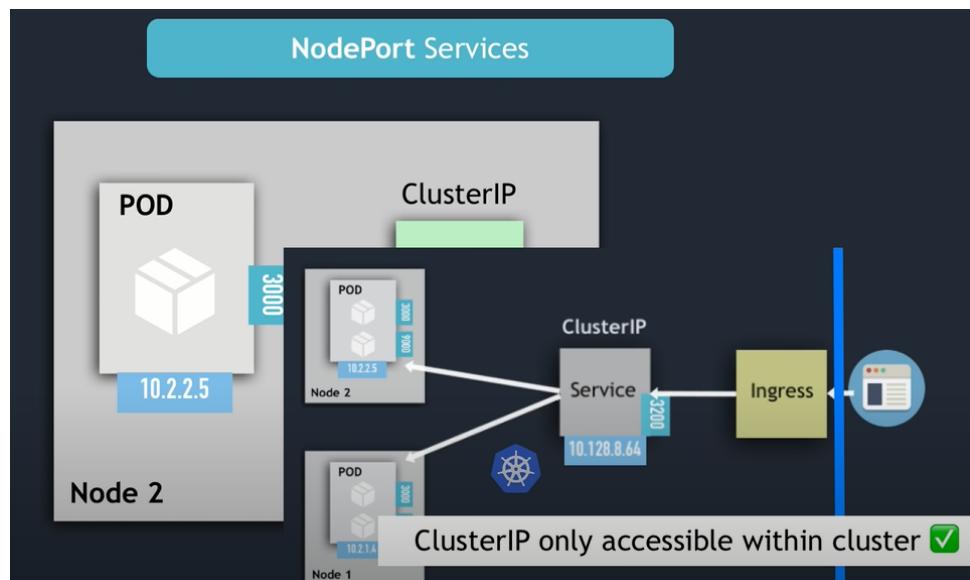


3 Service type attributes

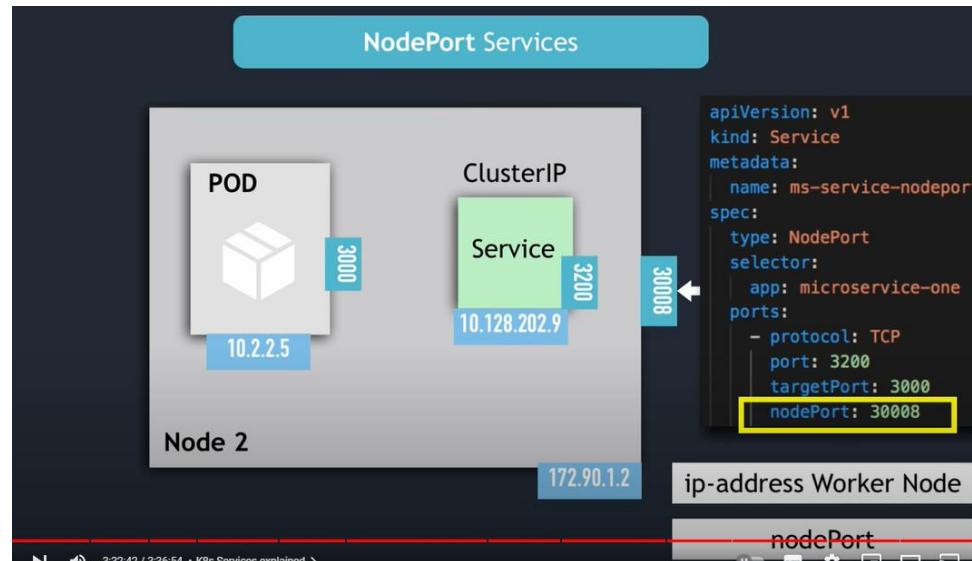
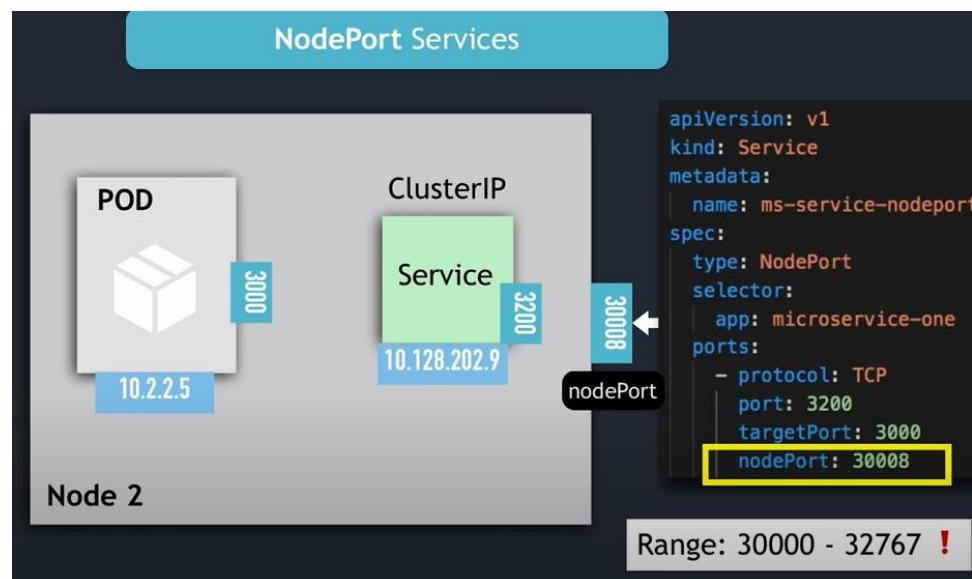
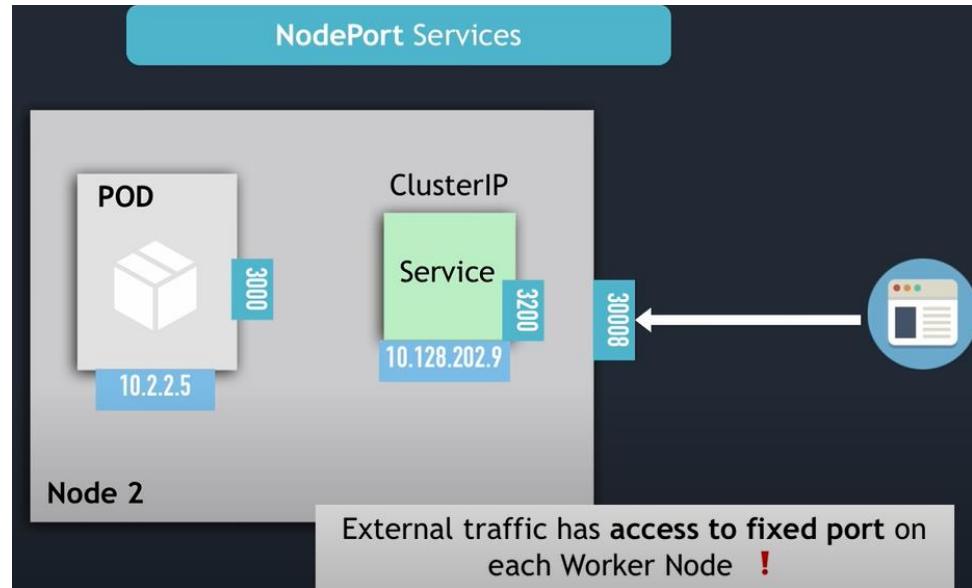
ClusterIP	NodePort	LoadBalancer
<pre>apiVersion: v1 kind: Service metadata: name: my-service spec: type: ClusterIP</pre>	<pre>apiVersion: v1 kind: Service metadata: name: my-service spec: type: NodePort</pre>	<pre>apiVersion: v1 kind: Service metadata: name: my-service spec: type: LoadBalancer</pre>

- ▶ DEFAULT, type not needed!
- ▶ internal service

ClusterIP

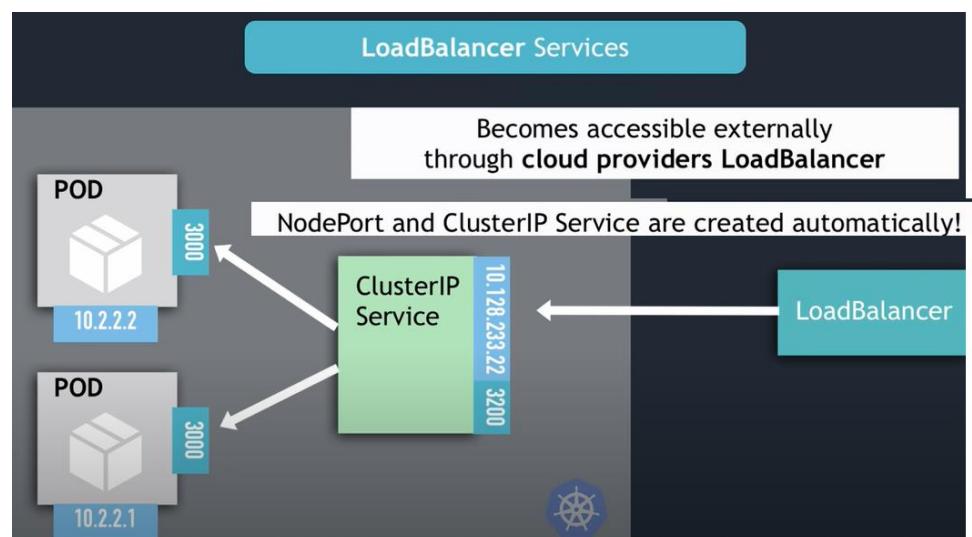
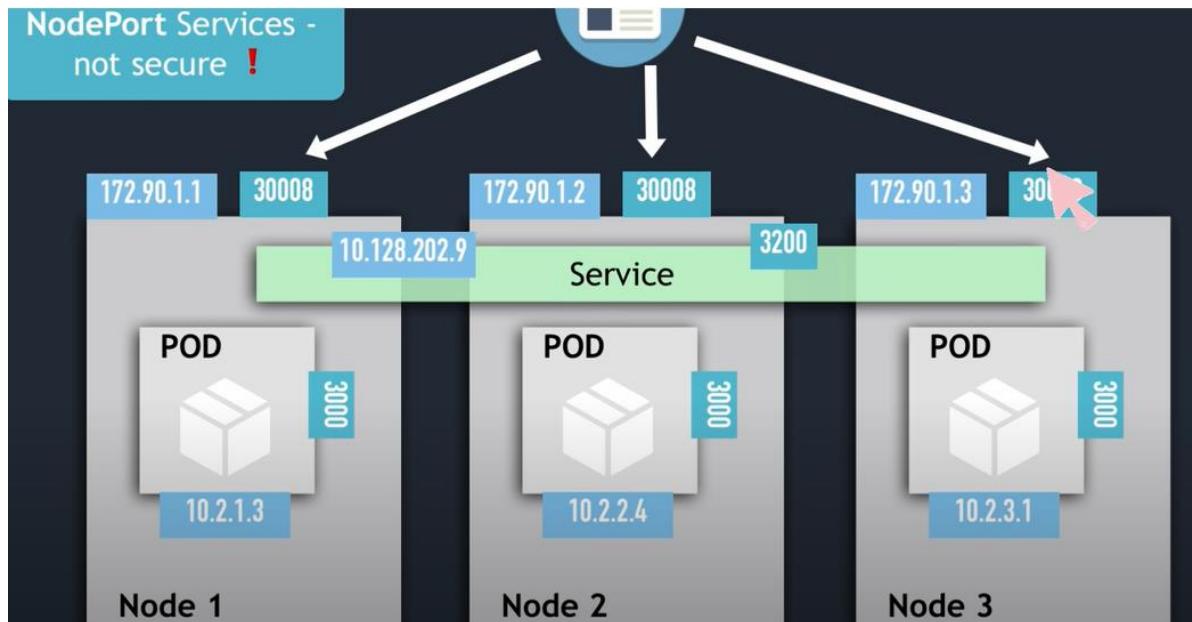


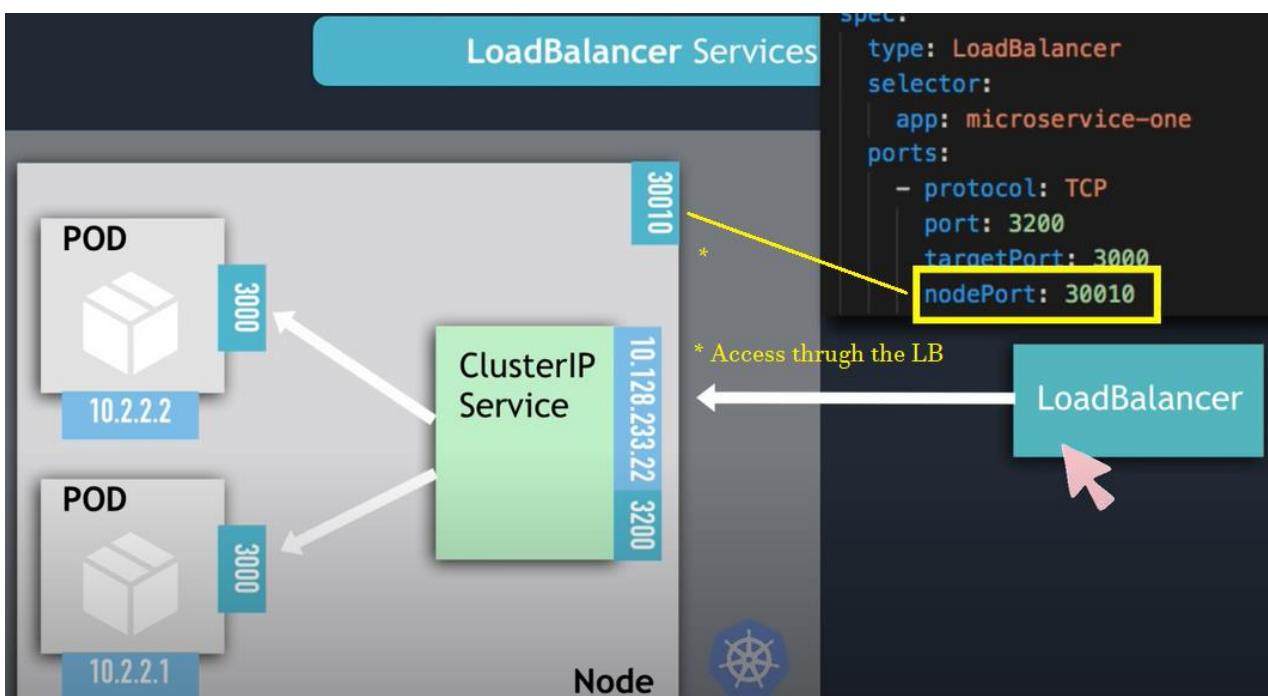
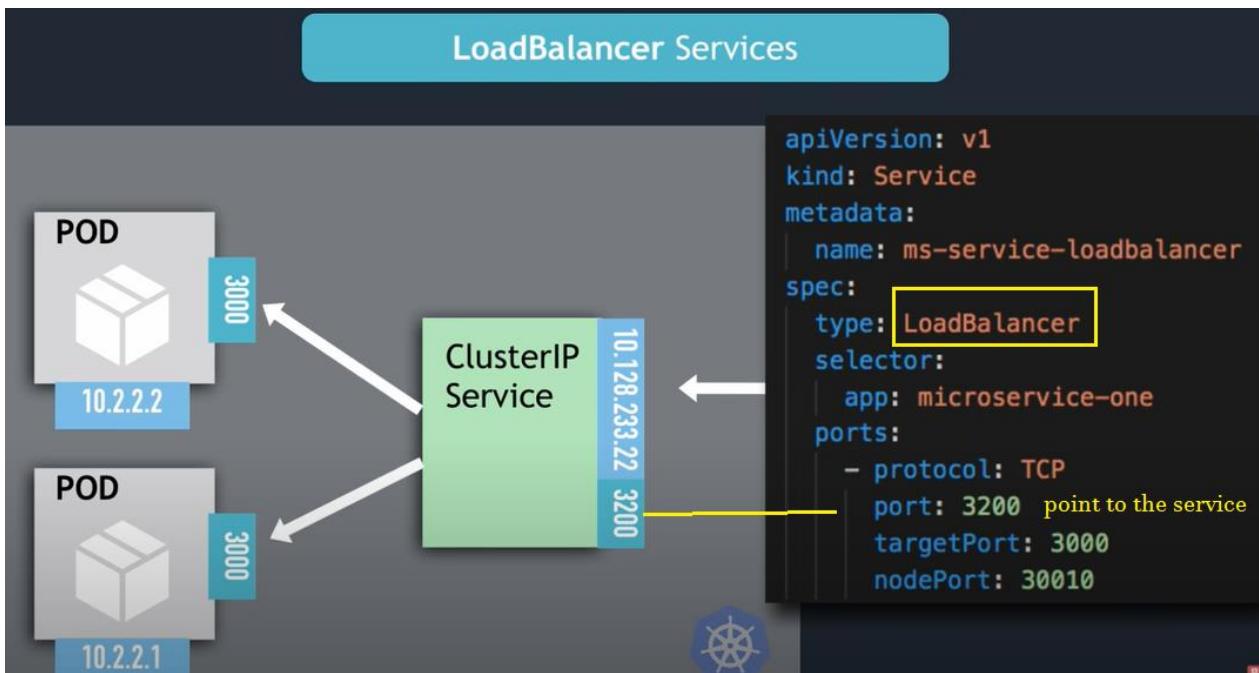
NodePort



NodePort Services						
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	
kubernetes	ClusterIP	10.128.0.1	<none>	443/TCP	20m	
mongodb-service	ClusterIP	10.128.204.105	<none>	27017/TCP	10m	
mongodb-service-headless	ClusterIP	None	<none>	27017/TCP	2m8s	
ms-service-nodeport	NodePort	10.128.202.9	<none>	3200:30008/TCP	8s	

cluster-ip:3200 node-ip:30008





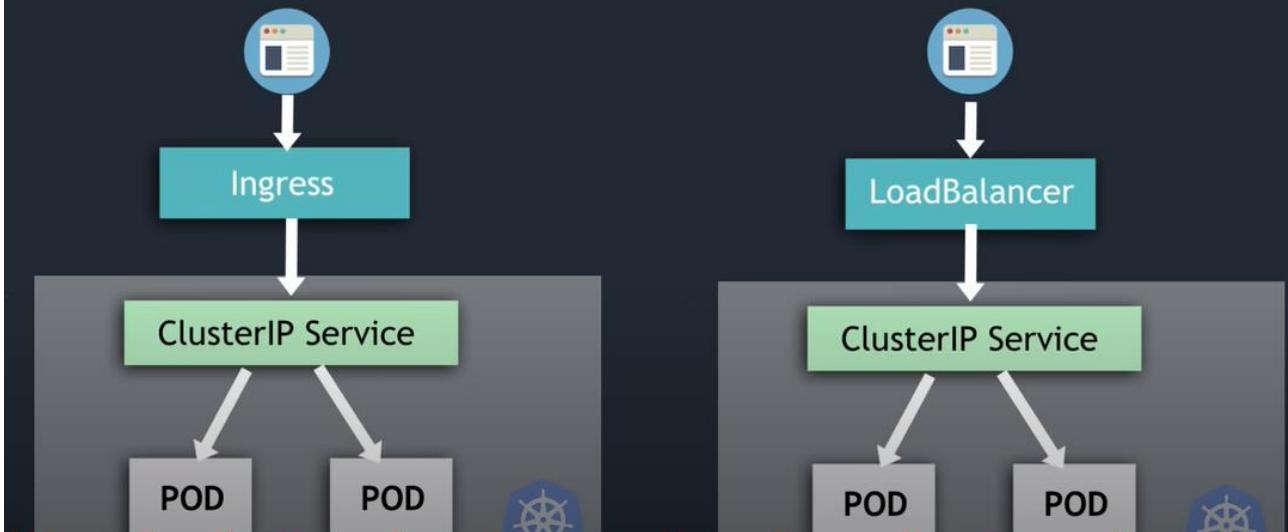
LoadBalancer Services

- ▶ LoadBalancer Service is an extension of NodePort Service
- ▶ NodePort Service is an extension of ClusterIP Service

```
[k8s-services]$ kubectl get svc
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)
kubernetes     ClusterIP 10.128.0.1  <none>        443/TCP
mongodb-service ClusterIP 10.128.204.105  <none>        27017/TCP
mongodb-service-headless ClusterIP None    <none>        27017/TCP
ms-service-loadbalancer LoadBalancer 10.128.233.22  172.104.255.5  3200:30010/TCP
ms-service-nodeport NodePort   10.128.202.9  <none>        3200:30008/TCP
```

Wrap-Up

- ▶ Configure Ingress or LoadBalancer for production environments



HELM

HPA – scaling Horizontally

In Kubernetes, you can use the Horizontal Pod Autoscaler (HPA) to automatically scale the number of replicas of a deployment or a statefulset based on metrics such as CPU utilization or memory usage.

To set up an HPA, you need to define a metric to monitor and set a target value for the metric. For example, you can configure the HPA to monitor CPU utilization and set a target value of 70%.

Once the HPA is set up, Kubernetes will automatically monitor the metric and adjust the number of replicas accordingly. If the CPU utilization goes above the target value, Kubernetes will add more replicas, and if it goes below the target value, Kubernetes will reduce the number of replicas.

Here is an example manifest for setting up an HPA:

[yaml](#) [Copy code](#)

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: my-app-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: my-app-deployment
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
    targetAverageUtilization: 70
```

In this example, the HPA is configured to monitor CPU utilization and adjust the number of replicas of a deployment named **my-app-deployment**. The minimum number of replicas is set to 2, and the maximum number of replicas is set to 10. The target CPU utilization is set to 70%.

With this configuration, Kubernetes will automatically adjust the number of replicas of the deployment based on the CPU utilization, ensuring that the application can handle changes in workload automatically.