

# Kubernetes

An Introduction



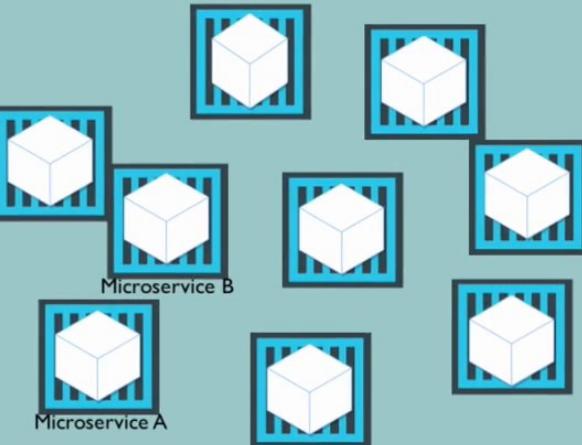


## Intro to K8s:

- ▶ What is Kubernetes?
- ▶ K8s Architecture

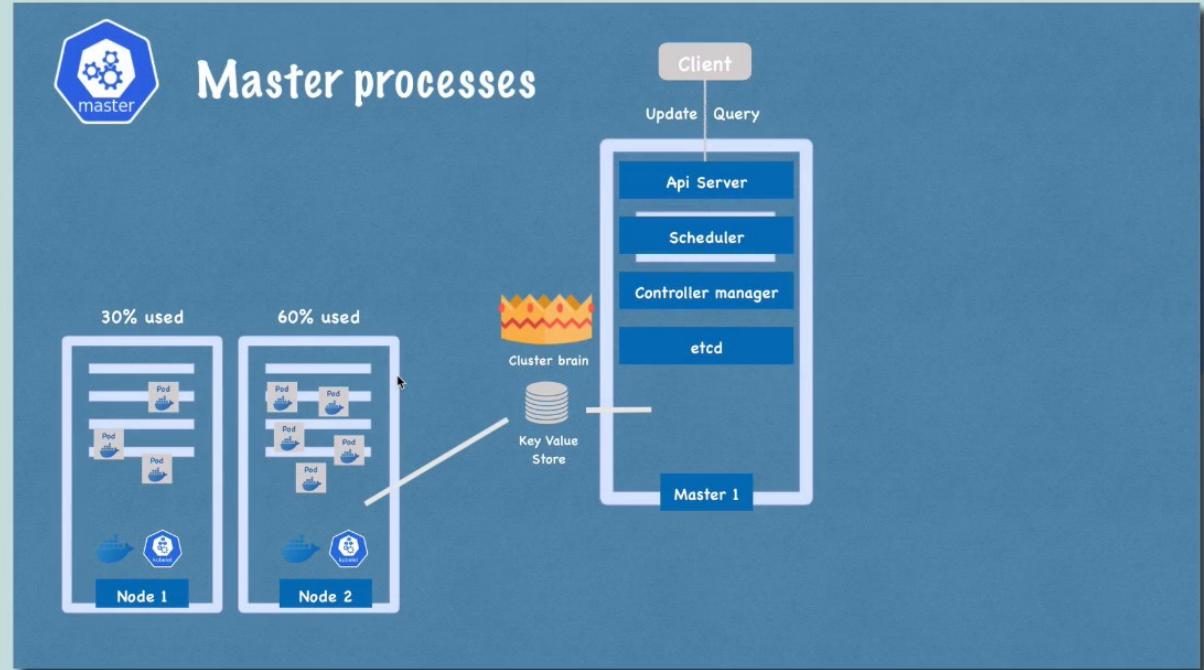
### The need for a container orchestration tool

- › Trend from **Monolith** to **Microservices**
- › Increased usage of containers



## Intro to K8s:

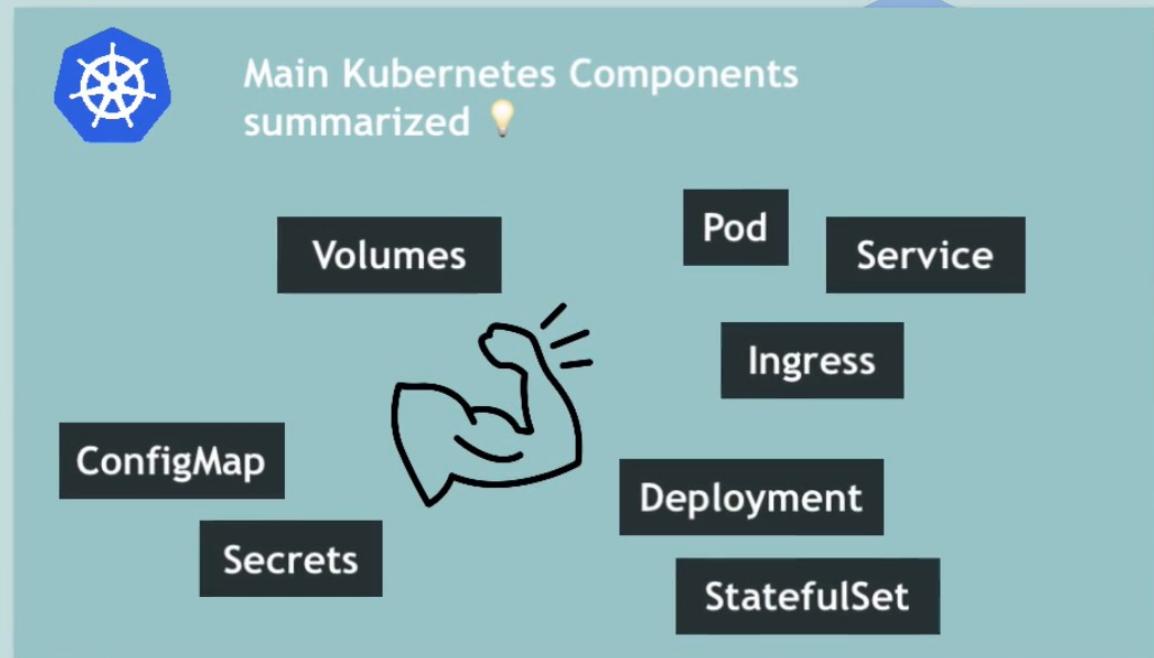
- ▶ What is Kubernetes?
- ▶ K8s Architecture





## Intro to K8s:

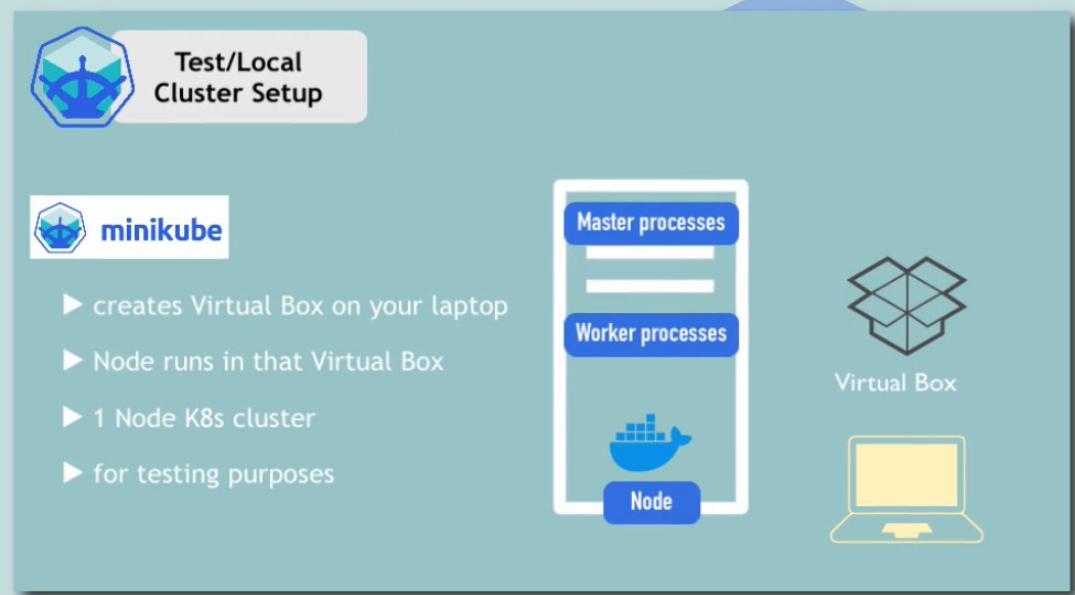
- ▶ What is Kubernetes?
- ▶ K8s Architecture
- ▶ Main K8s Components





## Intro to K8s:

- ▶ What is Kubernetes?
- ▶ K8s Architecture
- ▶ Main K8s Components
- ▶ Minikube and Kubectl - Local Setup





## Intro to K8s:

- ▶ What is Kubernetes?
- ▶ K8s Architecture
- ▶ Main K8s Components
- ▶ Minikube and Kubectl - Local Setup
- ▶ Main Kubectl Commands - K8s CLI





## Intro to K8s:

- ▶ What is Kubernetes?
- ▶ K8s Architecture
- ▶ Main K8s Components
- ▶ Minikube and Kubectl - Local S
- ▶ Main Kubectl Commands - K8s

**Basic kubectl commands**

**Create and debug Pods in a minikube cluster**

**CRUD commands**

- [Create deployment](#)      `kubectl create deployment [name]`
- [Edit deployment](#)      `kubectl edit deployment [name]`
- [Delete deployment](#)      `kubectl delete deployment [name]`

**Status of different K8s components**

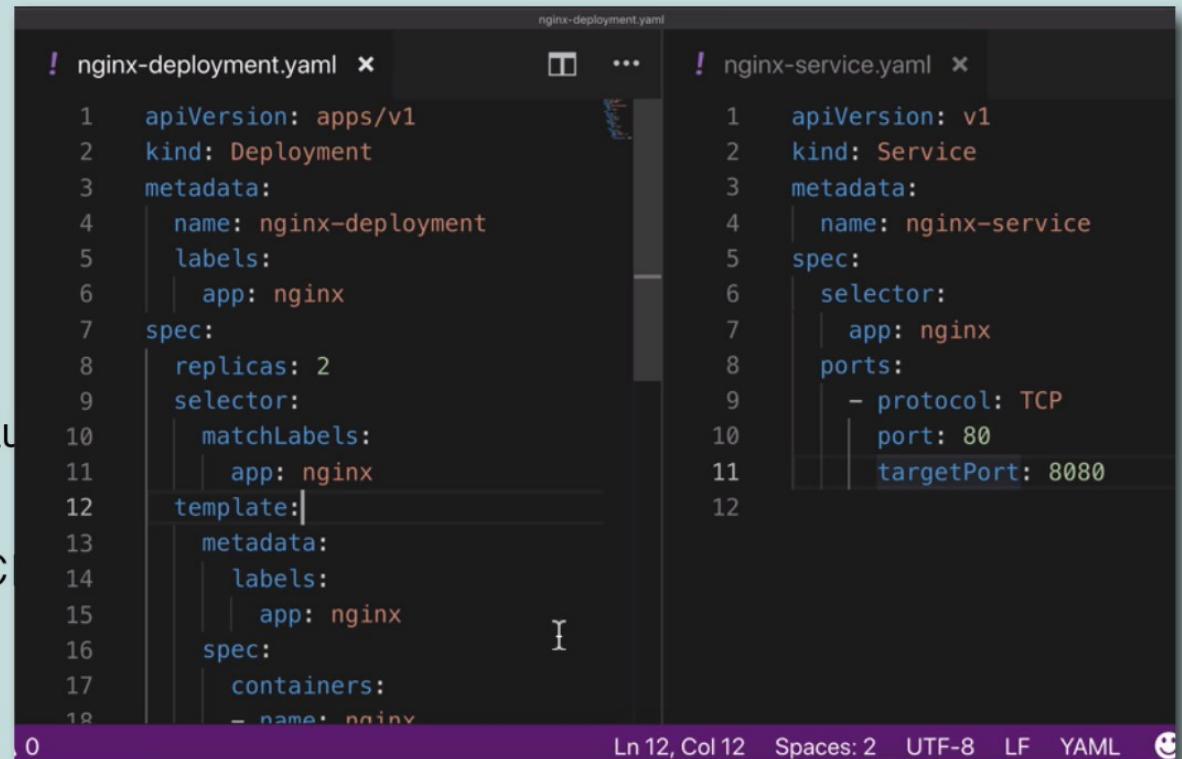
- `kubectl get nodes | pod | services | replicaset | deployment`

**Debugging pods**

- [Log to console](#)      `kubectl logs [pod name]`
- [Get Interactive Terminal](#)      `kubectl exec -it [pod name] -- bin/bash`

# Intro to K8s:

- ▶ What is Kubernetes?
- ▶ K8s Architecture
- ▶ Main K8s Components
- ▶ Minikube and Kubectl - Local Setup
- ▶ Main Kubectl Commands - K8s C
- ▶ K8s YAML Configuration File

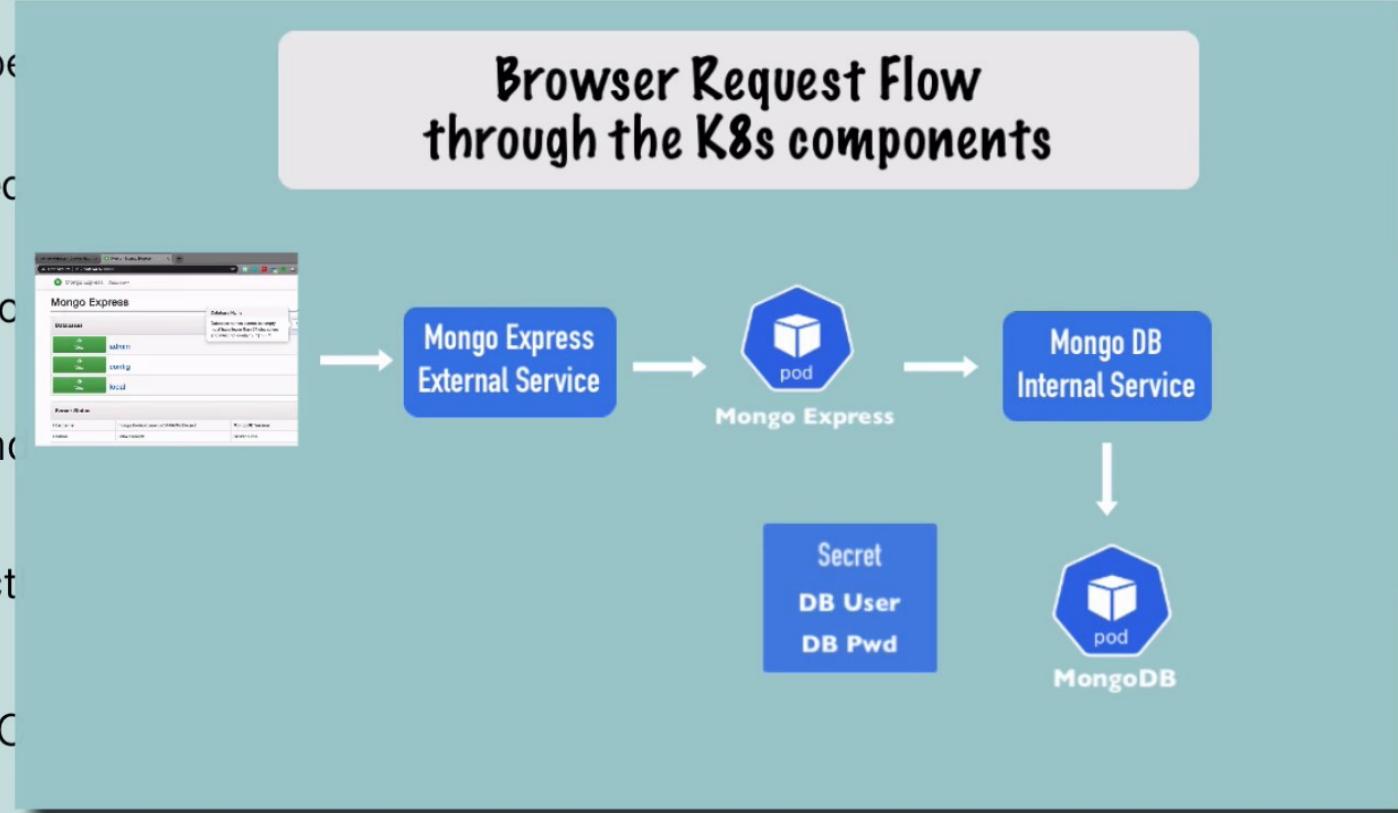


```
nginx-deployment.yaml
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: nginx-deployment
5   labels:
6     app: nginx
7 spec:
8   replicas: 2
9   selector:
10    matchLabels:
11      app: nginx
12   template:
13     metadata:
14       labels:
15         app: nginx
16     spec:
17       containers:
18         - name: nginx
0
Ln 12, Col 12  Spaces: 2  UTF-8  LF  YAML  ⚙
```

```
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
Ln 12, Col 12  Spaces: 2  UTF-8  LF  YAML  ⚙
```

## Intro to K8s:

- ▶ What is Kub...
- ▶ K8s Architec...
- ▶ Main K8s Co...
- ▶ Minikube and...
- ▶ Main Kubectl...
- ▶ K8s YAML Co...
- ▶ Hands-On Demo

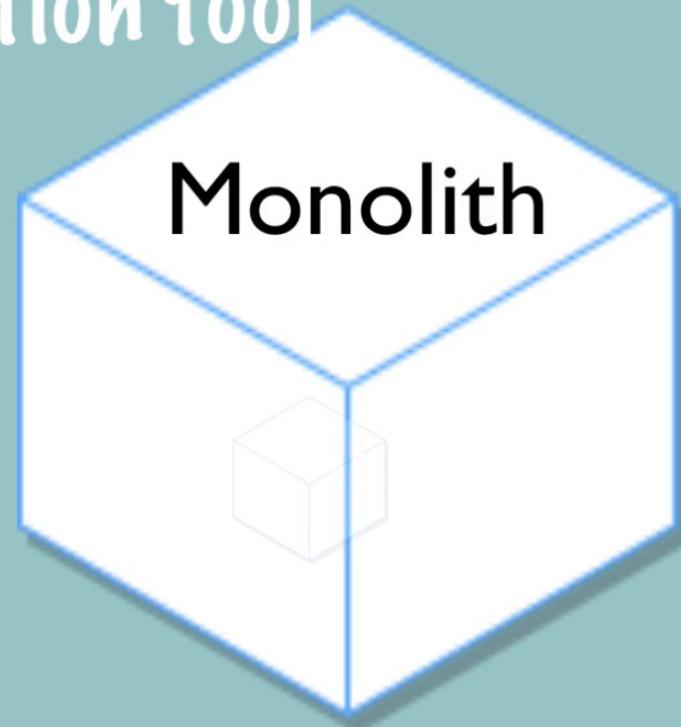


What **problems** does Kubernetes solve?

What are the **tasks** of an orchestration tool?

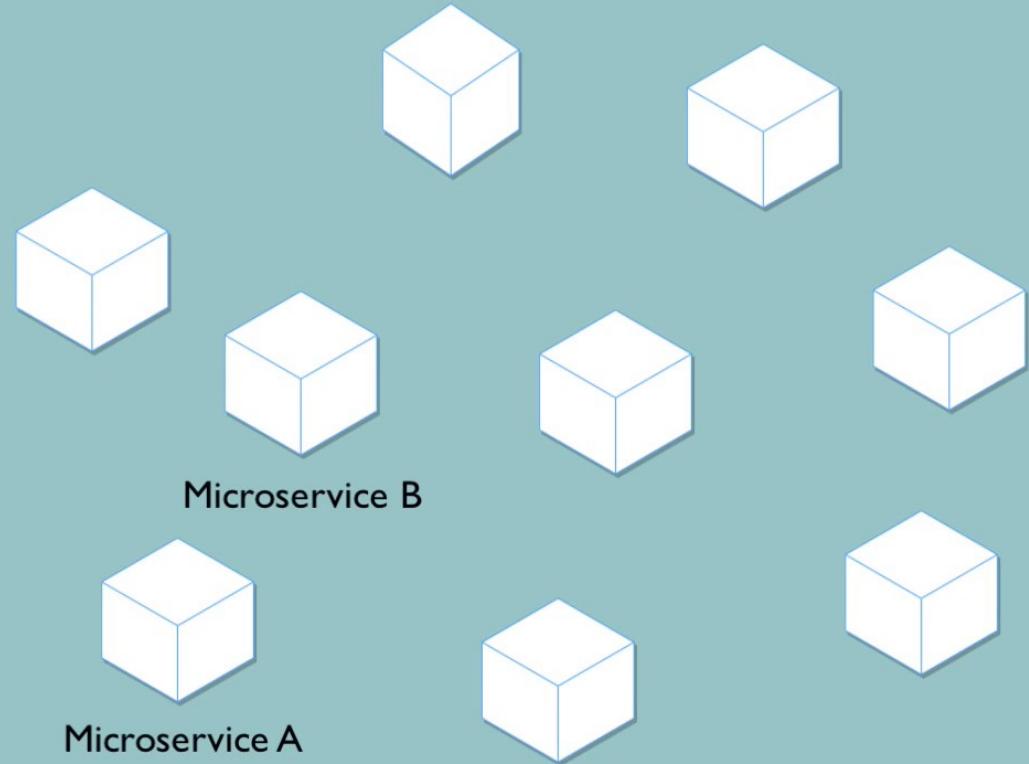
# The need for a container orchestration tool

> Trend from **Monolith**



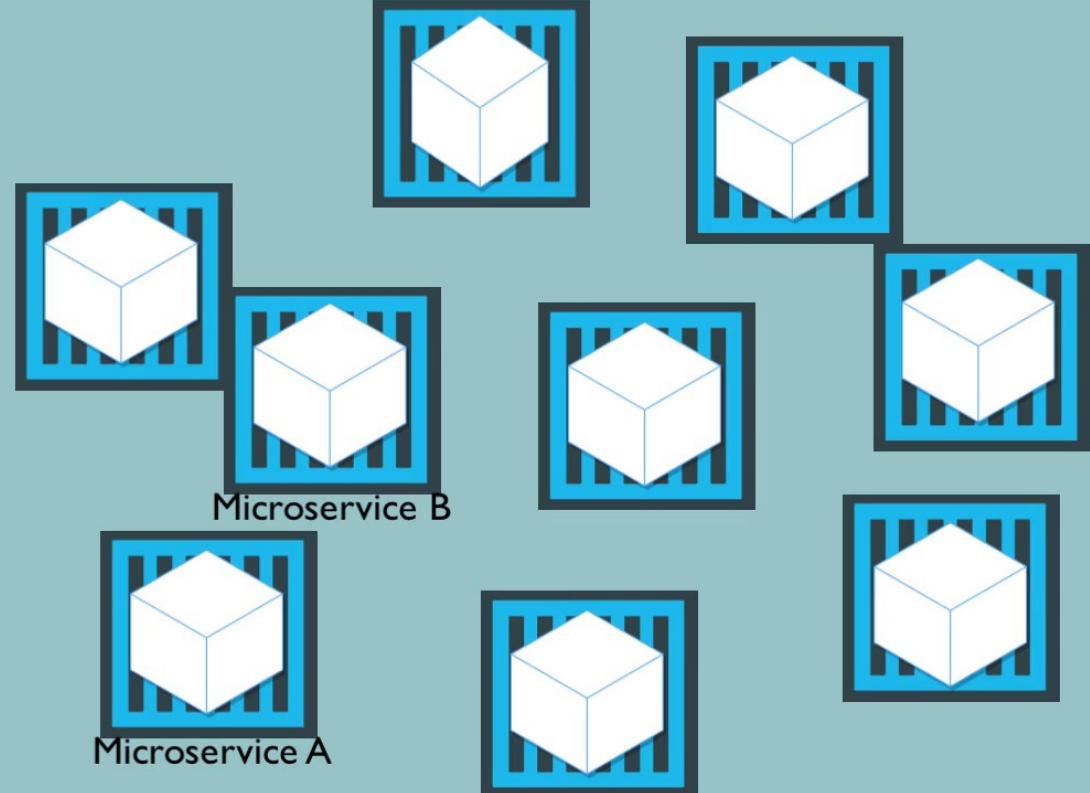
# The need for a container orchestration tool

> Trend from **Monolith** to **Microservices**



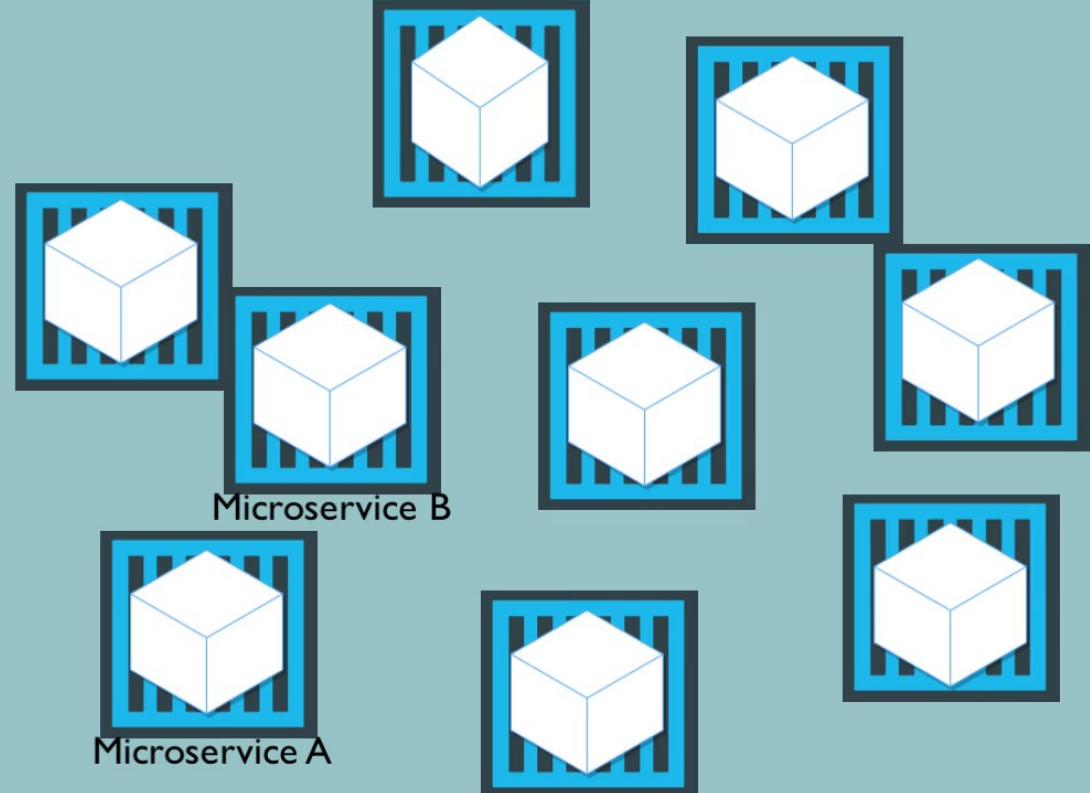
# The need for a container orchestration tool

- > Trend from **Monolith** to **Microservices**
- > Increased usage of containers



# The need for a container orchestration tool

- > Trend from **Monolith** to **Microservices**
- > Increased usage of containers



# The need for a container orchestration tool

- > Trend from Monolith to Microservices
- > Increased usage of containers
- > Demand for a proper way of managing those hundreds of containers

# What features do orchestration tools offer?

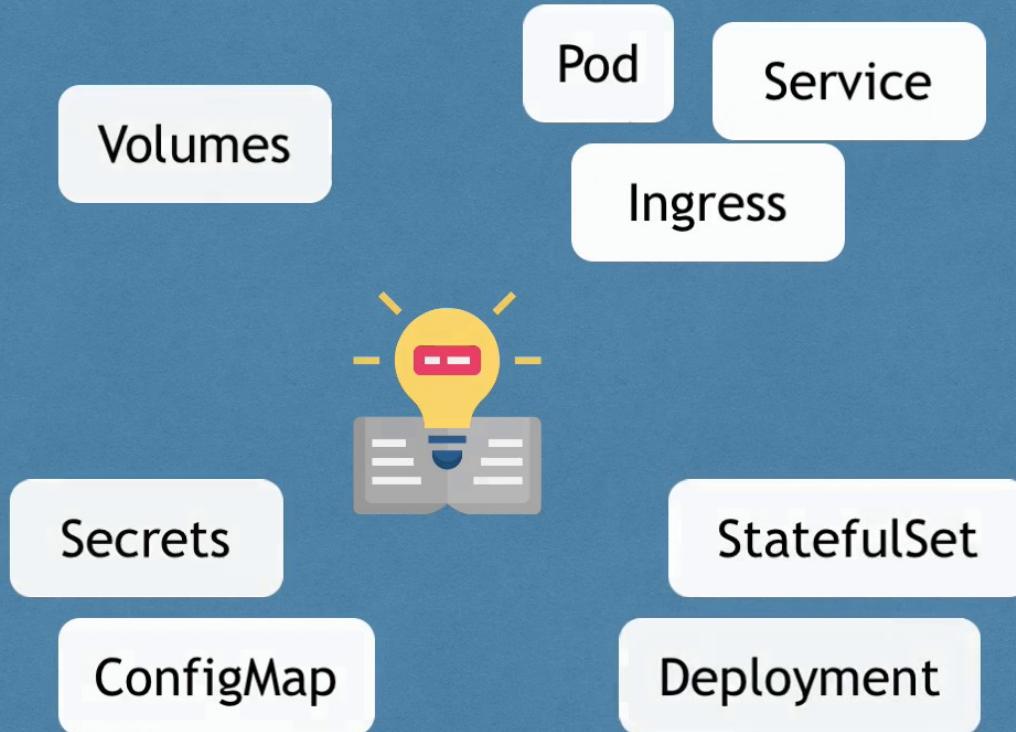


- > **High Availability** or no downtime
- > **Scalability** or high performance
- > **Disaster recovery** - backup and restore



# Kubernetes Components

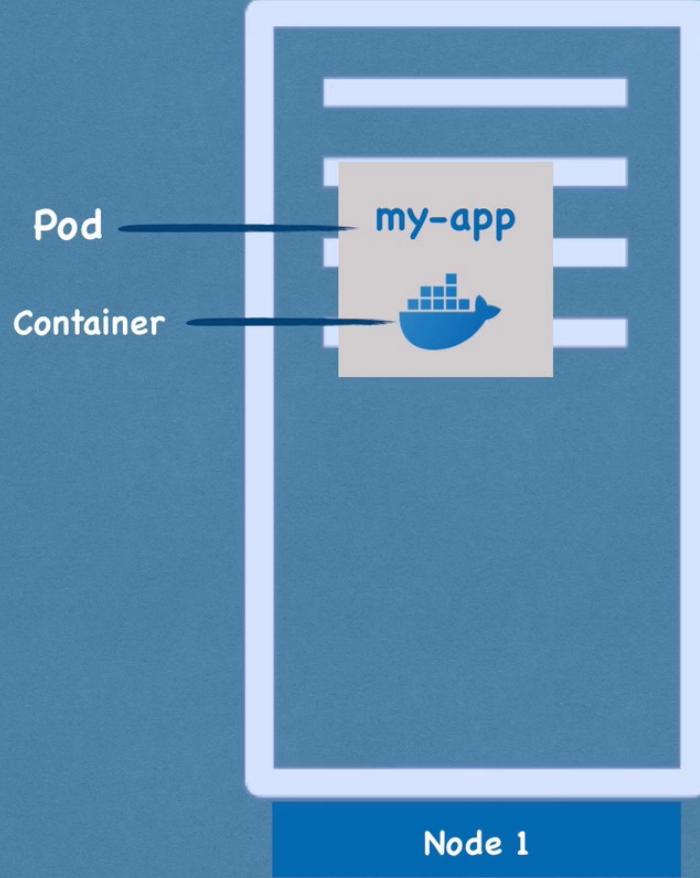
---





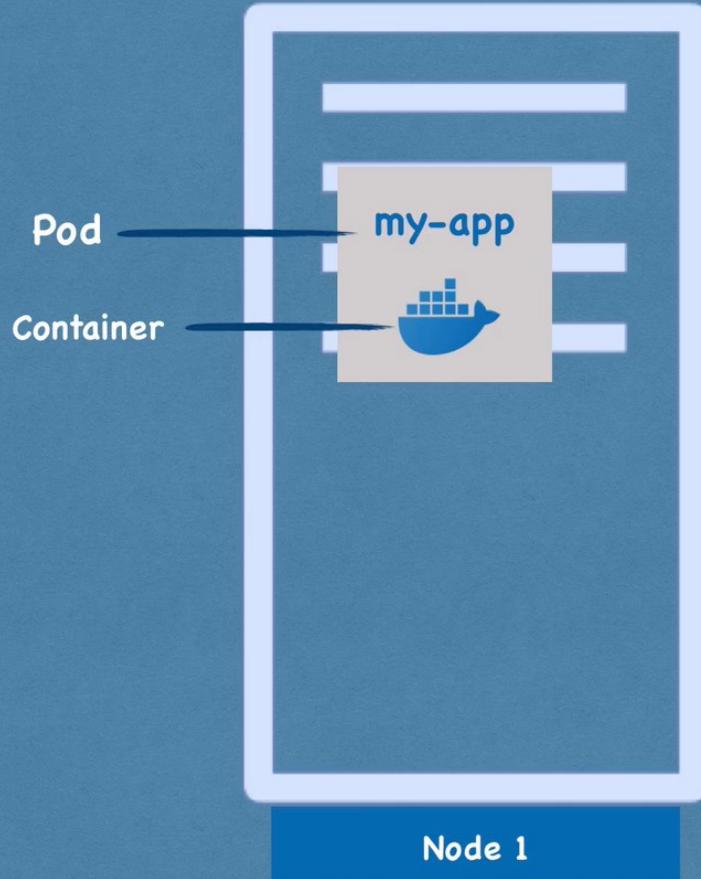
# Node and Pod





## Pod:

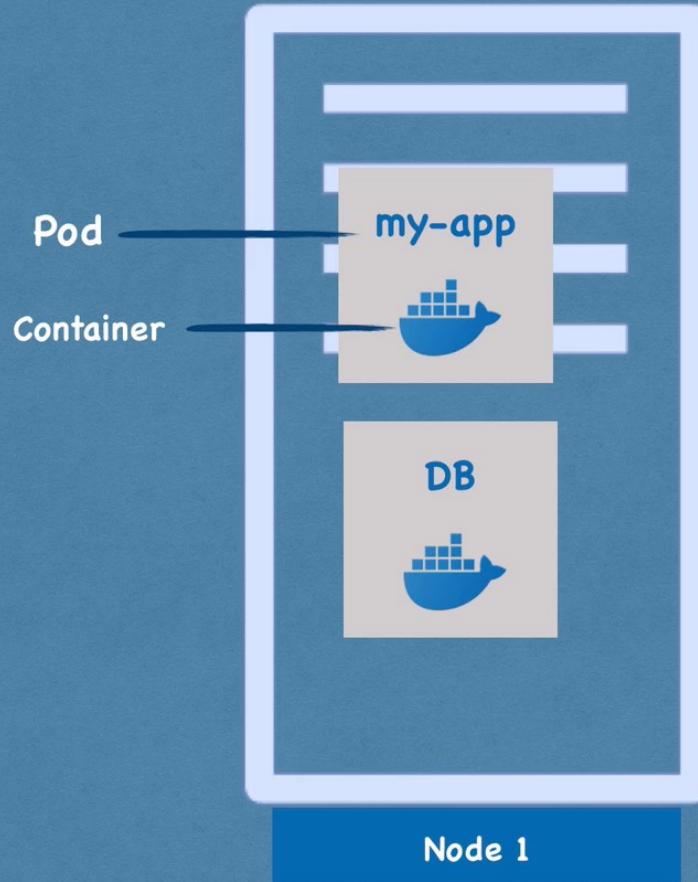
- ▶ Smallest unit of K8s
- ▶ Abstraction over container



## Pod:

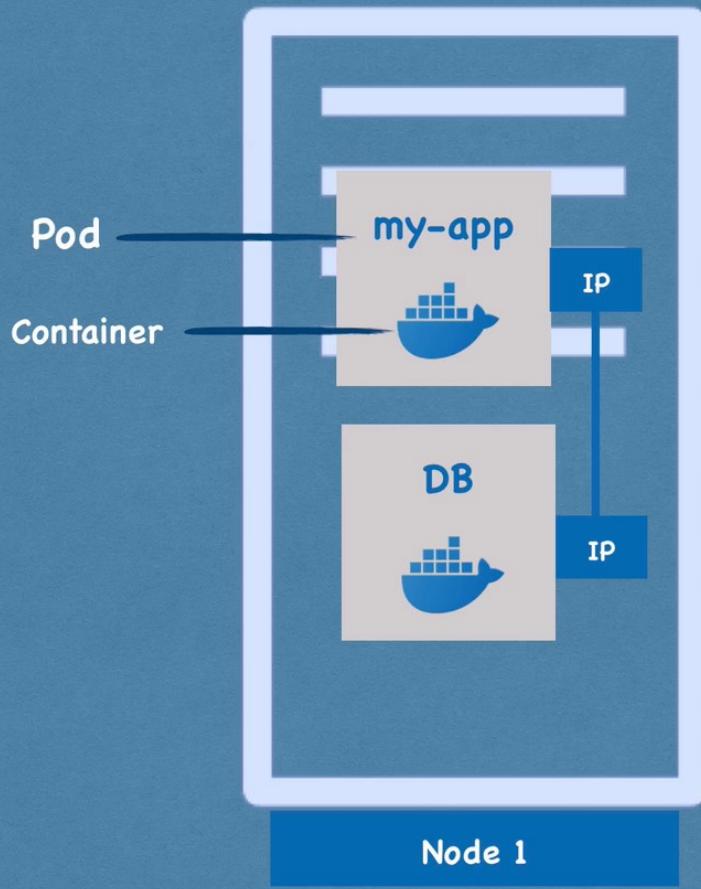
- ▶ Smallest unit of K8s
- ▶ Abstraction over container

You only interact with the  
Kubernetes layer



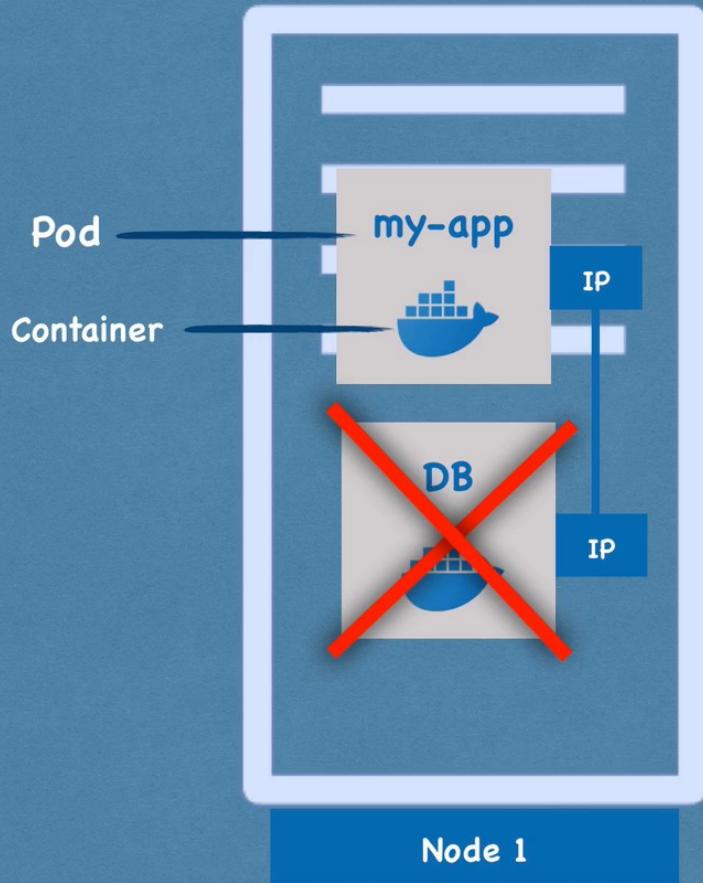
## Pod:

- ▶ Smallest unit of K8s
- ▶ Abstraction over container
- ▶ Usually 1 application per Pod



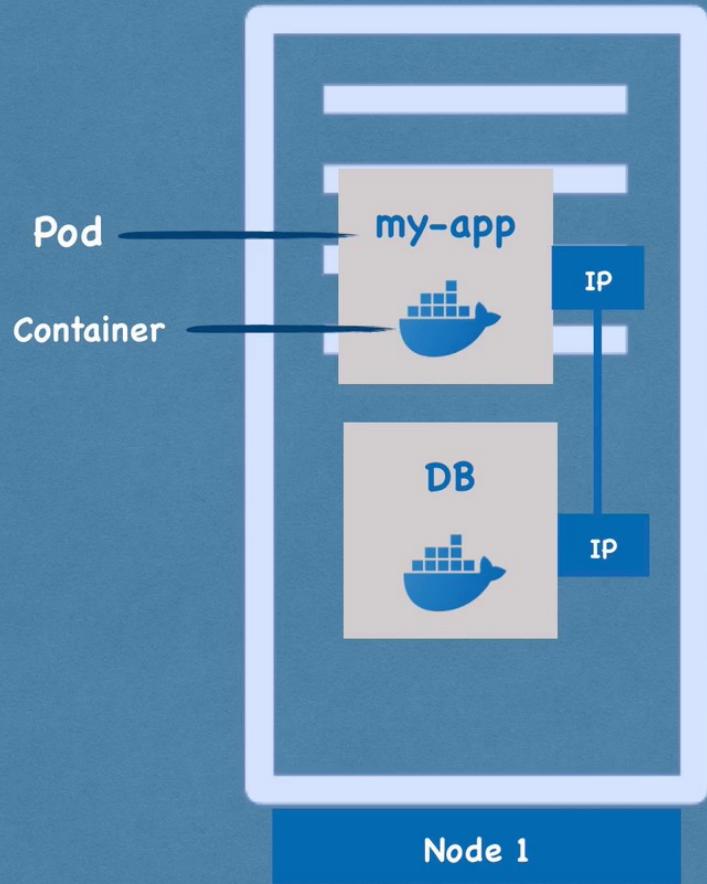
## Pod:

- ▶ Smallest unit of K8s
- ▶ Abstraction over container
- ▶ Usually 1 application per Pod
- ▶ Each Pod gets its own IP address



## Pod:

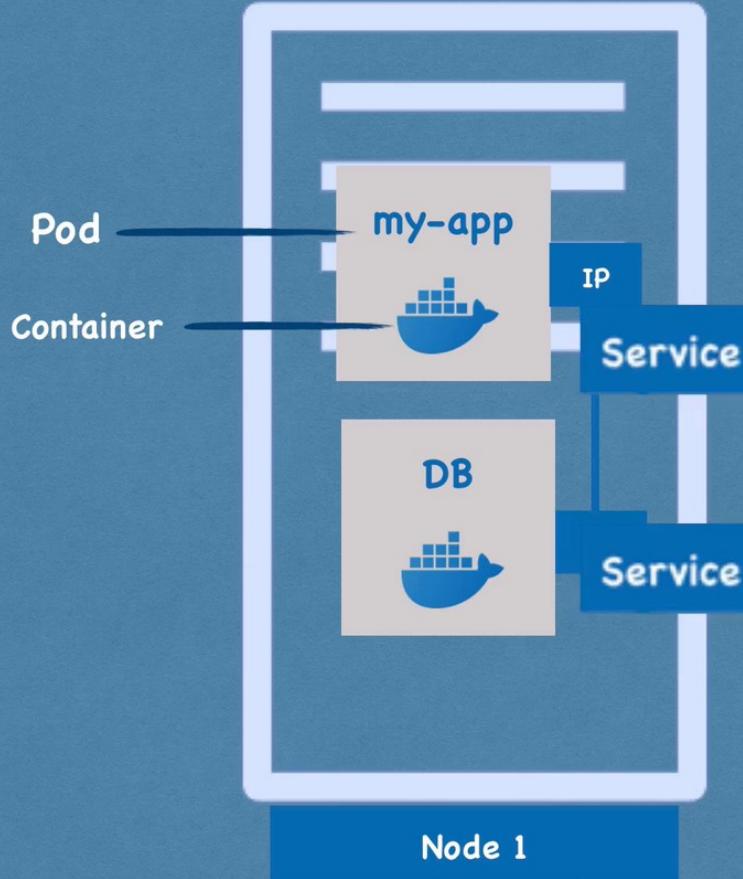
- ▶ Smallest unit of K8s
- ▶ Abstraction over container
- ▶ Usually 1 application per Pod
- ▶ Each Pod gets its own IP address



## Pod:

- ▶ Smallest unit of K8s
- ▶ Abstraction over container
- ▶ Usually 1 application per Pod
- ▶ Each Pod gets its own IP address
- ▶ New IP address on re-creation





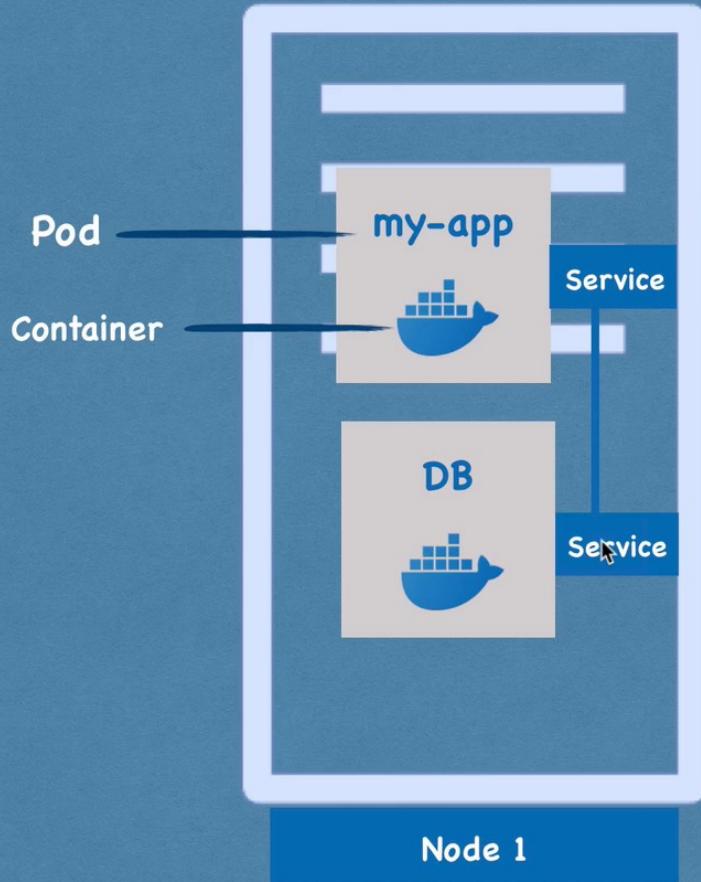
## Pod:

- ▶ Smallest unit of K8s
- ▶ Abstraction over container
- ▶ Usually 1 application per Pod
- ▶ Each Pod gets its own IP address
- ▶ New IP address on re-creation



# Service and Ingress

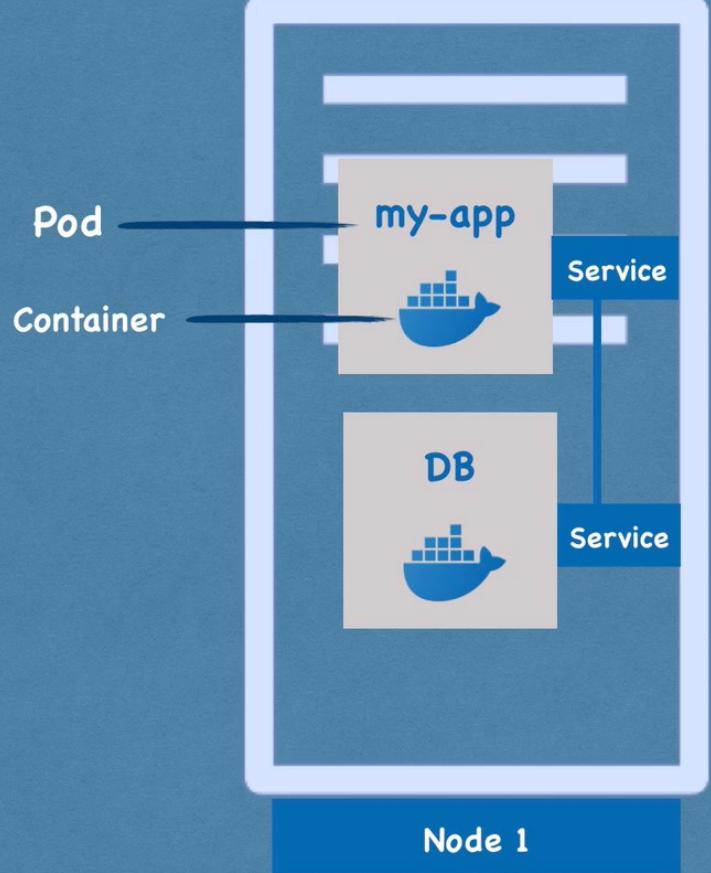




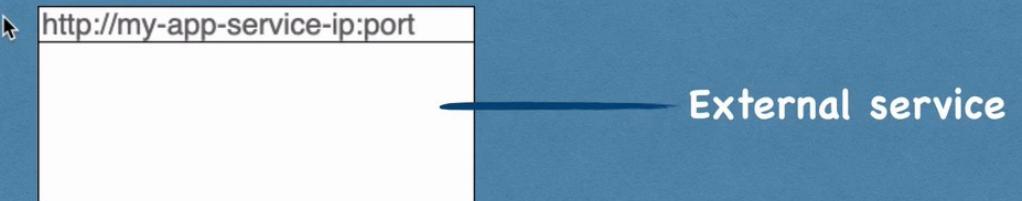
## Service:

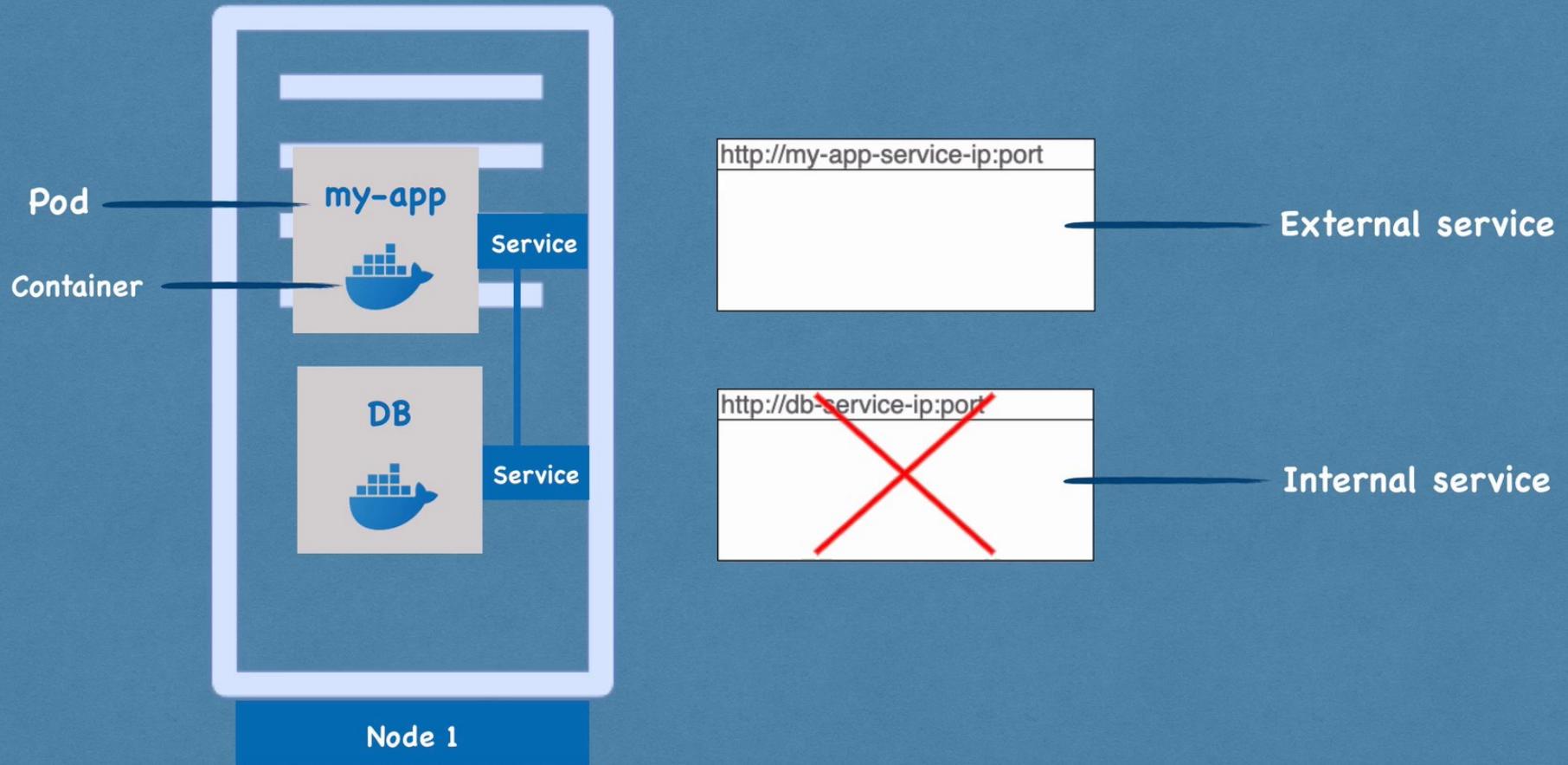
- ▶ permanent IP address
- ▶ lifecycle of Pod and Service  
NOT connected

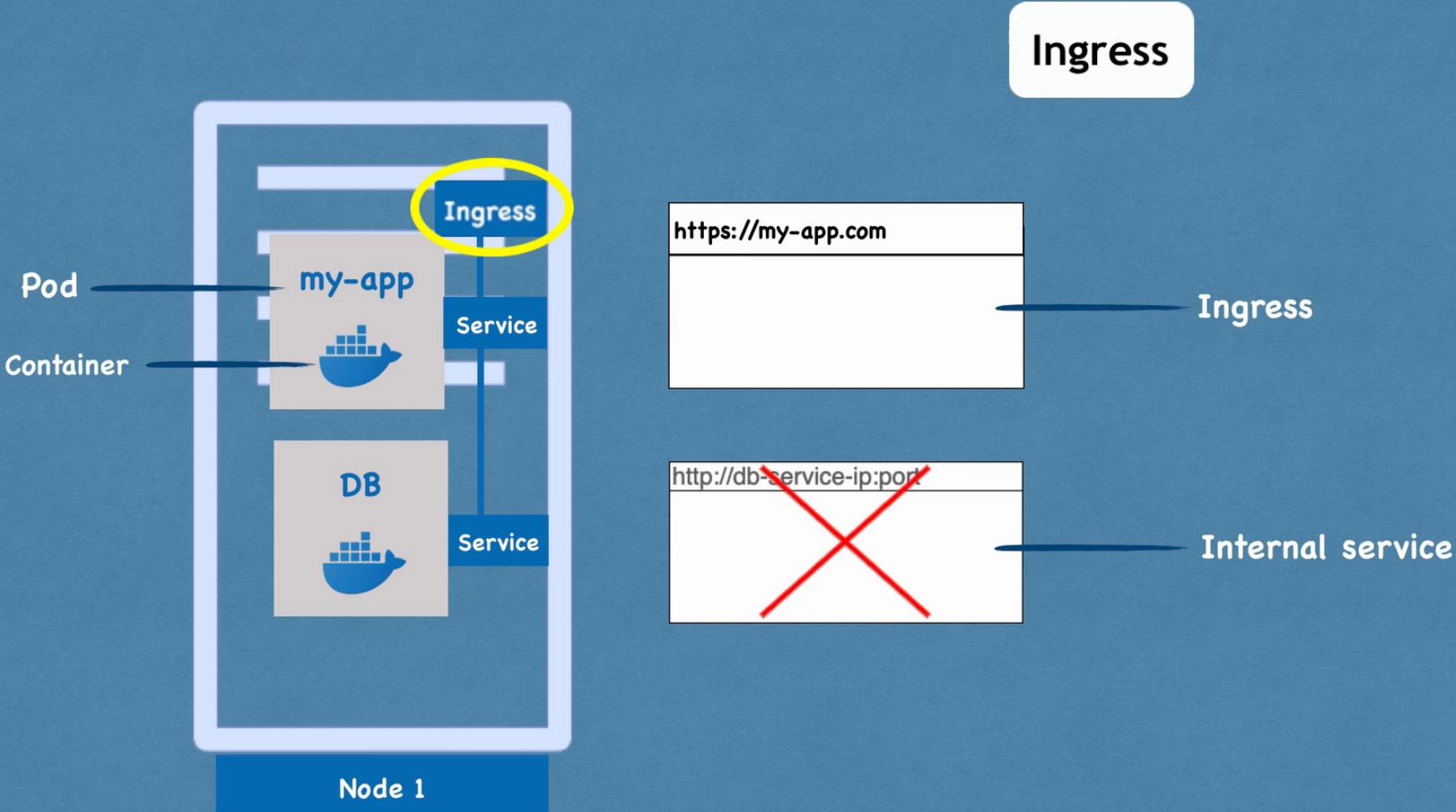




App should be accessible through browser



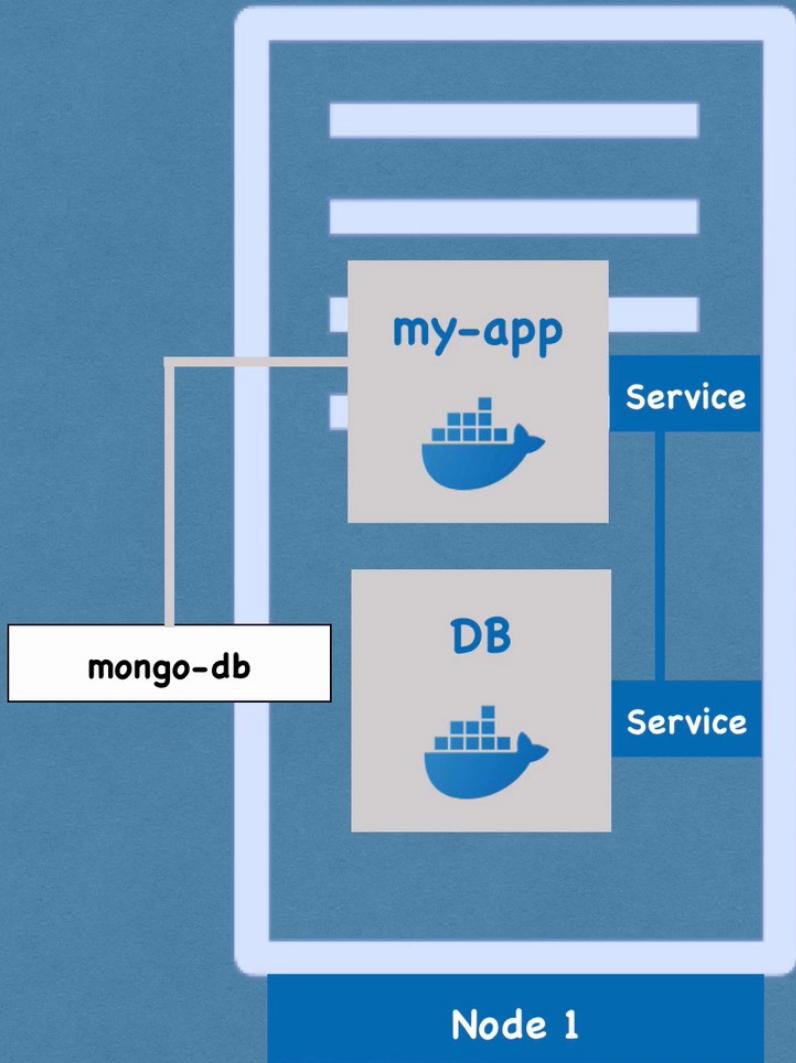






# ConfigMap and Secret





Database URL usually in the  
built application!

Repository



re-build

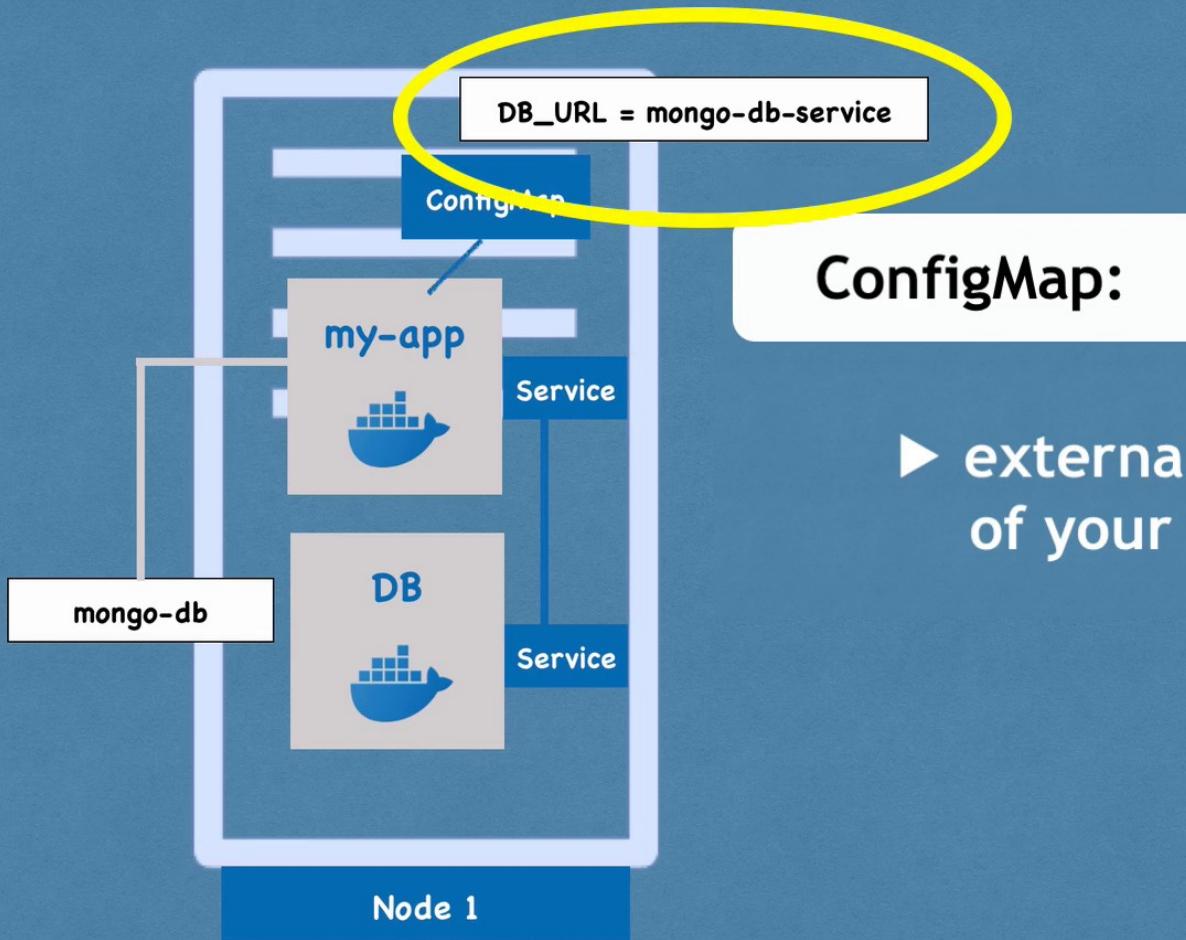


push it to repo

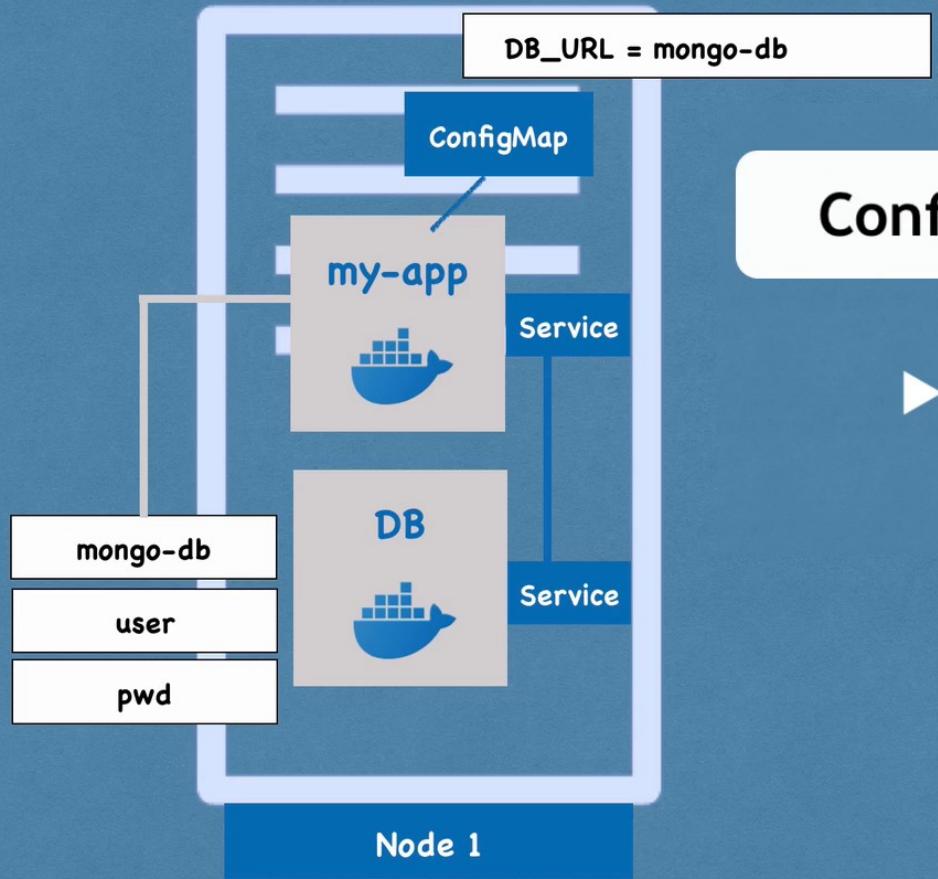


pull it in your pod



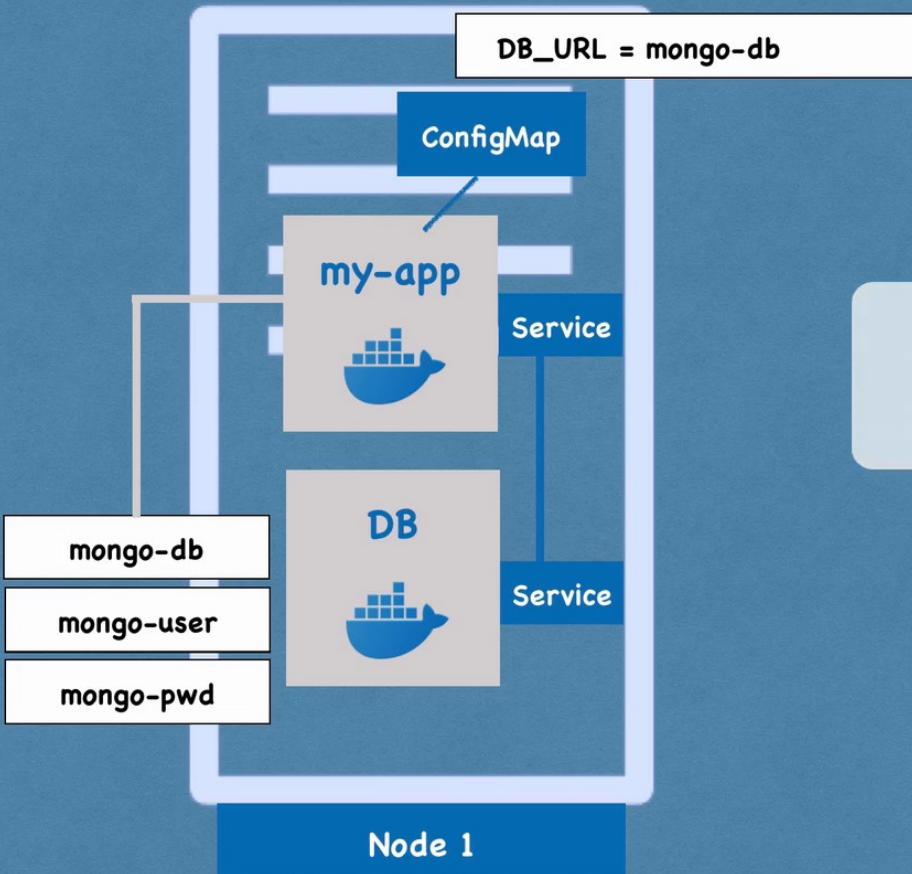


► external configuration  
of your application

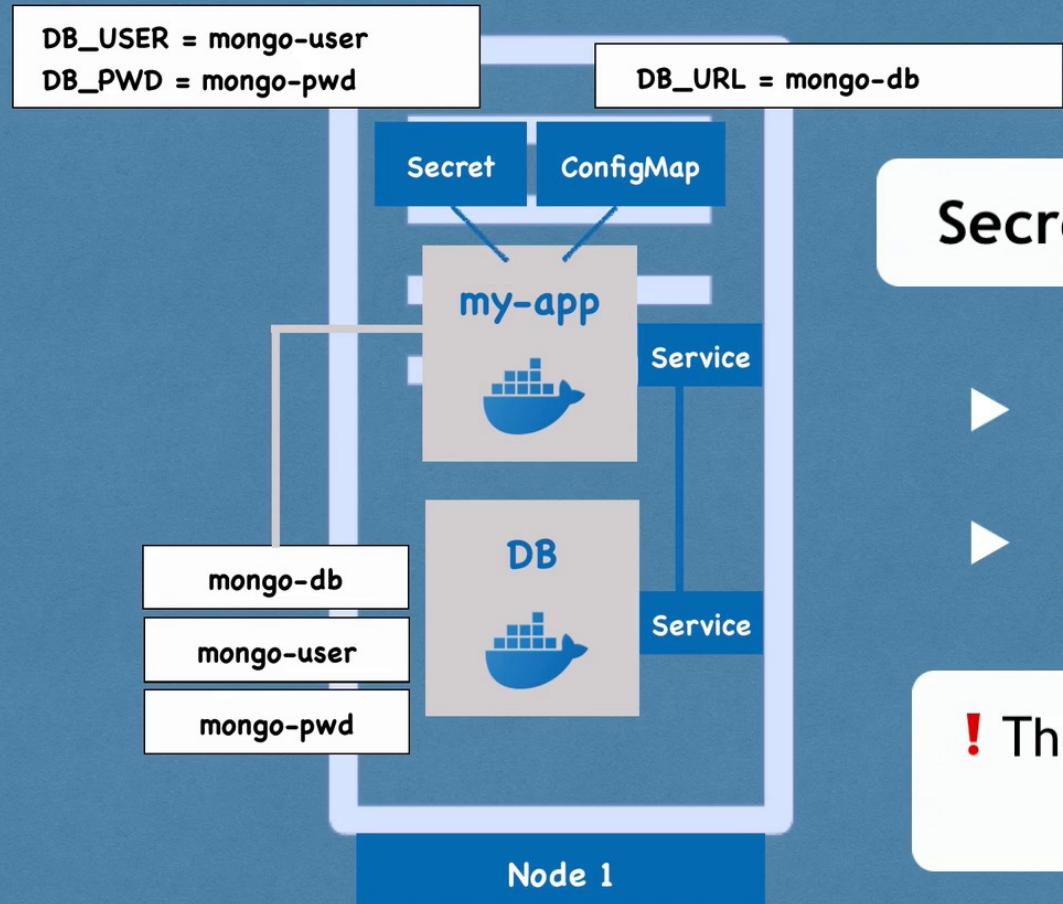


## ConfigMap:

- ▶ external configuration of your application



! Don't put credentials into  
ConfigMap

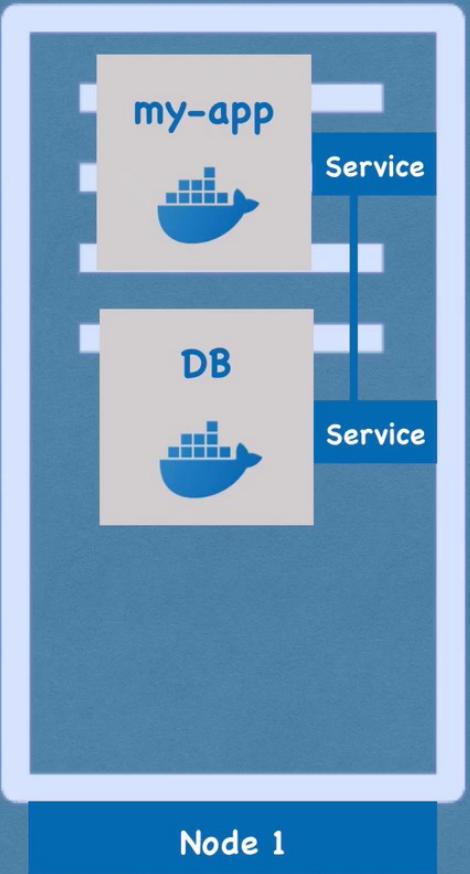


**Secret:**

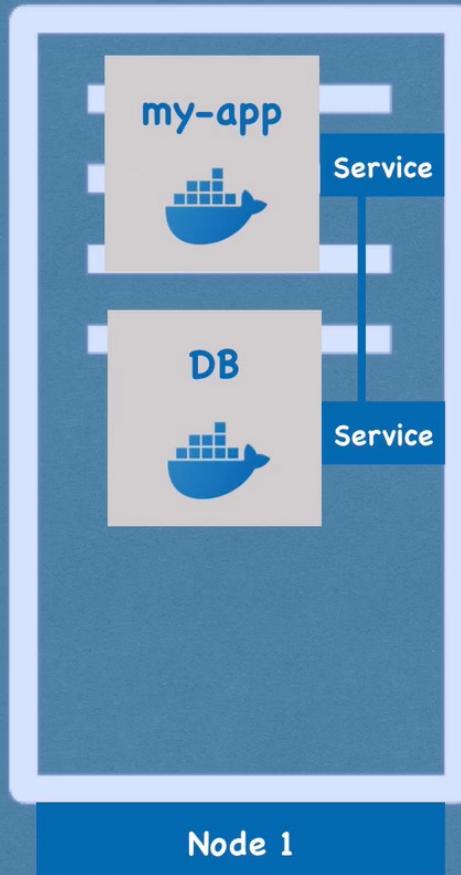


- ▶ used to store secret data
- ▶ base64 encoded

**!** The built-in security mechanism is not enabled by default!



 Use it as environment variables  
or as a properties file



✓ Pod

✓ Service

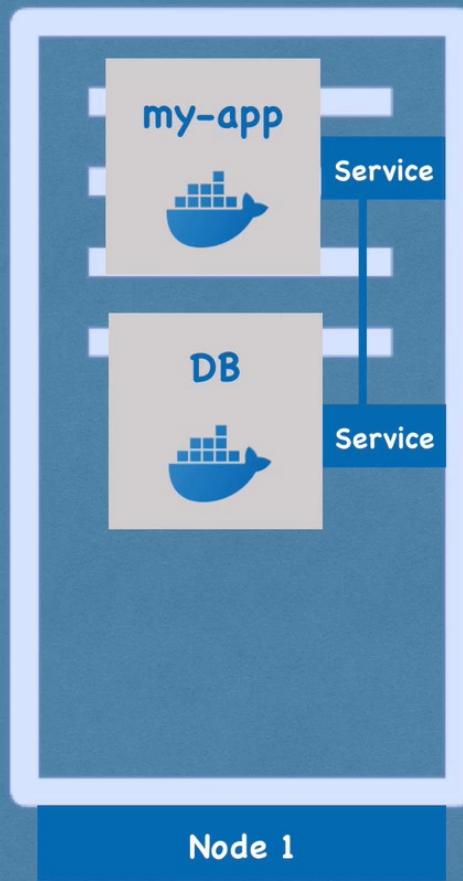
✓ ConfigMap

✓ Ingress

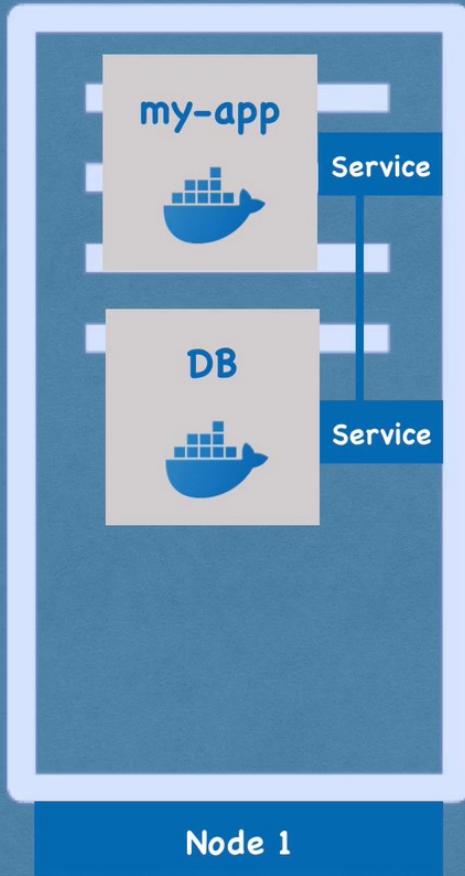
✓ Secrets

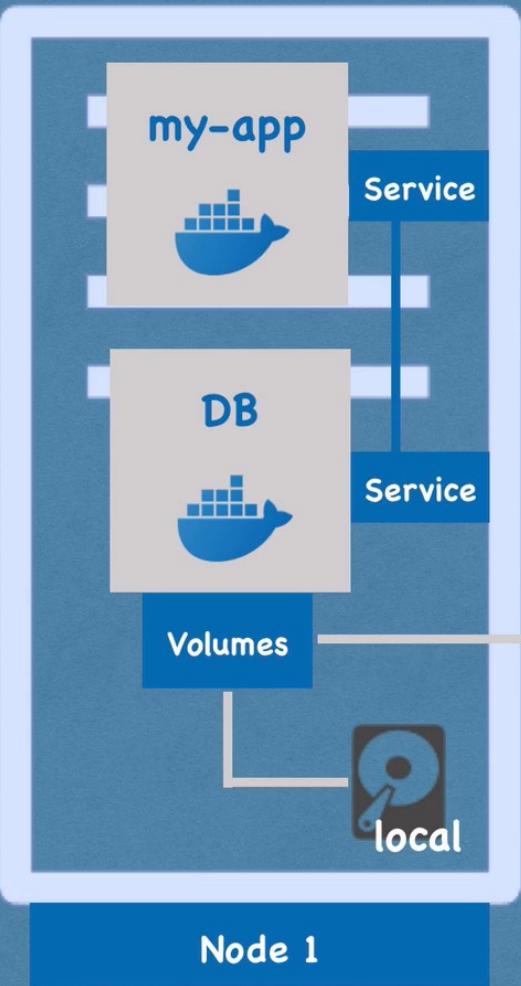


# Volumes



Data storage

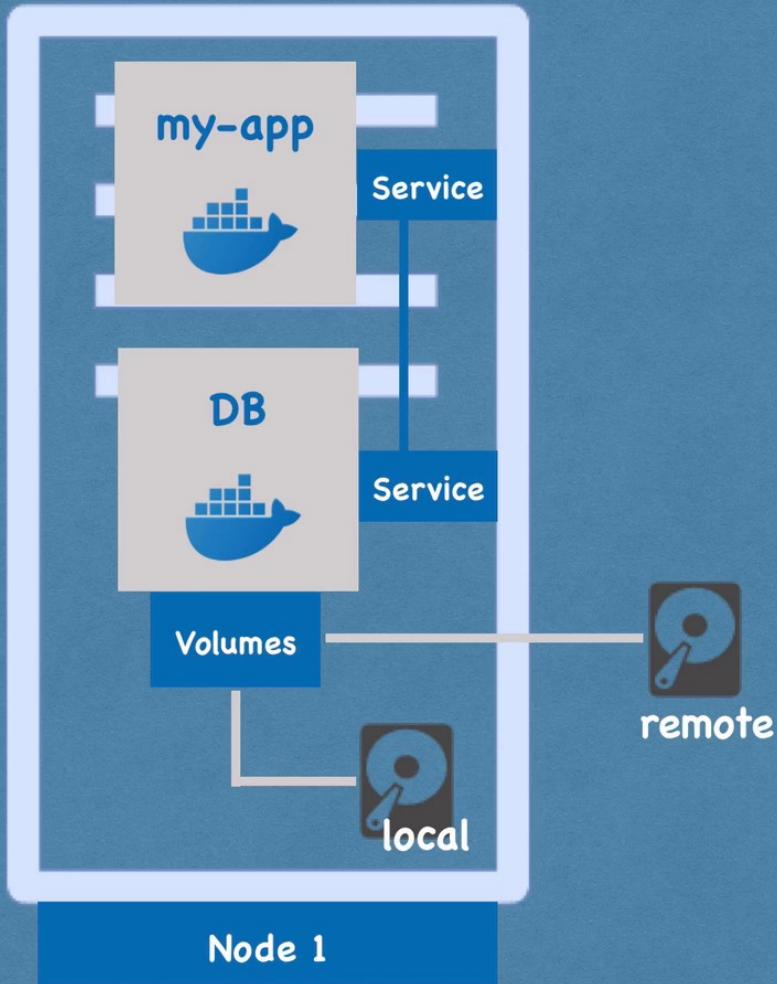




Storage on local machine

or **remote**, outside of the K8s cluster



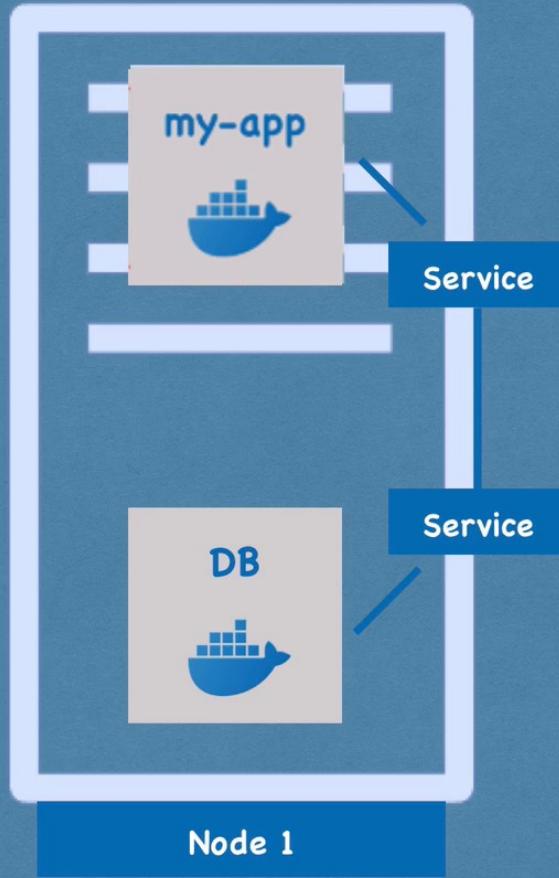


! K8s doesn't manage data persistence!

`https://my-app.com`



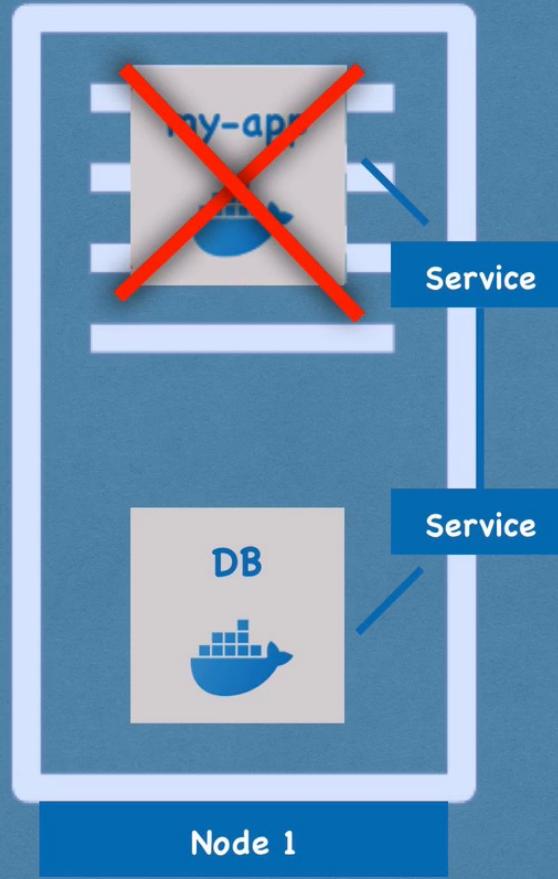
User

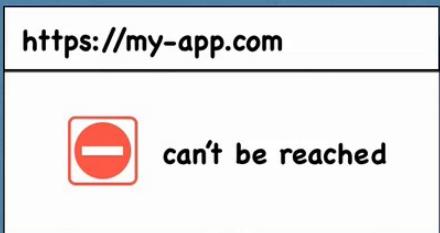


<https://my-app.com>

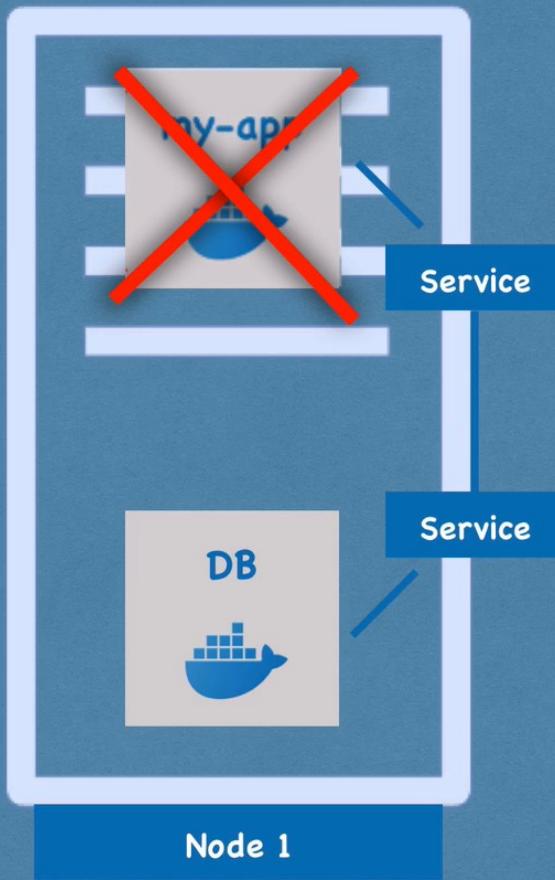


User

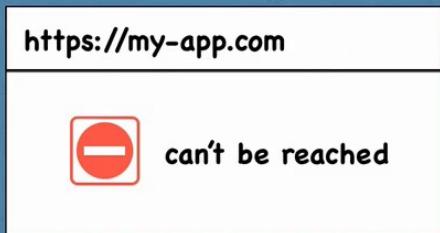




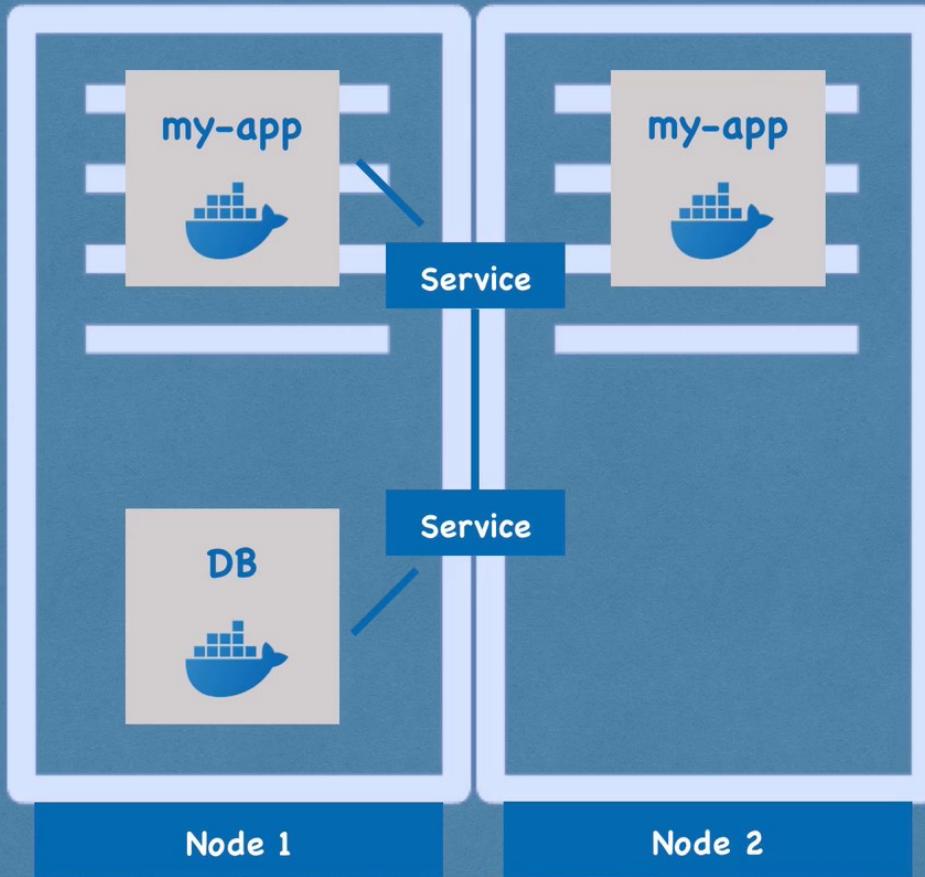
User



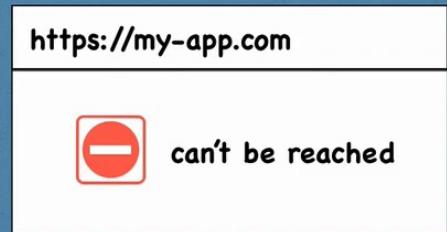
# Replicate everything



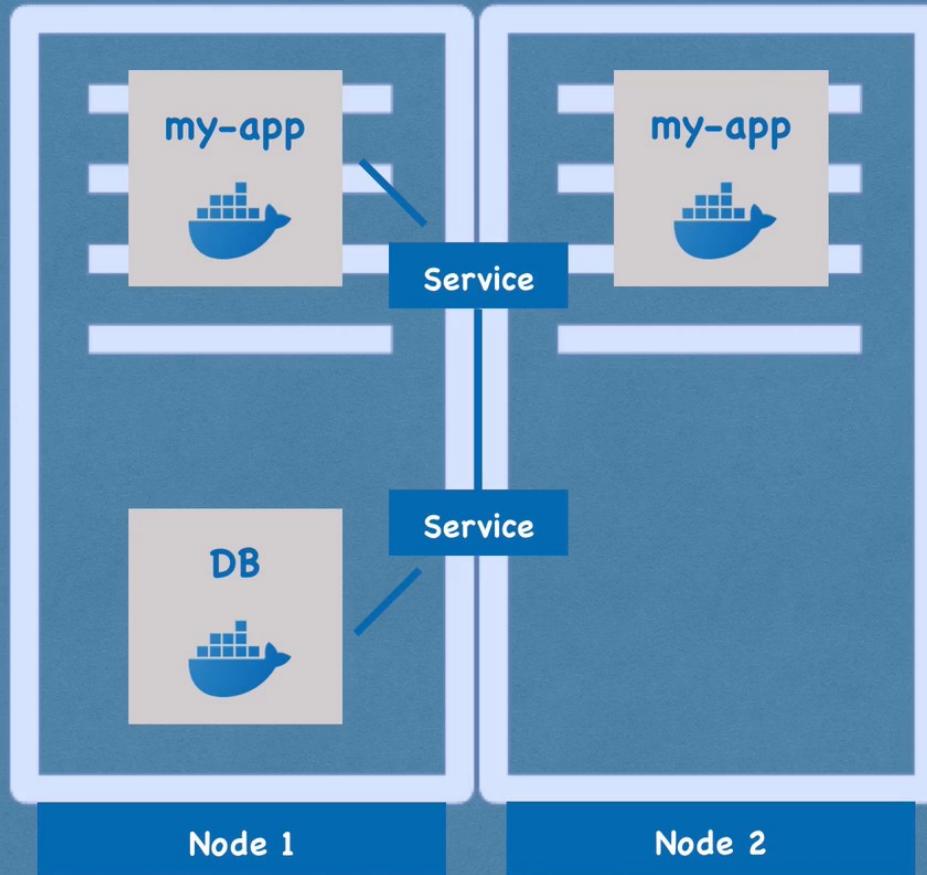
User

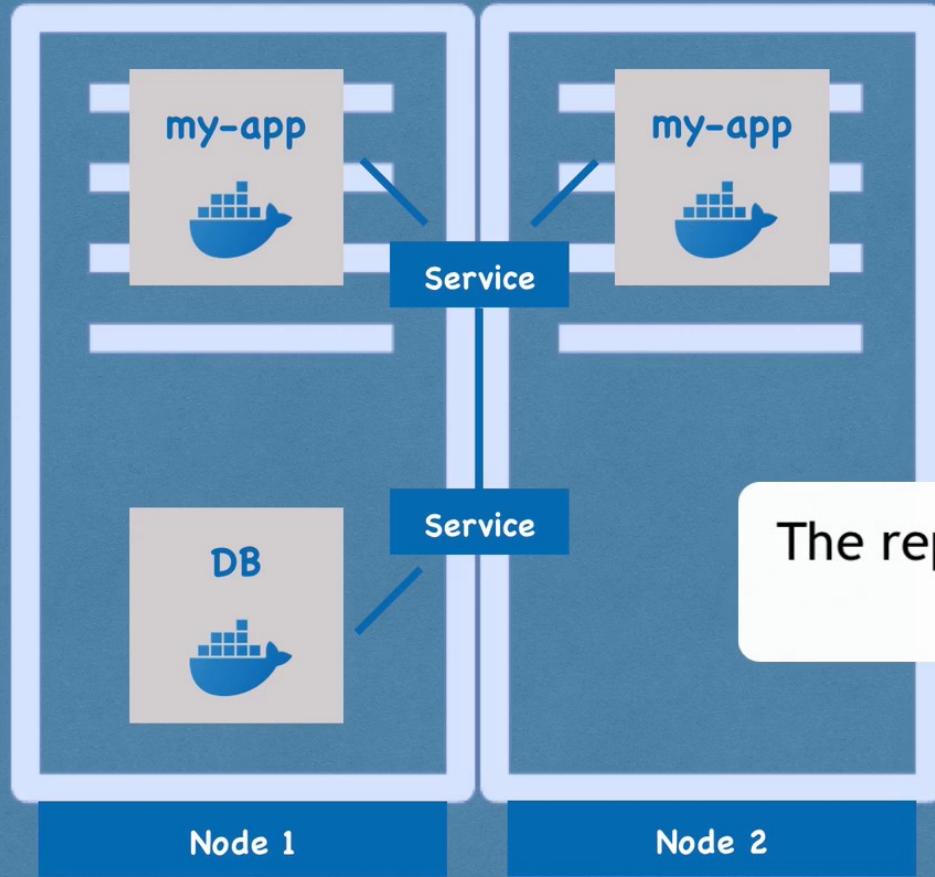
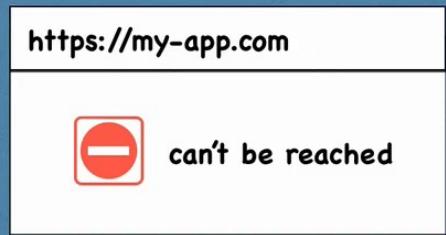


Replicate everything

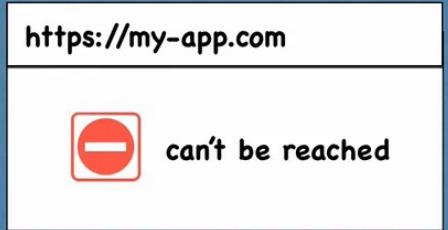


User

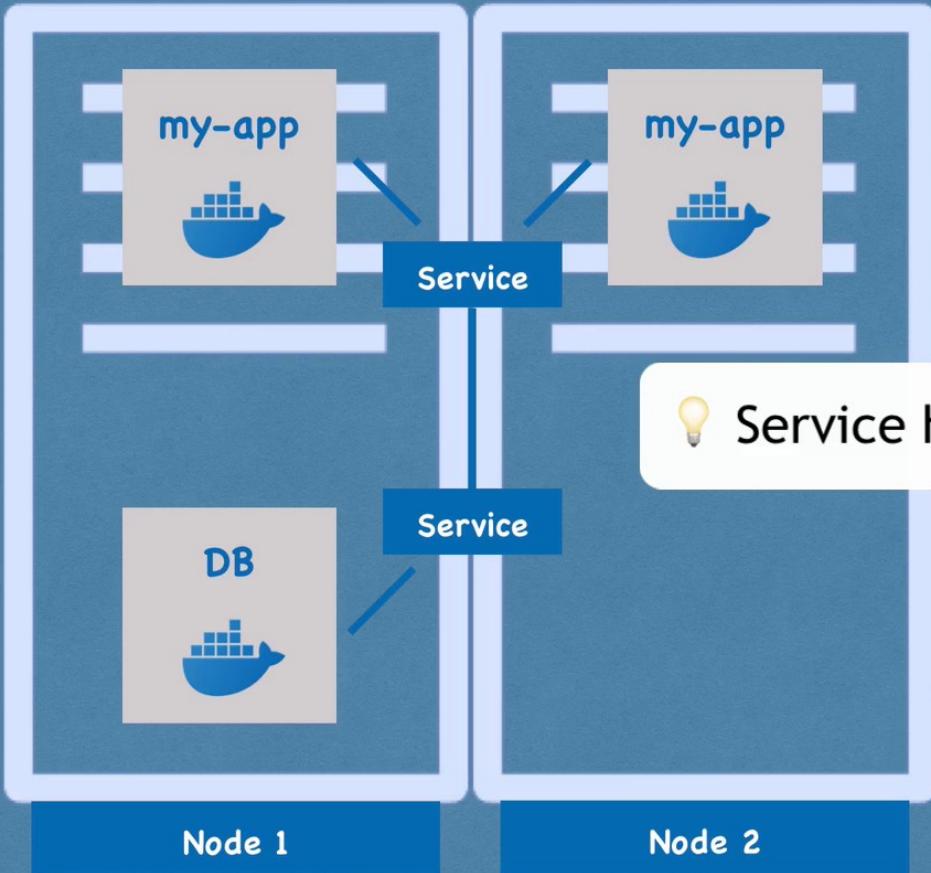




The replica is connected to the same Service



User

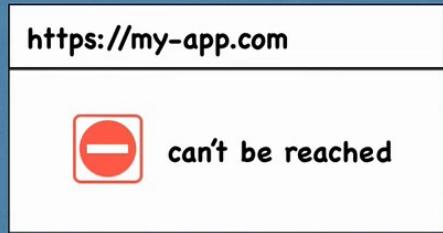


💡 Service has 2 functionalities:

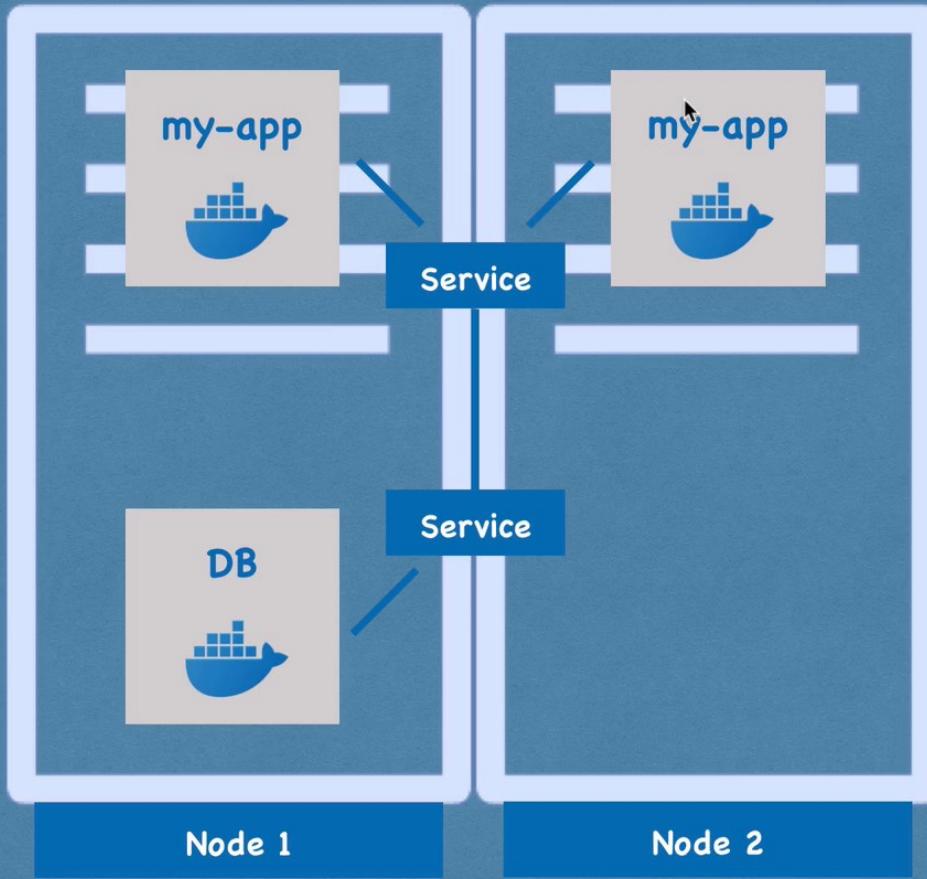
- permanent IP

- load balancer

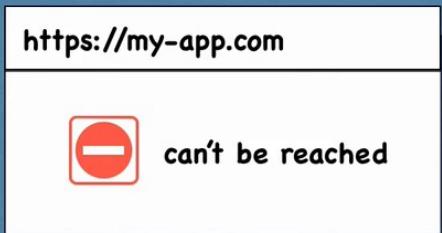
💡 define blueprints for pods



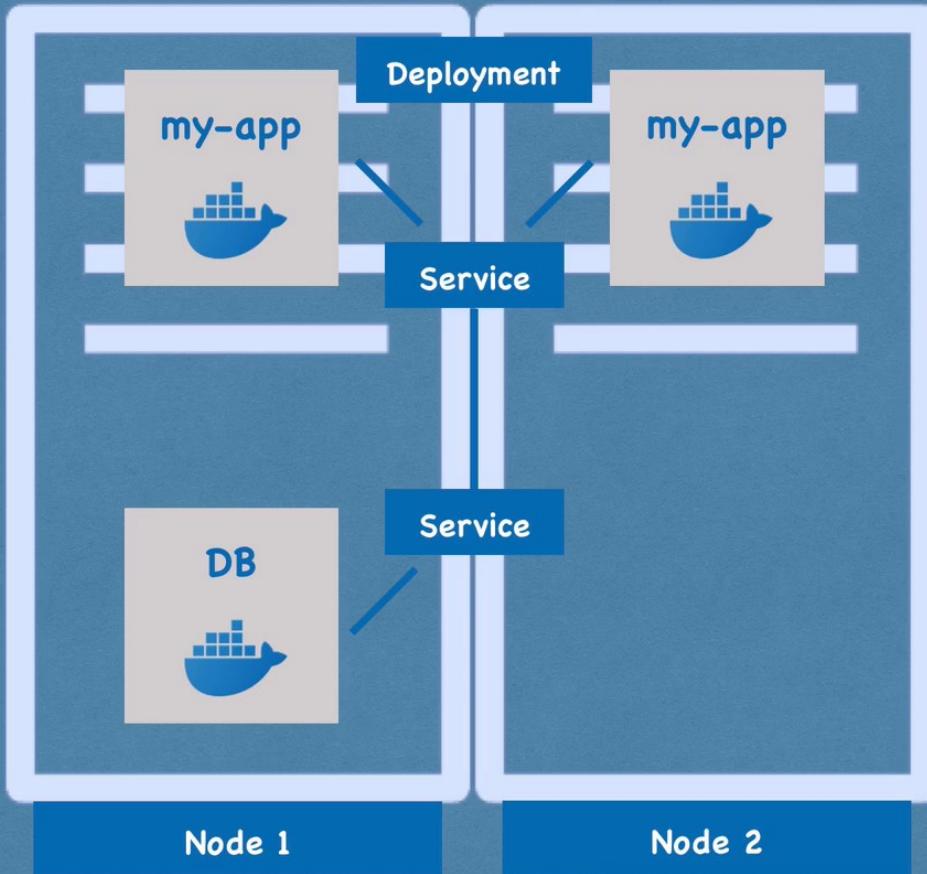
User



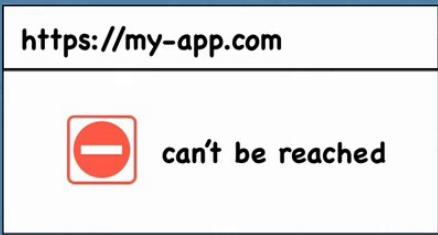
## Deployment:



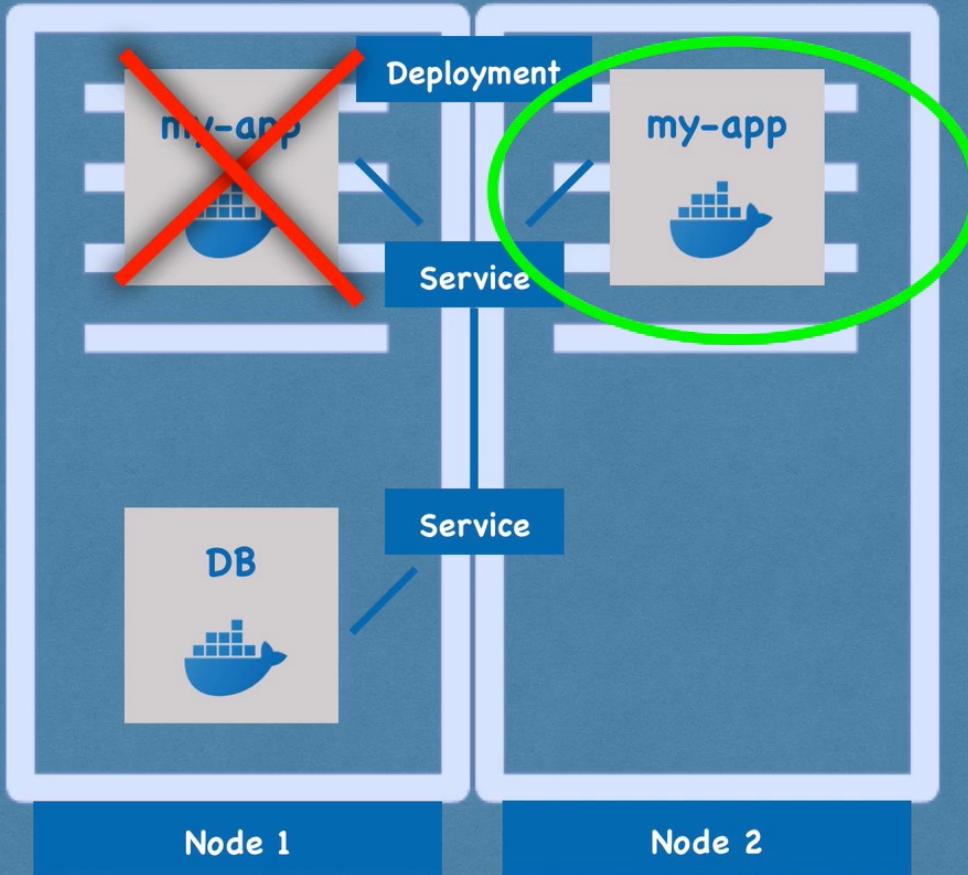
User

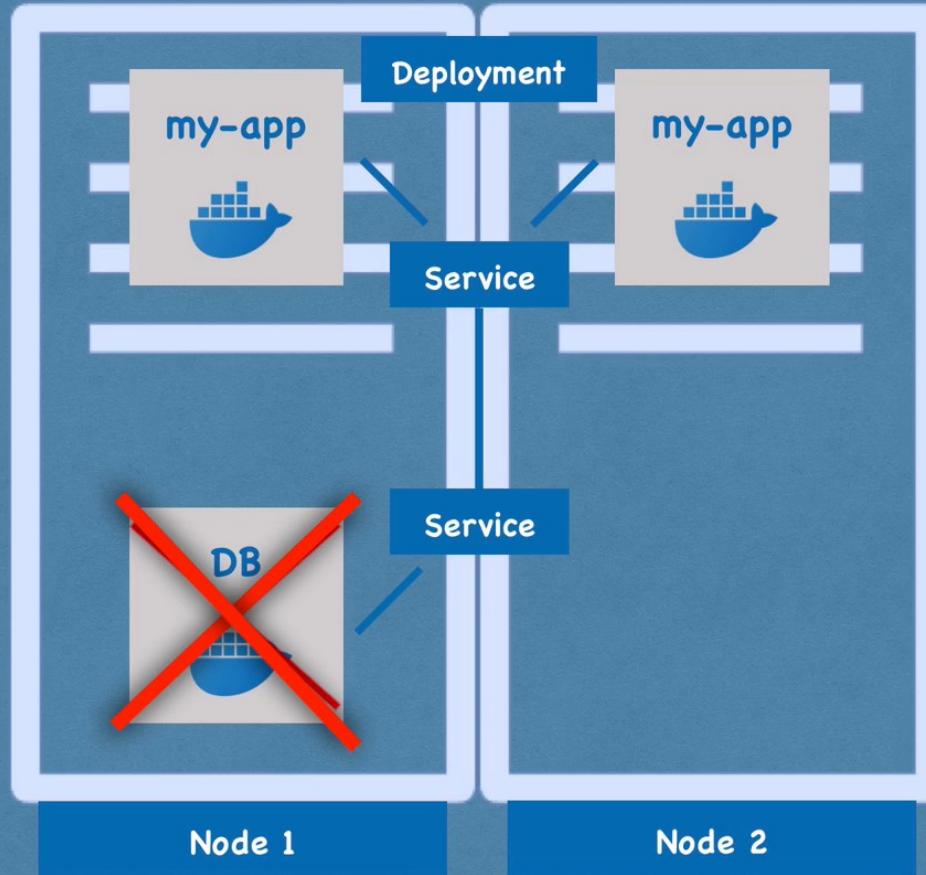


- ▶ blueprint for *my-app* pods
- ▶ you create Deployments
- ▶ abstraction of Pods

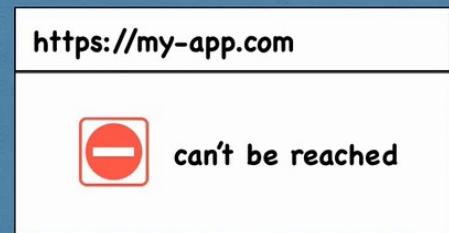


User

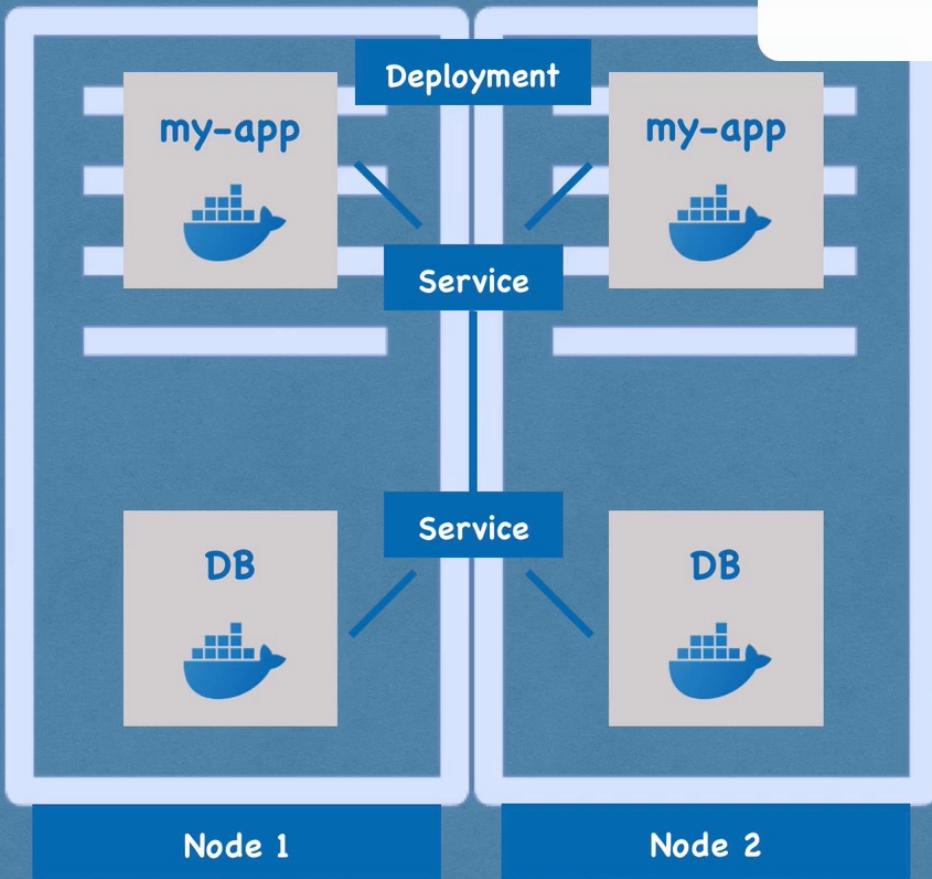




! DB can't be replicated via Deployment!



User

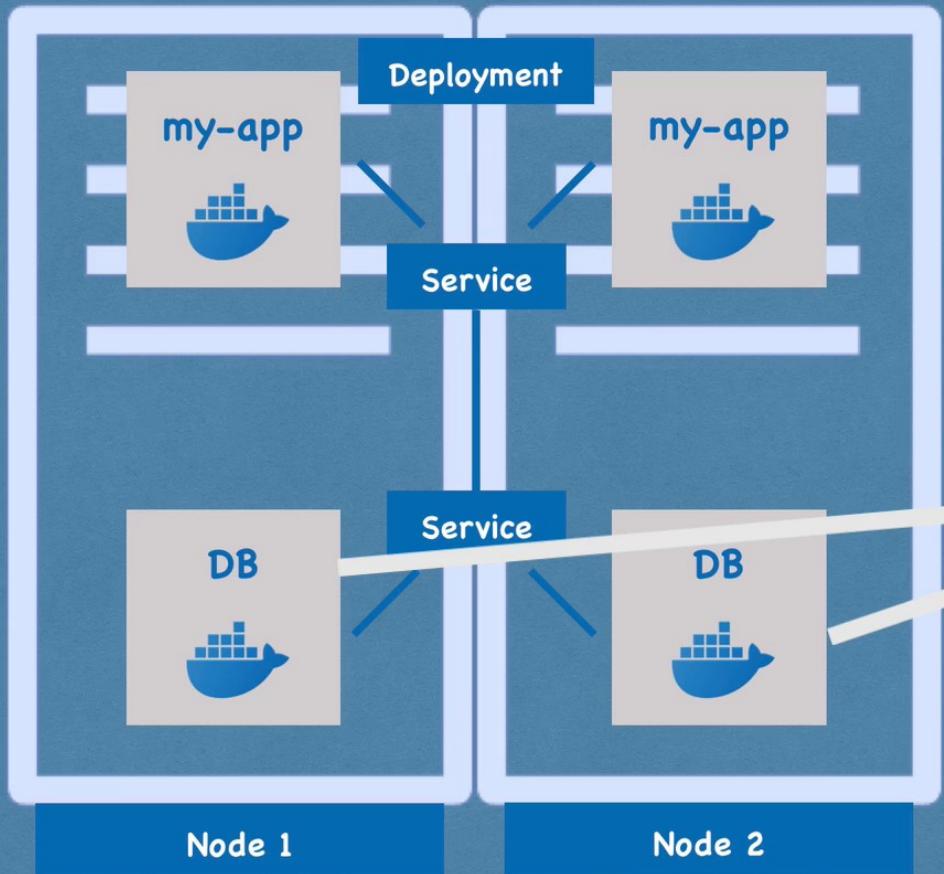


<https://my-app.com>

 can't be reached



User

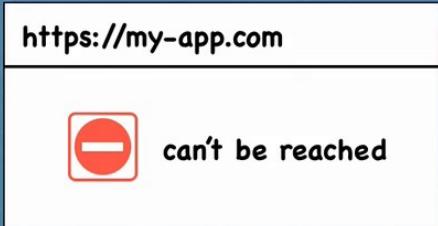


! Avoid data inconsistencies

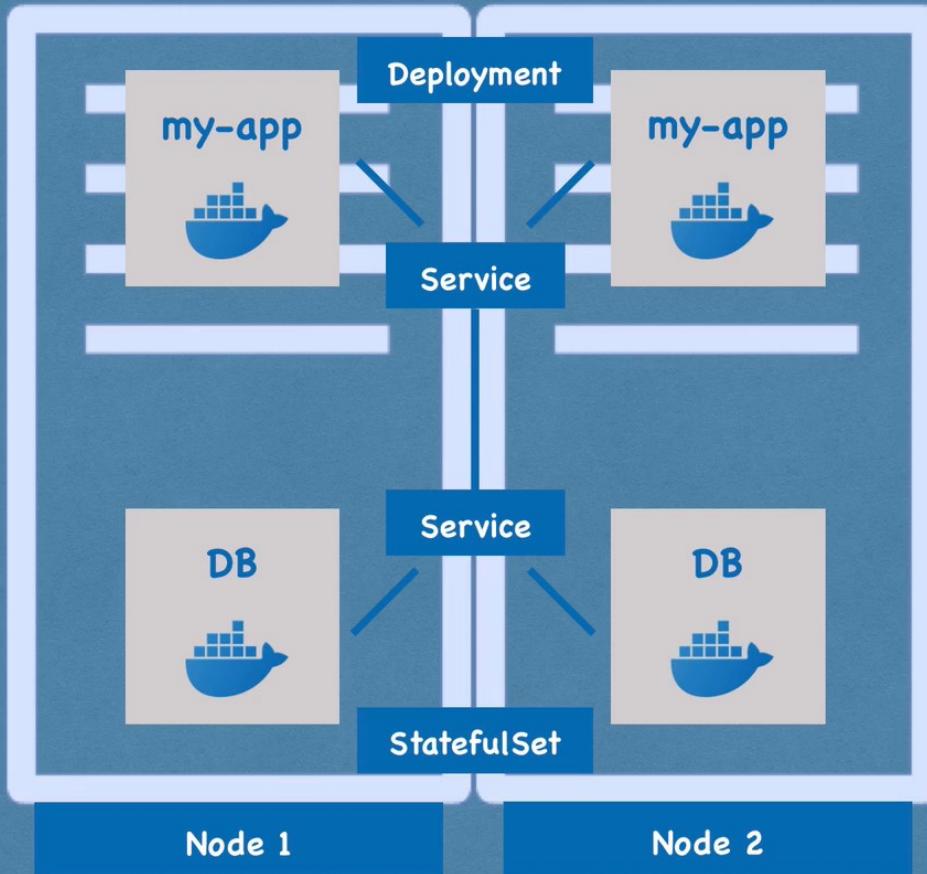




StatefulSet



User



► for STATEFUL apps



elastic



mongoDB



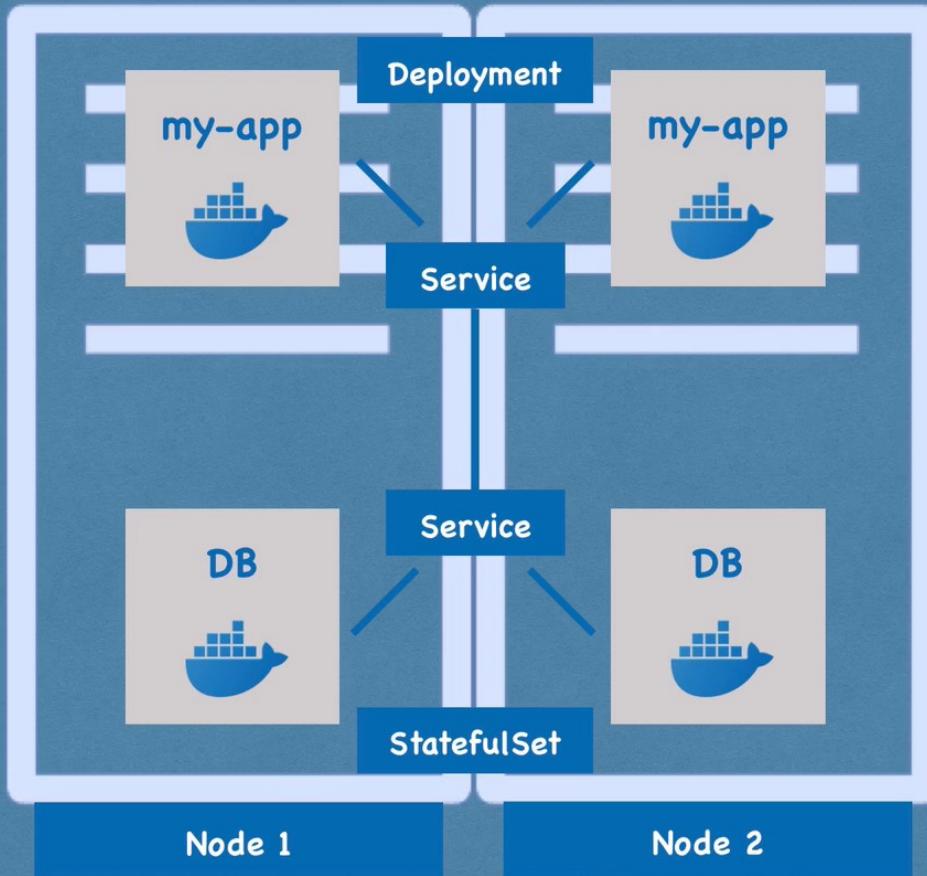
MySQL™



`https://my-app.com`  
✗ can't be reached



User



Deployment for  
stateLESS Apps

StatefulSet for  
stateFUL Apps  
or Databases

Deploying StatefulSet not  
easy

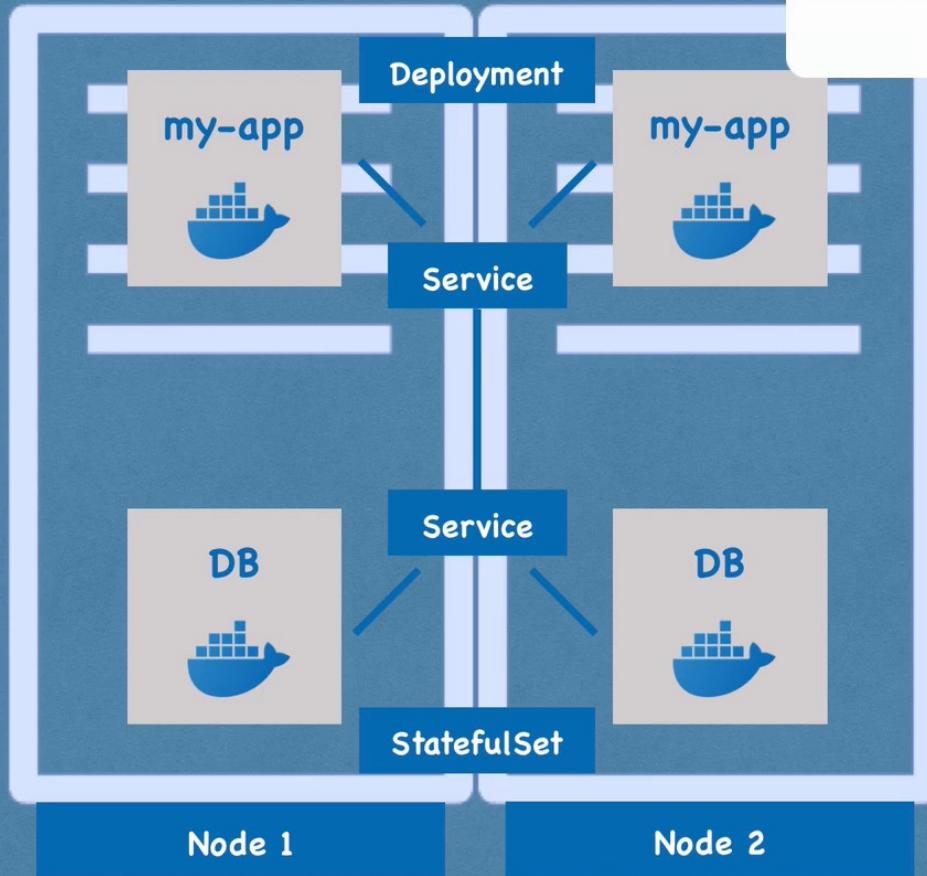


<https://my-app.com>

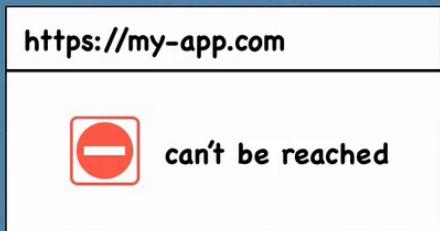
 can't be reached



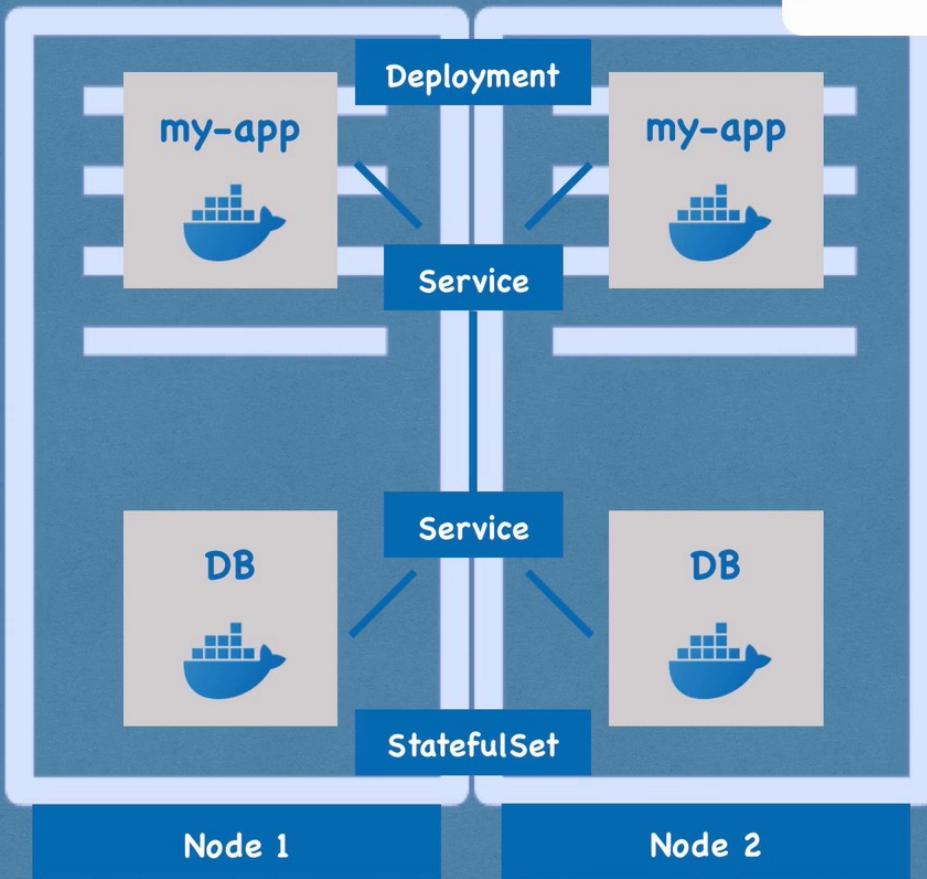
User



DB are often hosted outside of K8s cluster



User

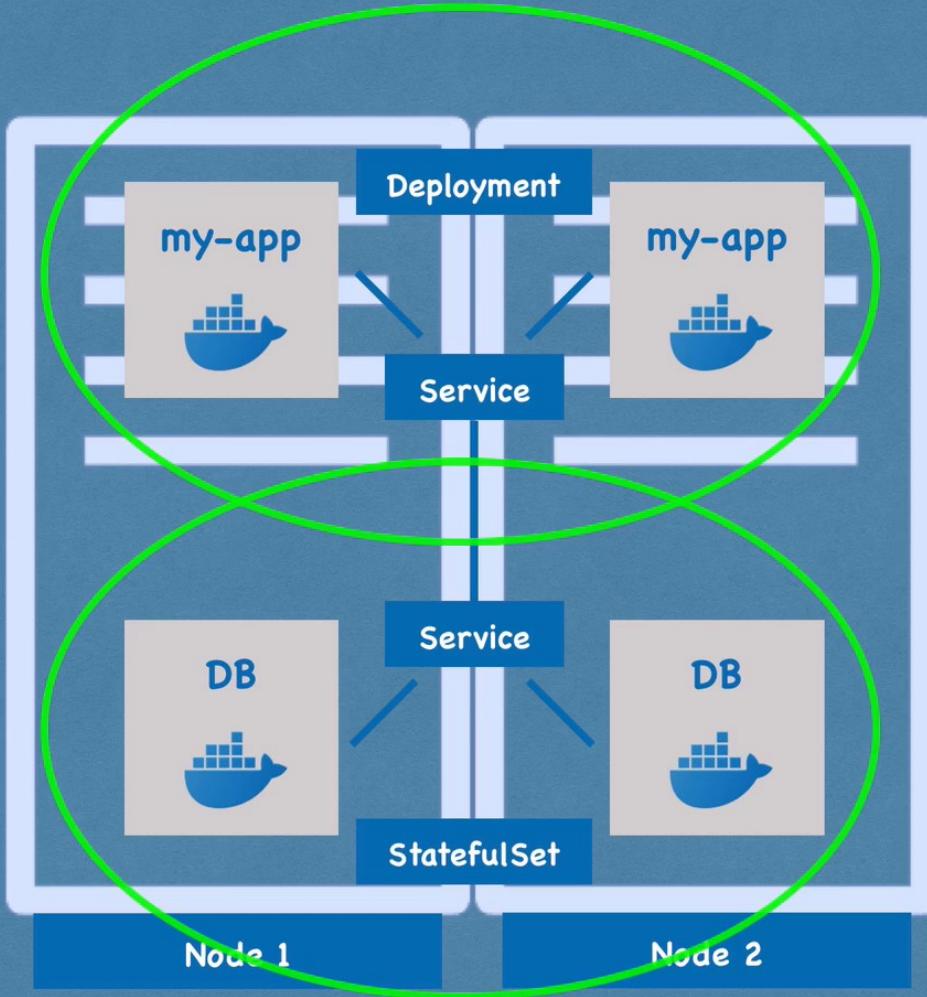


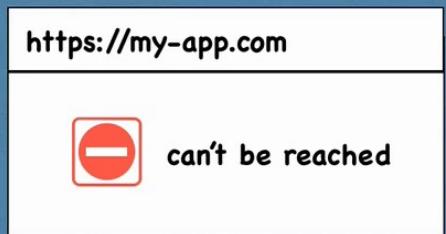
`https://my-app.com`

 can't be reached

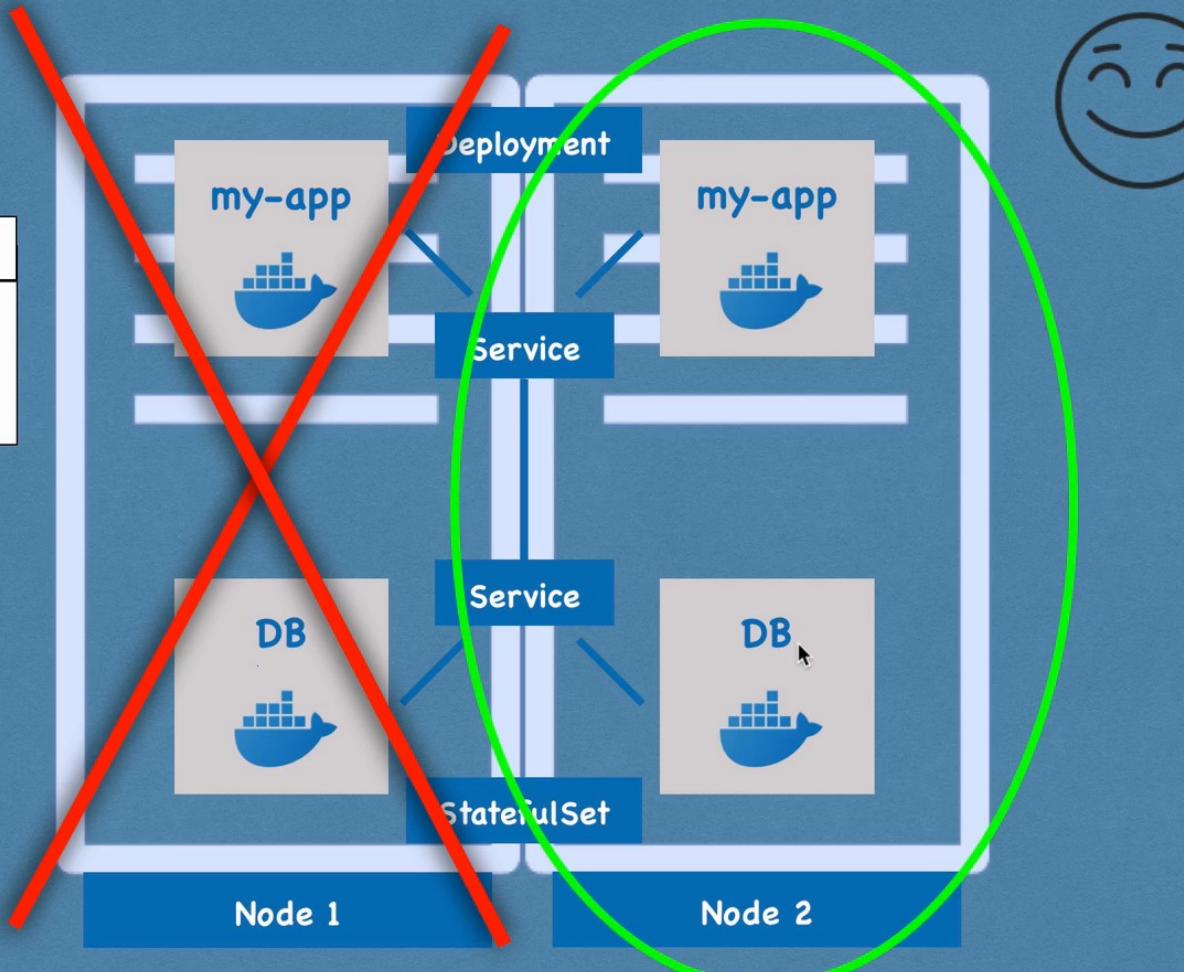


User





User





# Main Kubernetes Components summarized

★ abstraction of containers



Pod

Service



★ communication



Ingress

★ route traffic into cluster



# Main Kubernetes Components summarized



Pod

Service

★ external configuration

Ingress

ConfigMap



Secrets





# Main Kubernetes Components summarized



Volumes



Pod

Service

Ingress

ConfigMap

Secrets



# Main Kubernetes Components summarized

ConfigMap

Secrets

Volumes



Deployment

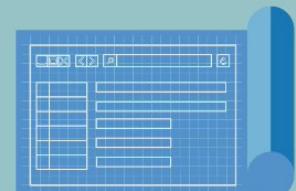


StatefulSet

Pod

Service

Ingress





# Main Kubernetes Components summarized

Volumes

Pod

Service

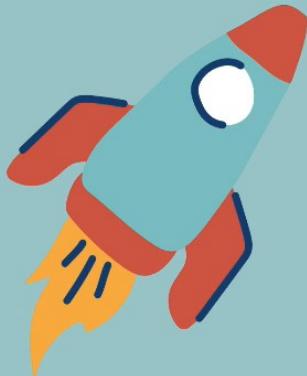
Ingress

ConfigMap

Deployment

Secrets

StatefulSet





# Kubernetes Architecture



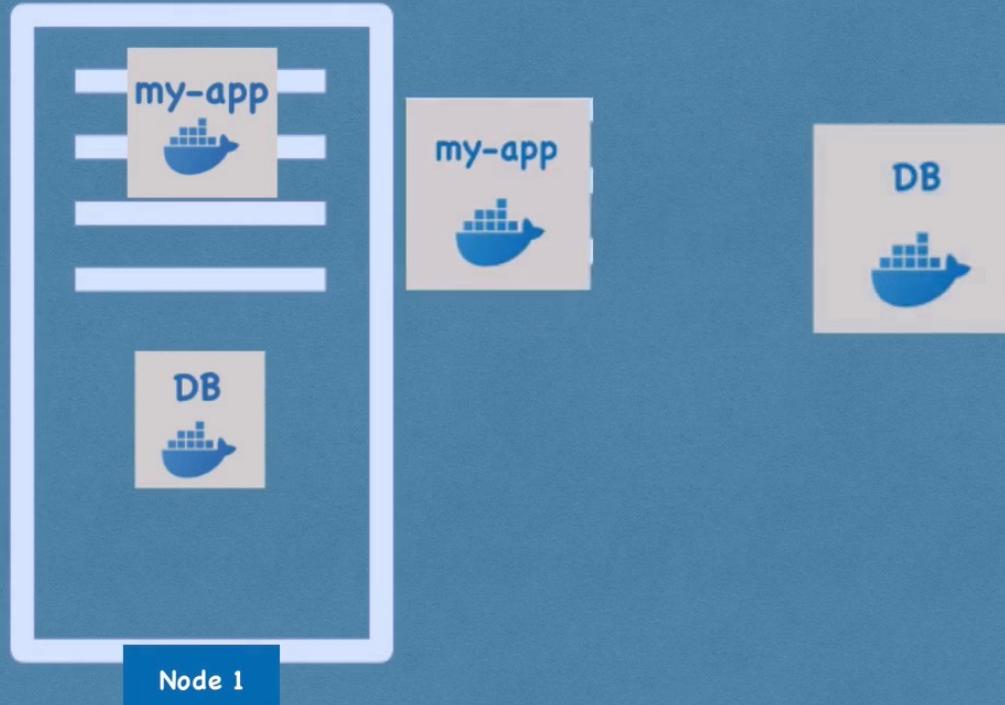




# Node processes

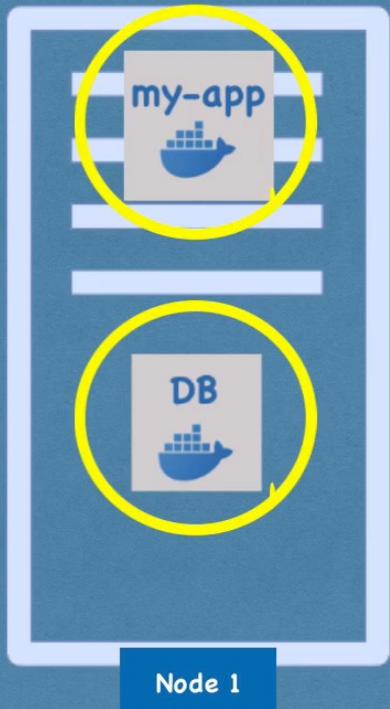


# Worker machine in K8s cluster





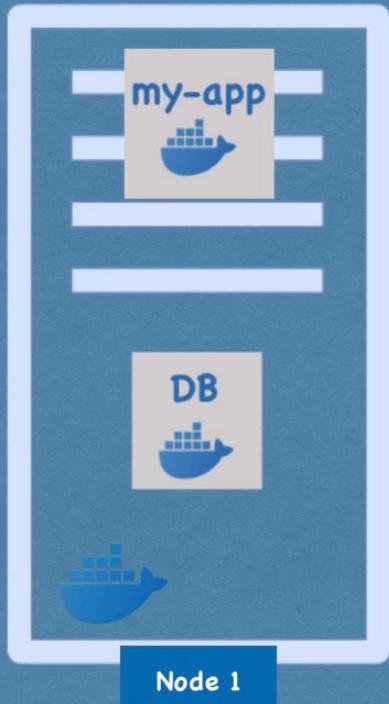
# Worker machine in K8s cluster



- ▶ each Node has multiple Pods on it



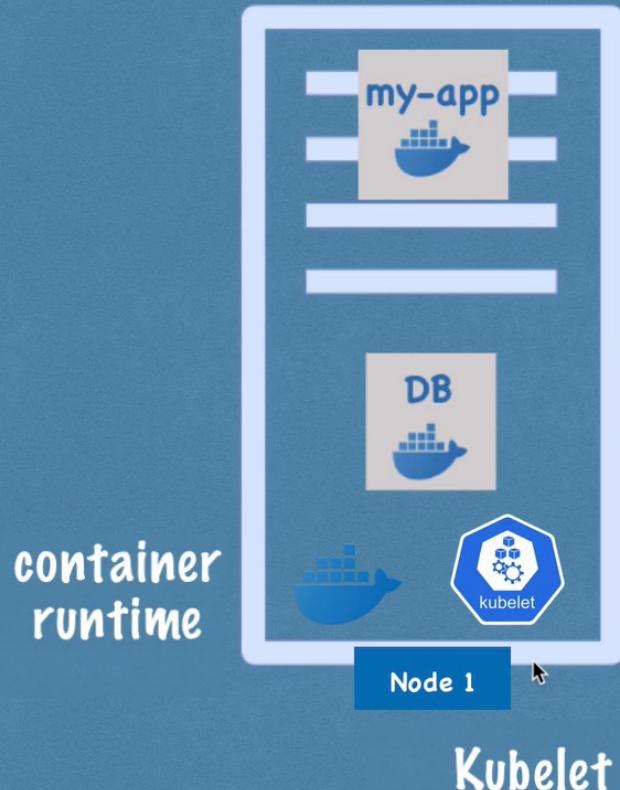
# Worker machine in K8s cluster



- ▶ each Node has multiple Pods on it
- ▶ 3 processes must be installed on every Node
- ▶ Worker Nodes do the actual work



# Worker machine in K8s cluster



Kubelet interacts with both - the container and node

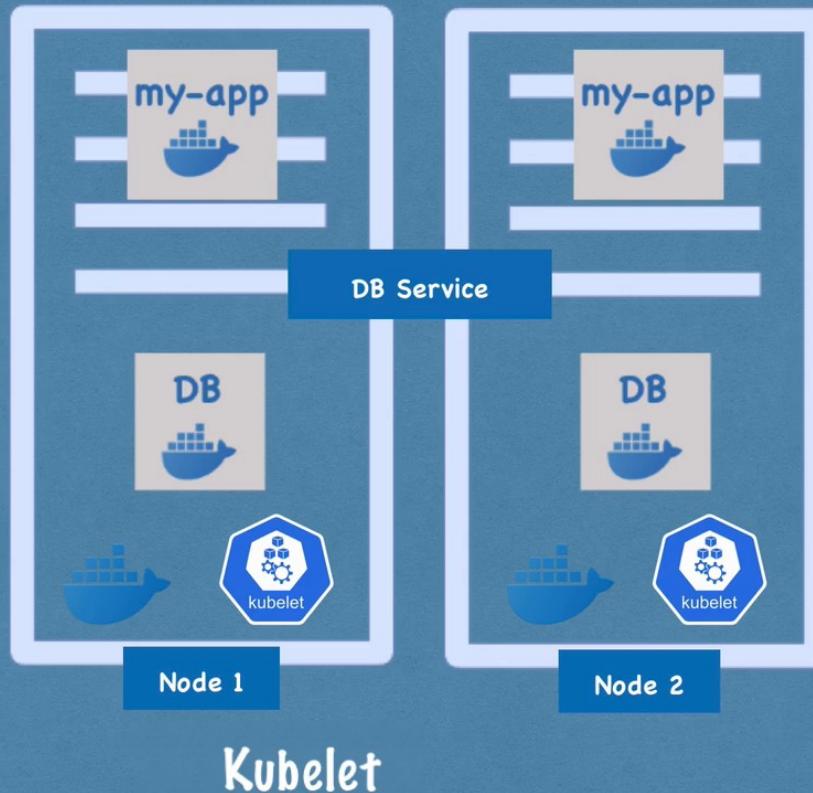
Kubelet starts the pod with a container inside



# Worker machine in K8s cluster

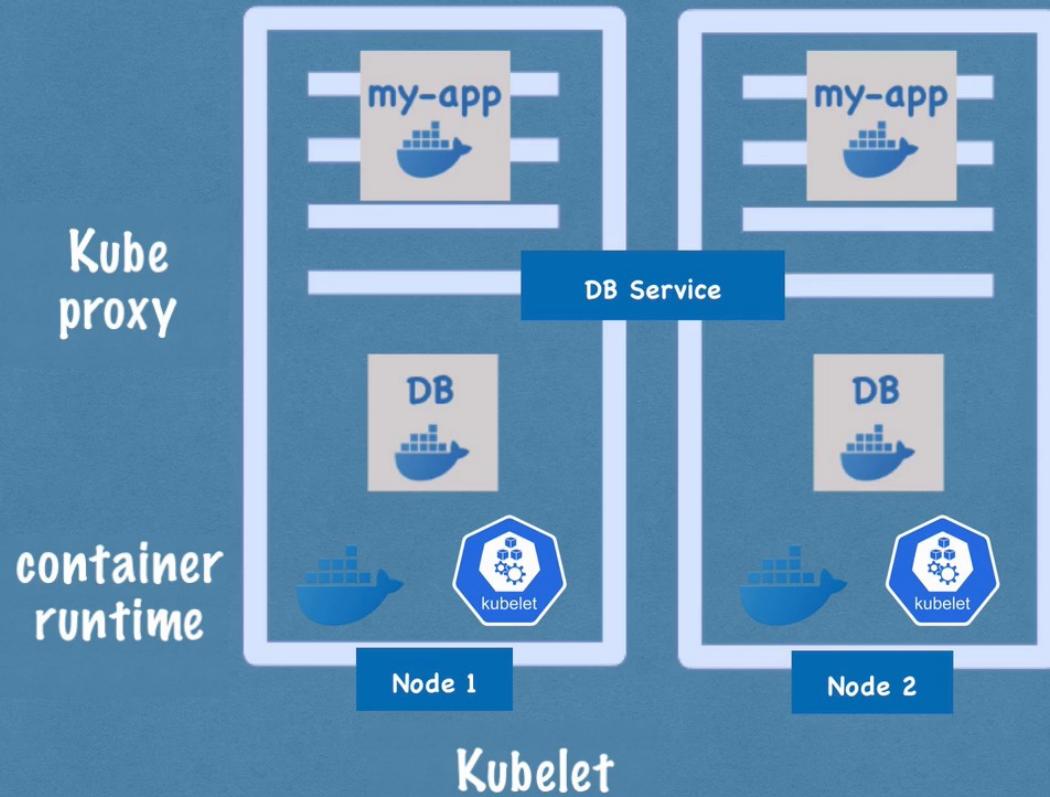
Communication  
via Services

container  
runtime





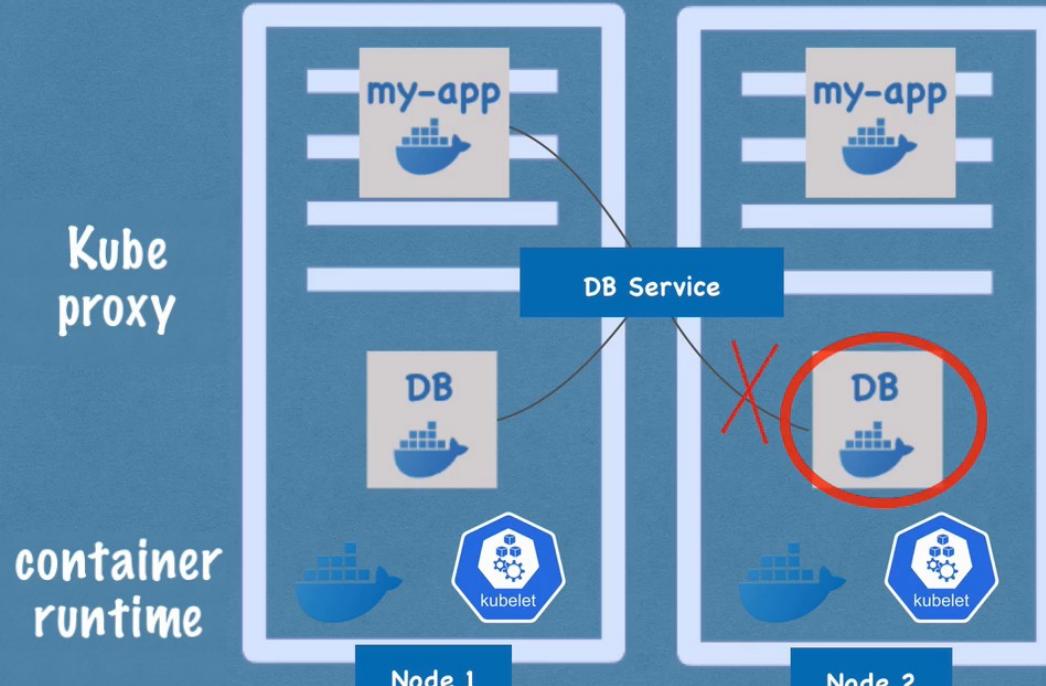
# Worker machine in K8s cluster



**Kube Proxy  
forwards the  
requests**



# Worker machine in K8s cluster



Kube  
proxy

container  
runtime

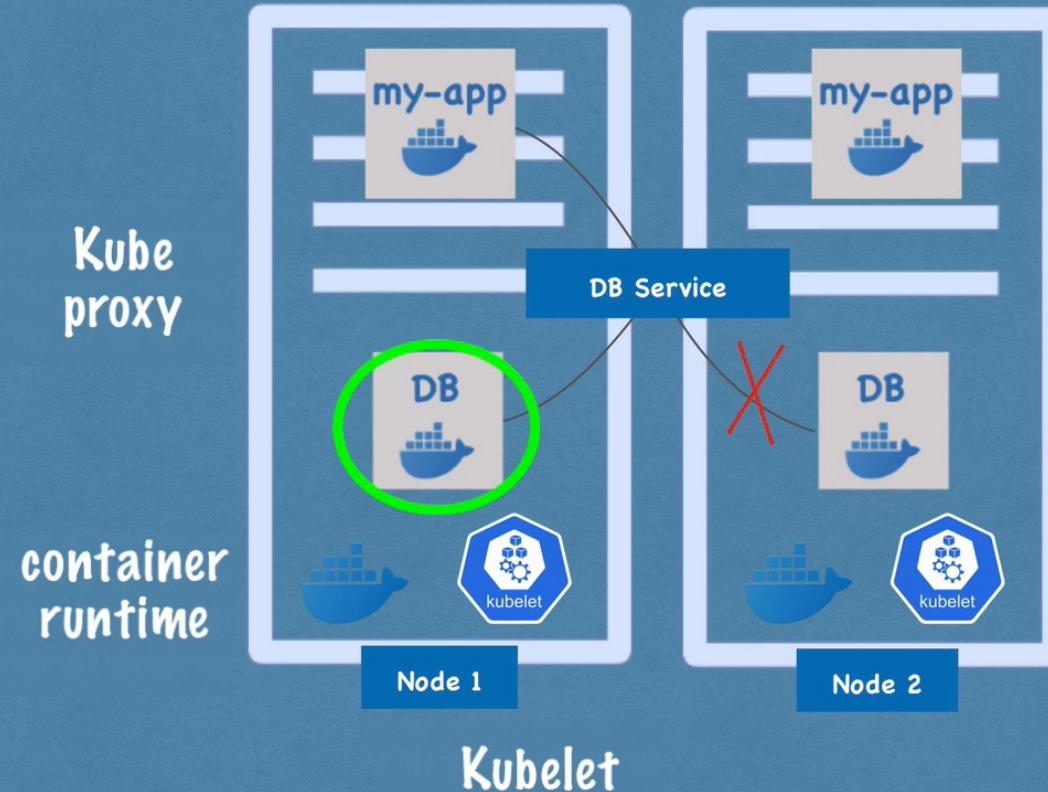
Node 1

Node 2

Kubelet

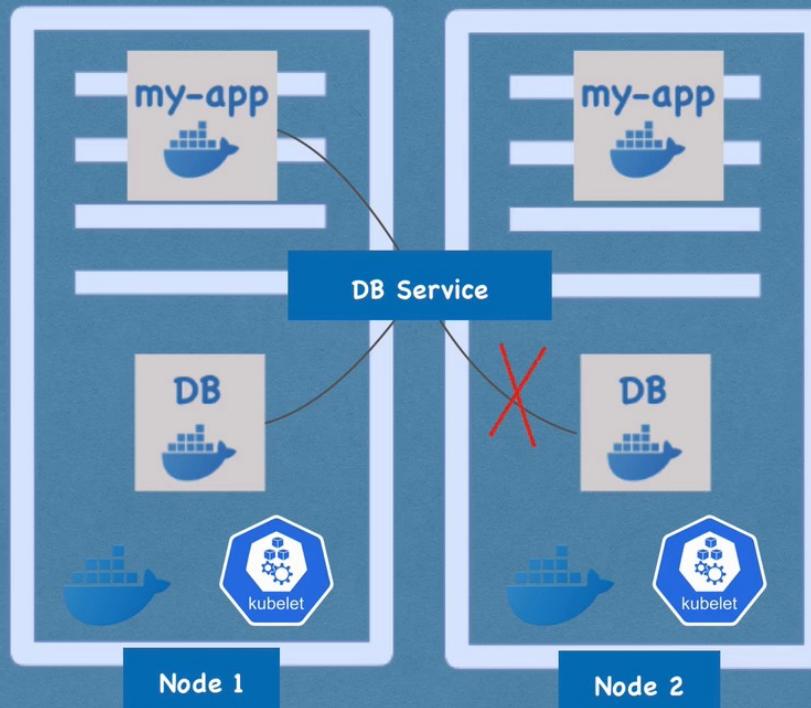


# Worker machine in K8s cluster





# Worker machine in K8s cluster



**3 Node processes:**

1. Kubelet

2. Kube Proxy

3. Container runtime



# So, how do you interact with this cluster?

How to:

- ▶ schedule pod?
- ▶ monitor?
- ▶ re-schedule/re-start pod?
- ▶ join a new Node?

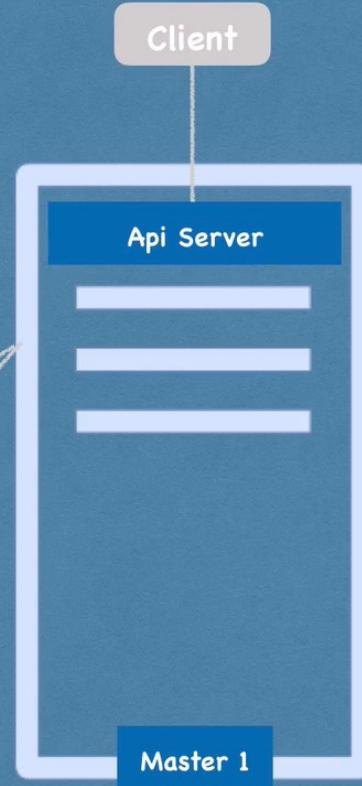
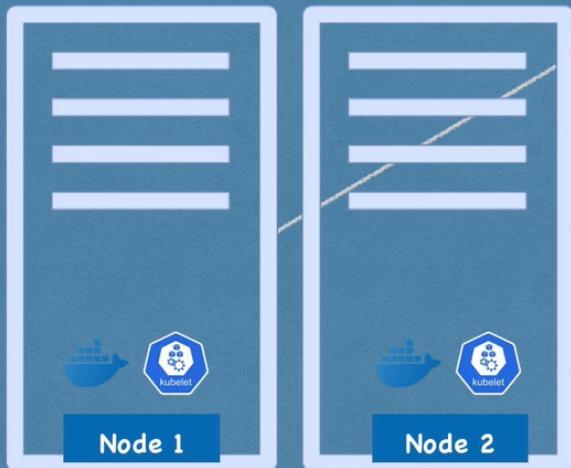


**Managing processes are done by  
Master Nodes**





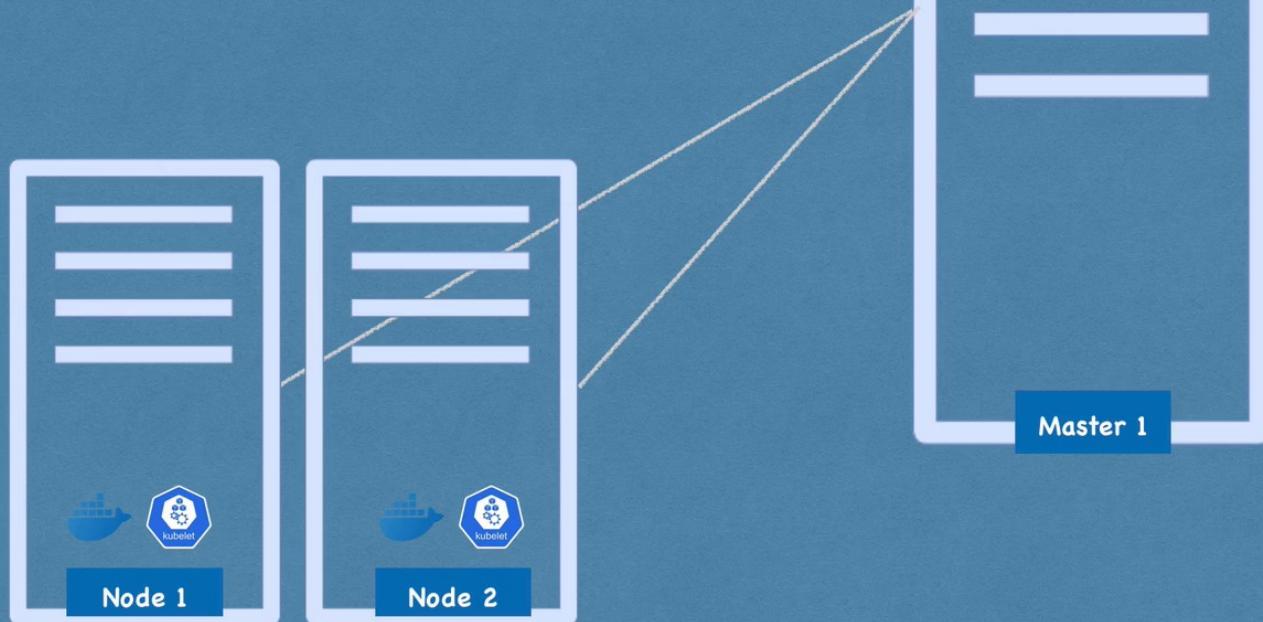
# Master processes



4 processes run on every master node!



# Master processes

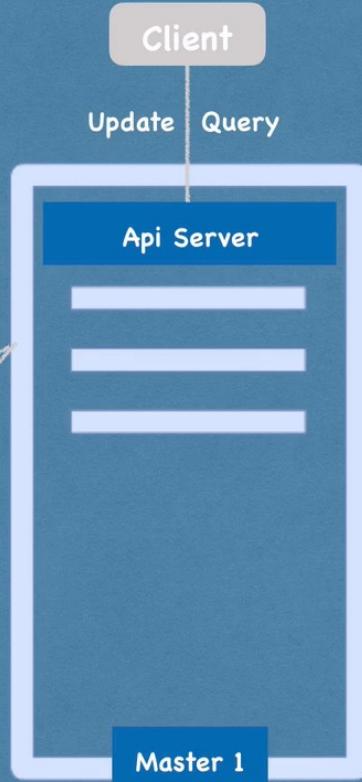
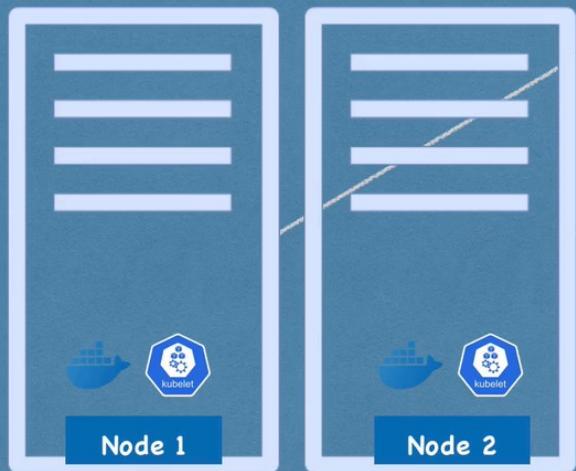


- ▶ cluster gateway
- ▶ acts as a gatekeeper for authentication!



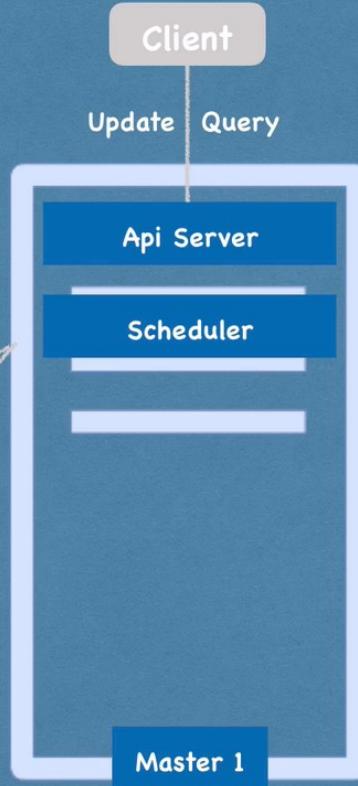
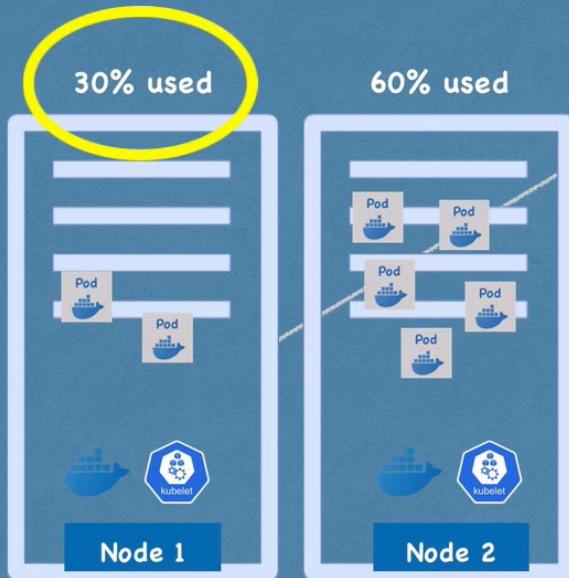


# Master processes





# Master processes



Schedule new Pod

API Server

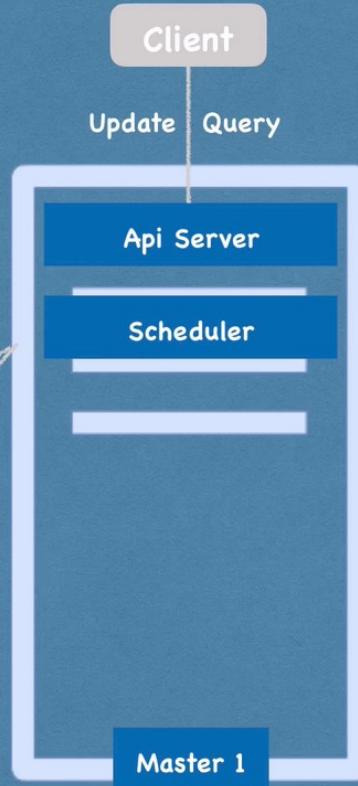
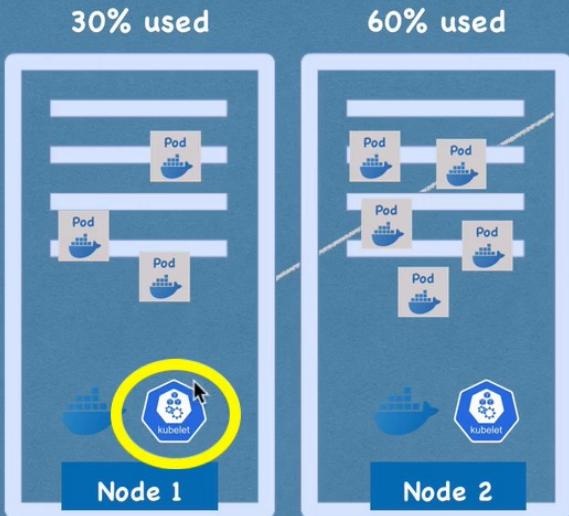
Scheduler

Where to put the Pod?



# Master processes

Scheduler just decides on which Node new Pod should be scheduled



Schedule new Pod

API Server

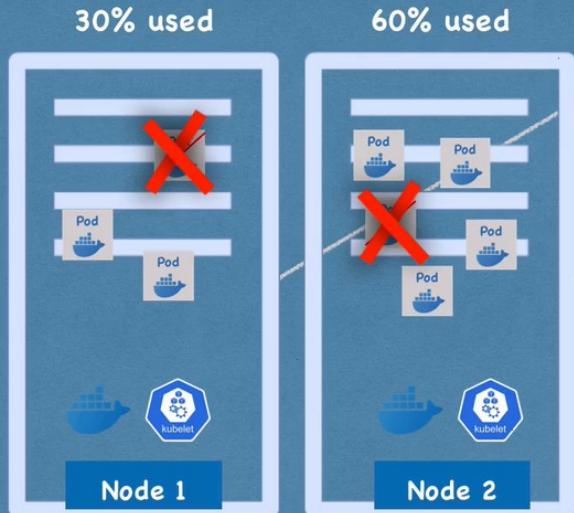
Scheduler

Where to put the Pod?

Kubelet



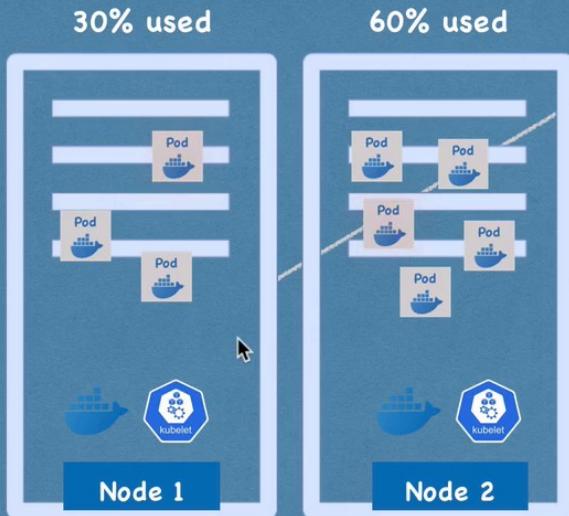
# Master processes



► detects cluster state changes

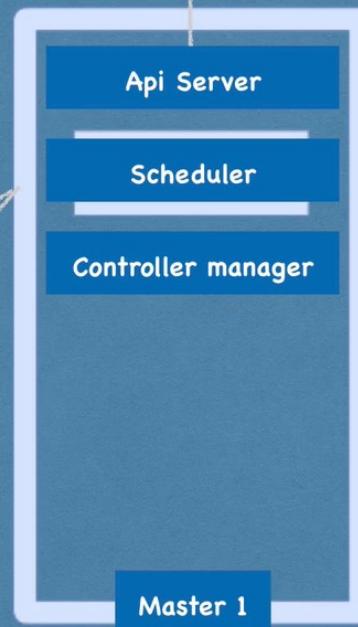


# Master processes



Client

Update Query



Controller Manager

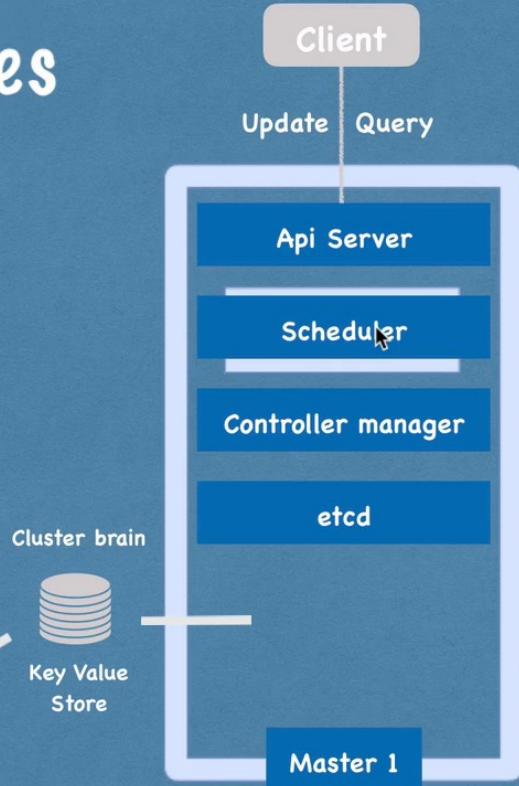
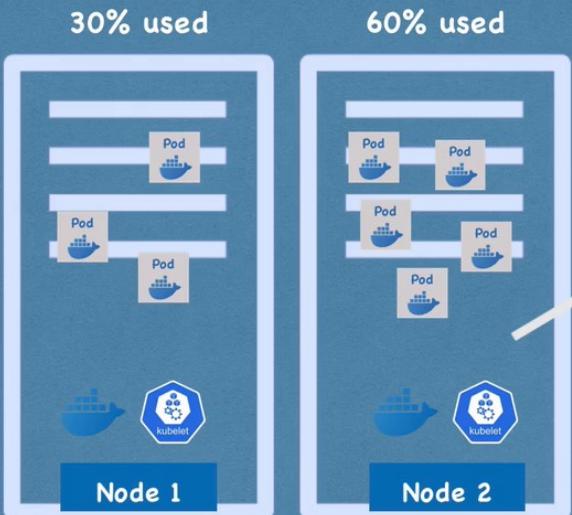
Scheduler

Kubelet





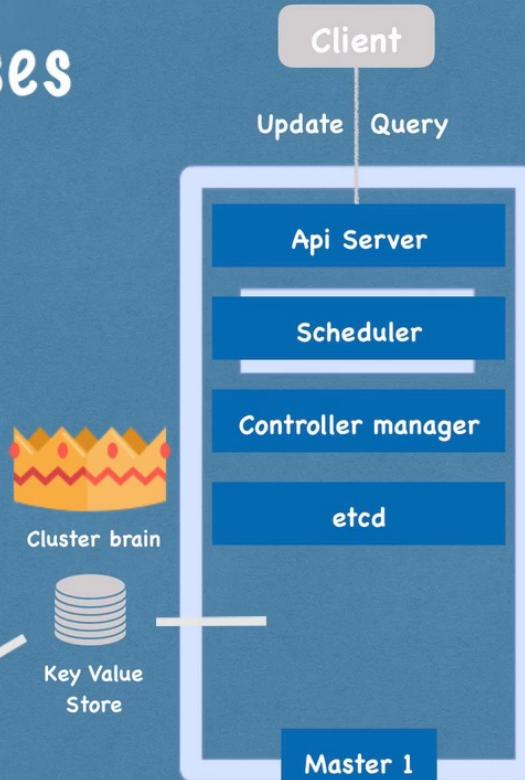
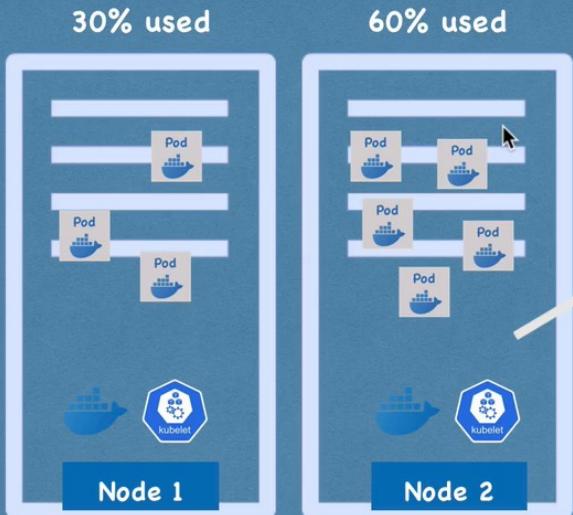
# Master processes



- ▶ etcd is the **cluster brain!**
- ▶ Cluster changes get stored in the key value store



# Master processes

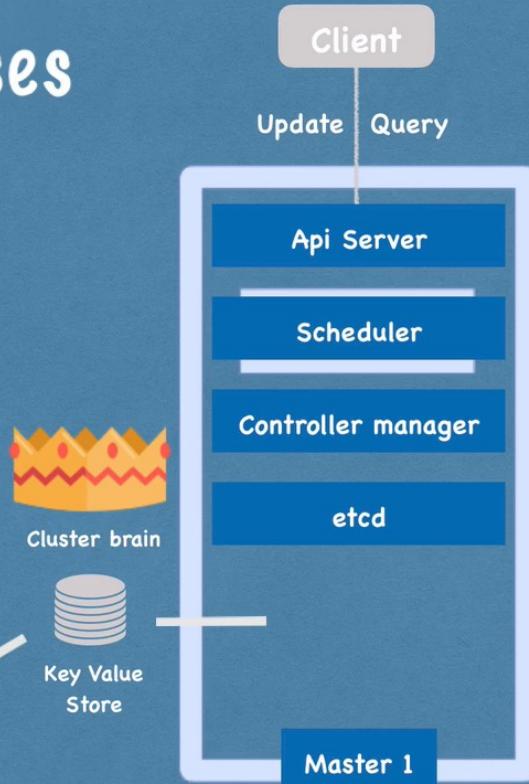
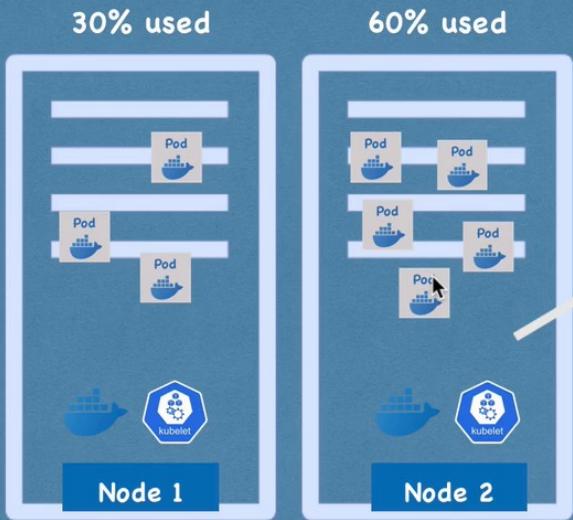


- ▶ Is the cluster healthy?
- ▶ What resources are available?
- ▶ Did the cluster state change?





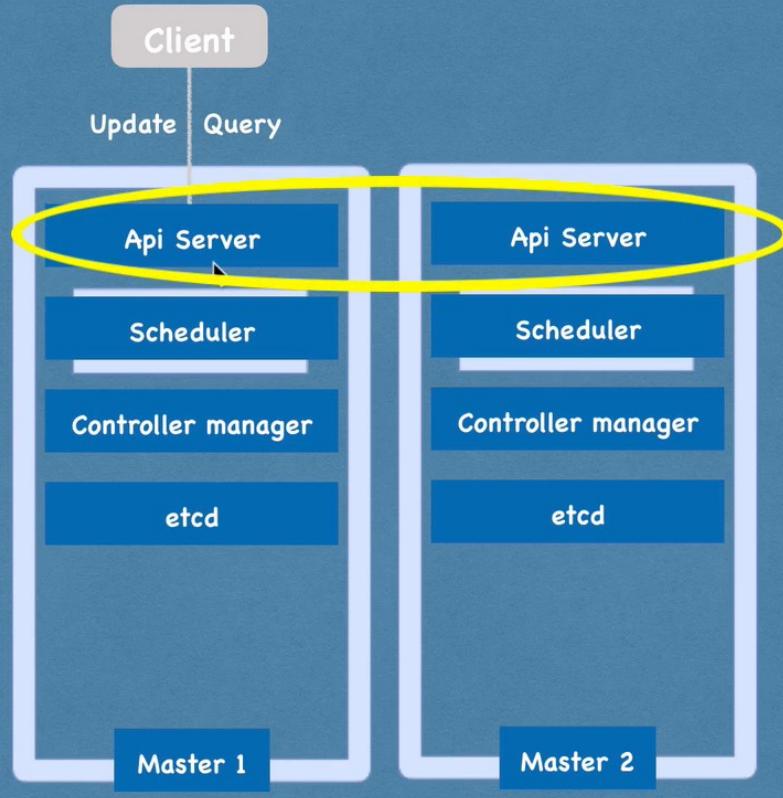
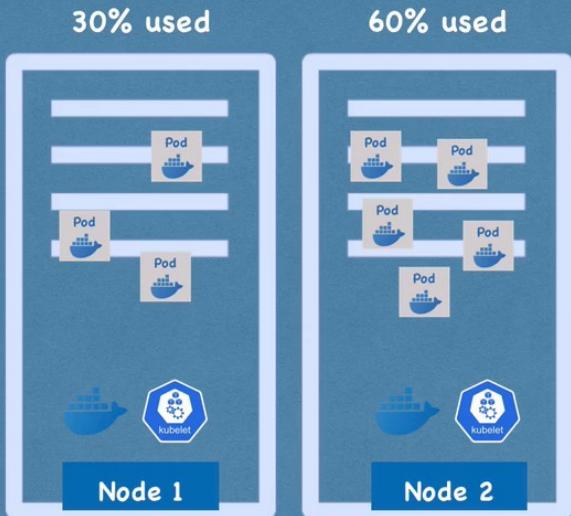
# Master processes



Application data is NOT stored in etcd!



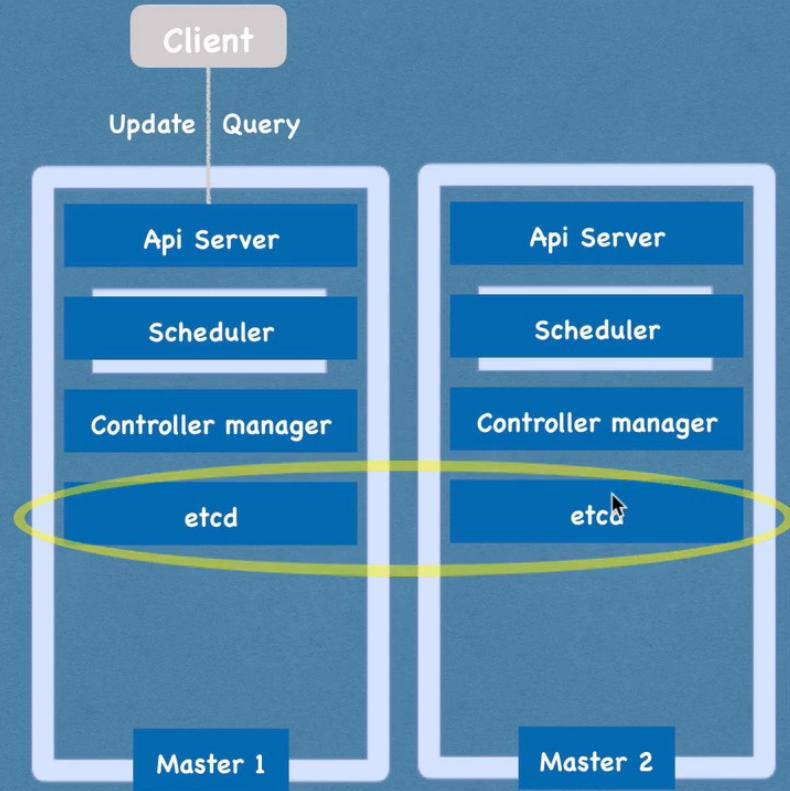
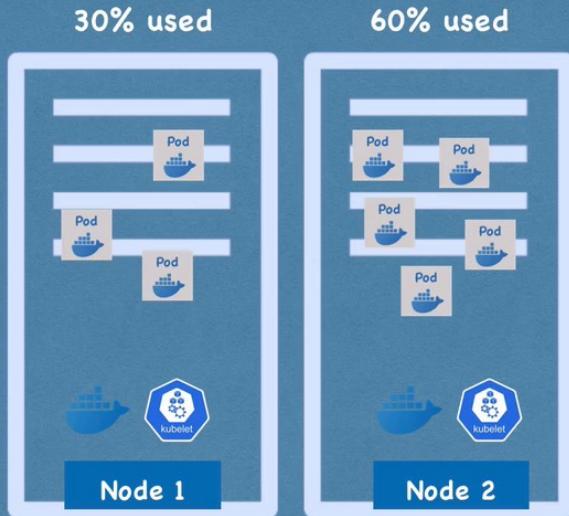
# Master processes



API server is load balanced.



# Master processes

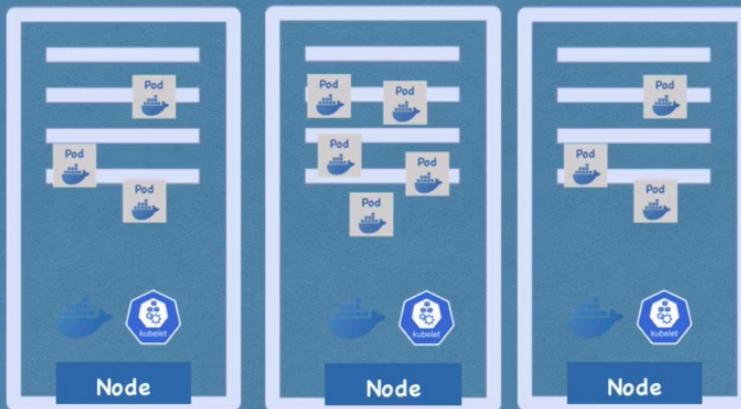
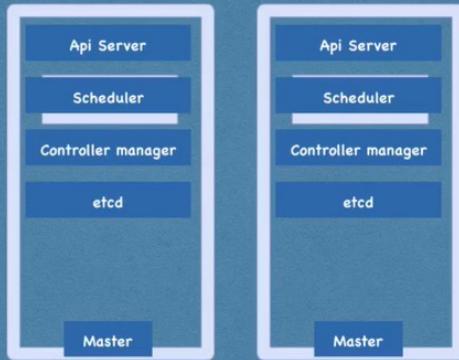


Distributed storage across all master nodes

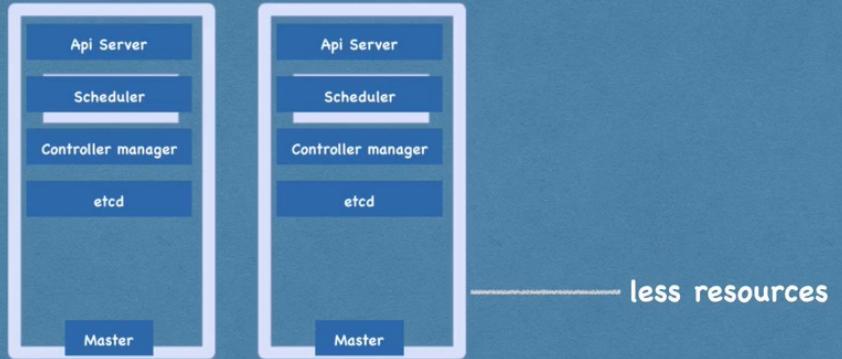
# Example Cluster Set-Up

2 Master Nodes

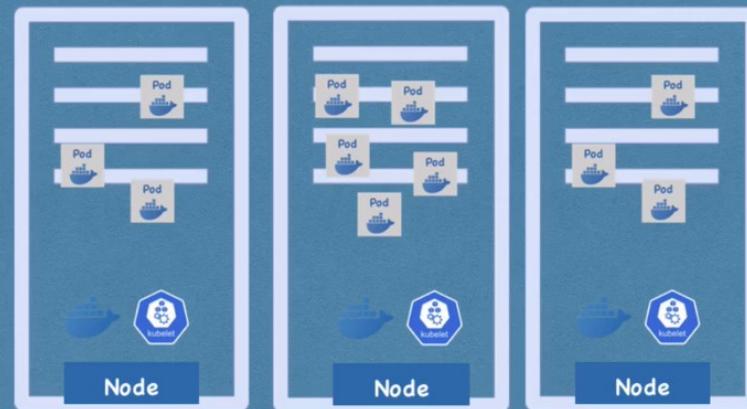
3 Worker Nodes



# Example Cluster Set-Up

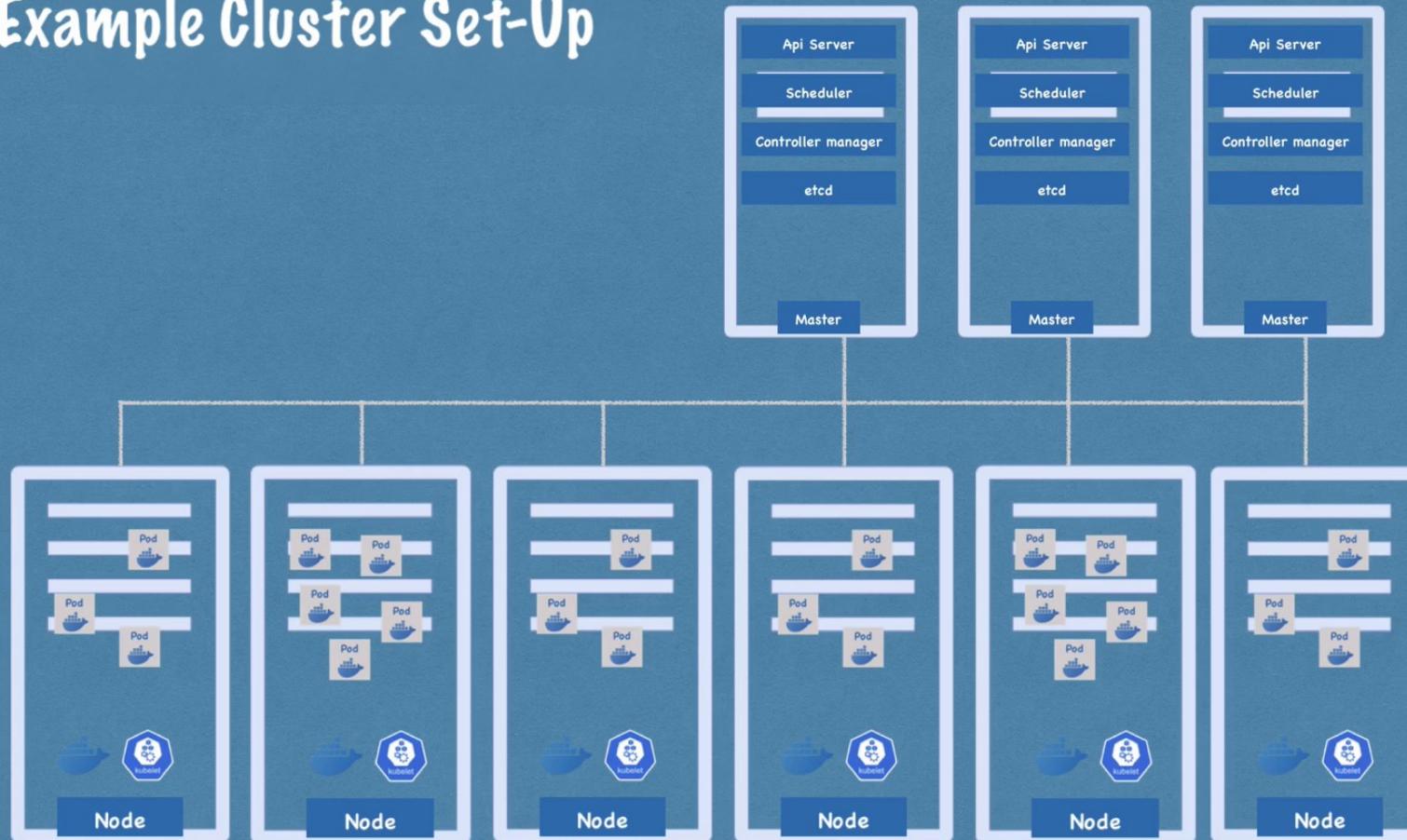


CPU | RAM | STORAGE



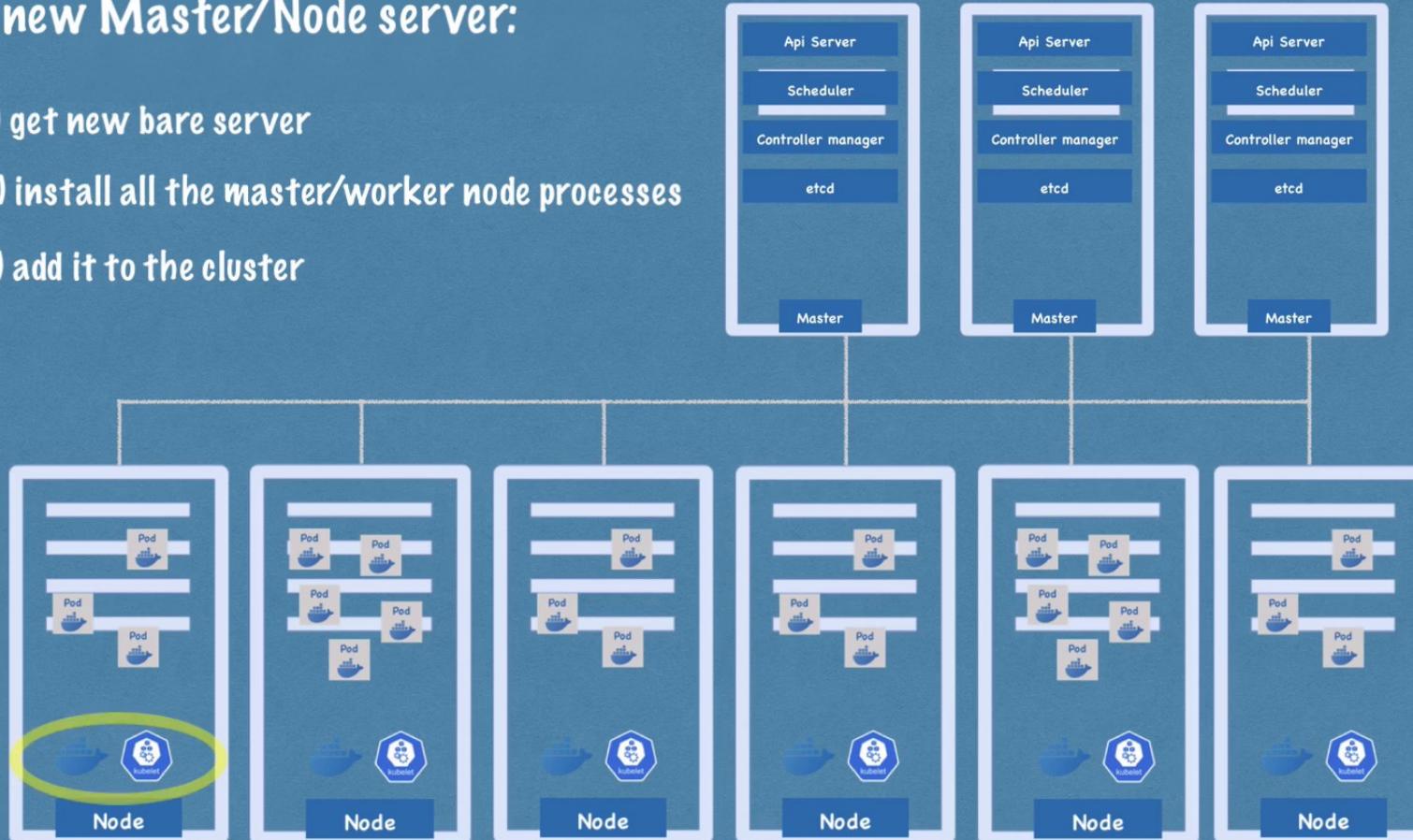
more resources

# Example Cluster Set-Up



# Add new Master/Node server:

- 1) get new bare server
- 2) install all the master/worker node processes
- 3) add it to the cluster



## Minikube and kubectl

- Local Setup





## Minikube and Kubectl

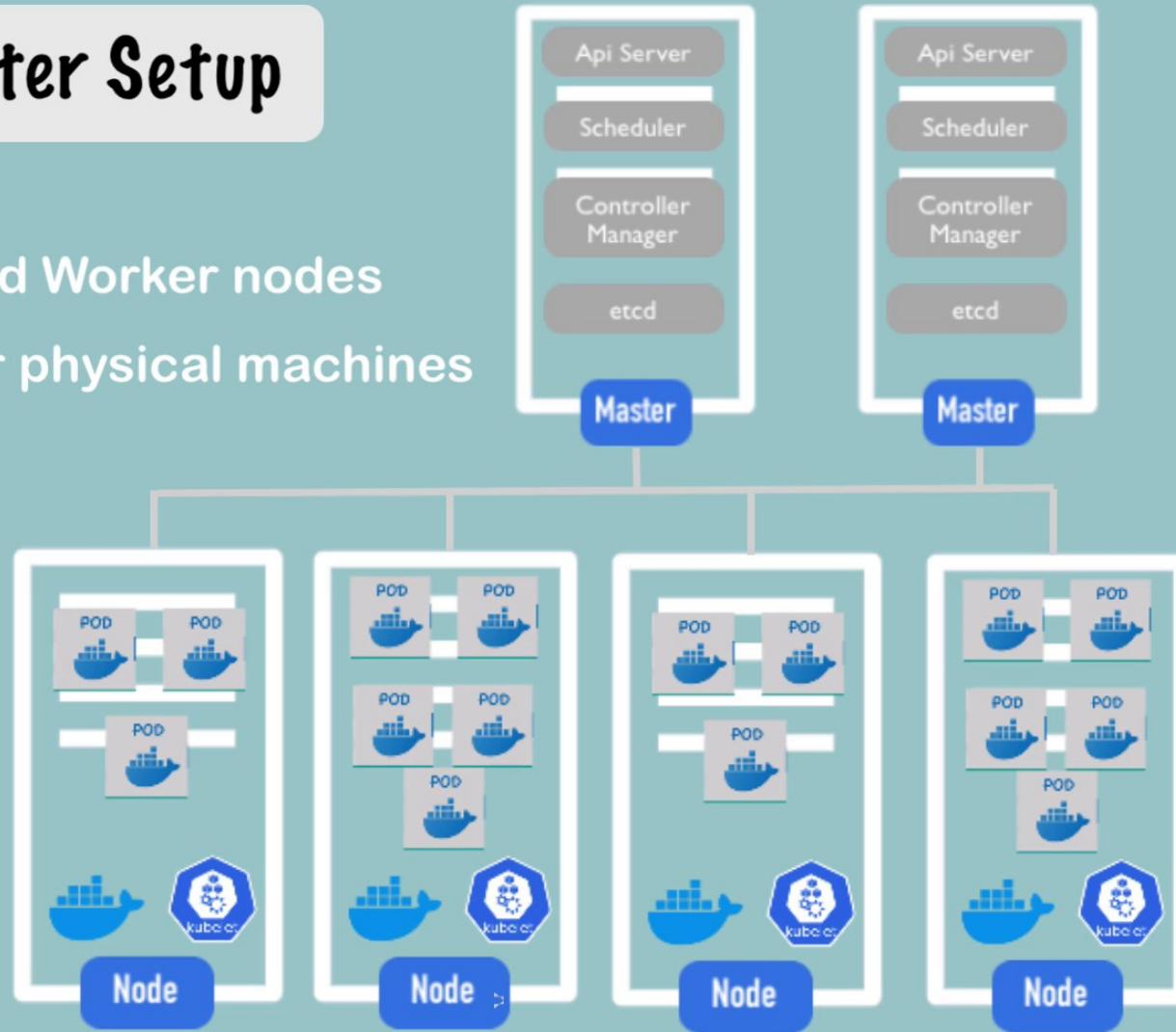
- ▶ What is minikube?
- ▶ What is kubectl?
- ▶ Set-up Minikube cluster



# What is minikube ?

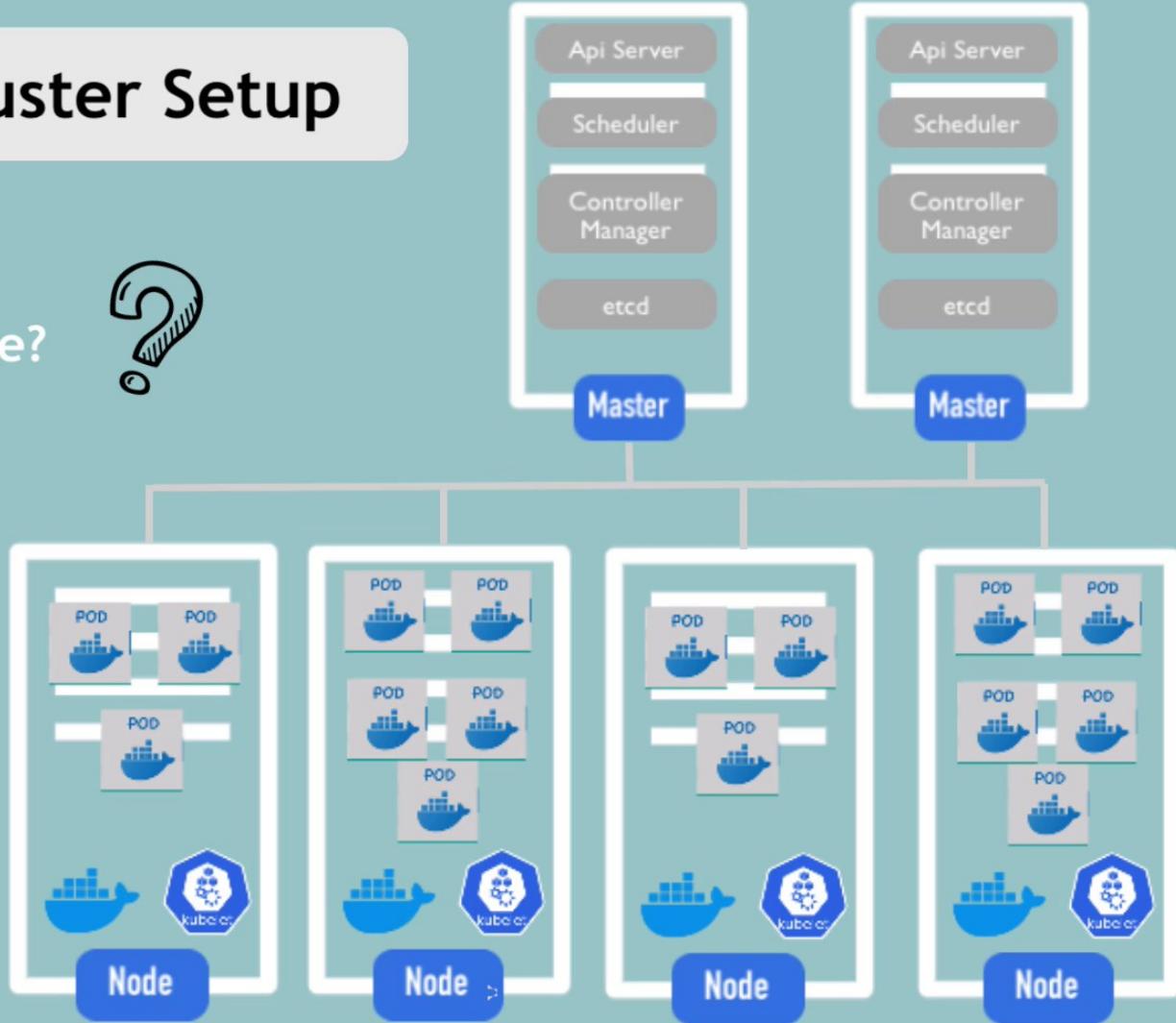
# Production Cluster Setup

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



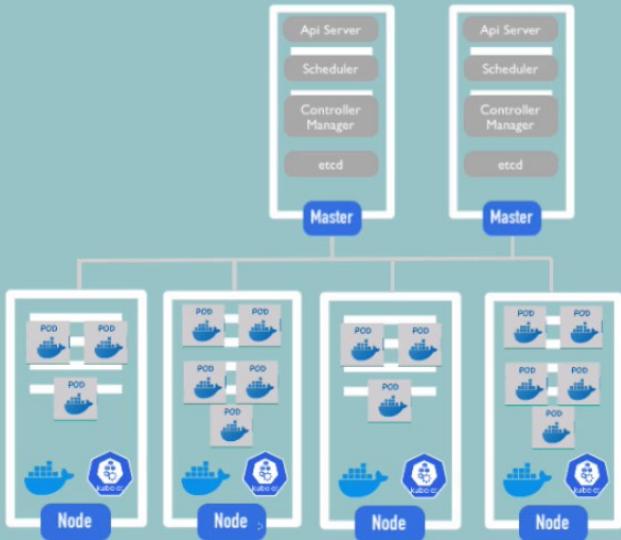
# Production Cluster Setup

Test on local machine?

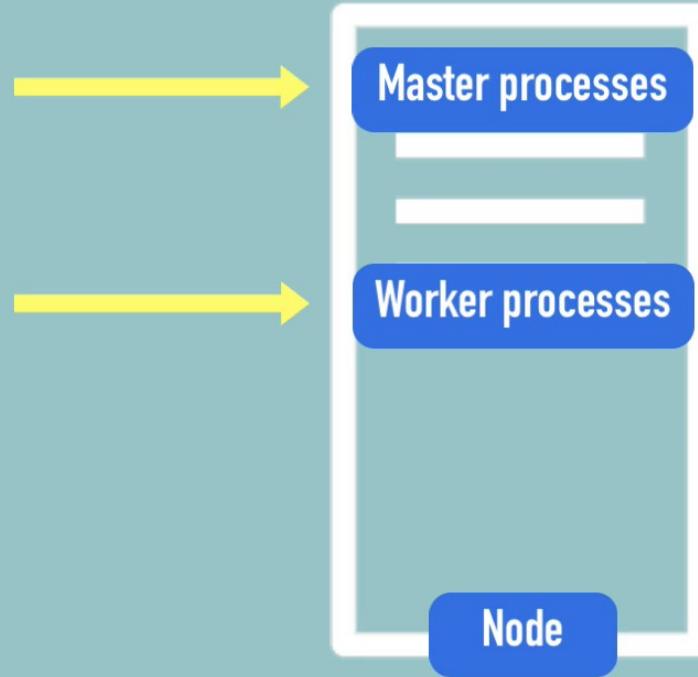




## Test/Local Cluster Setup

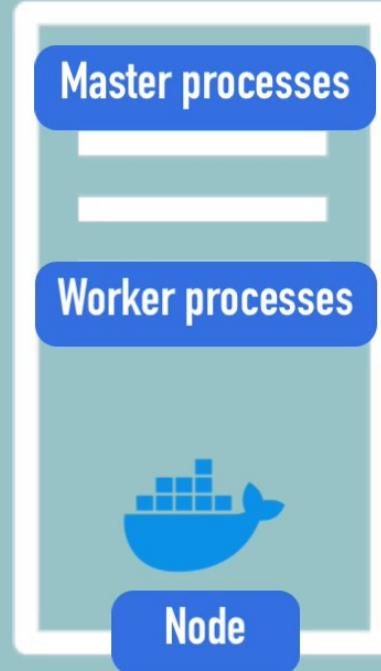
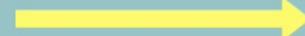
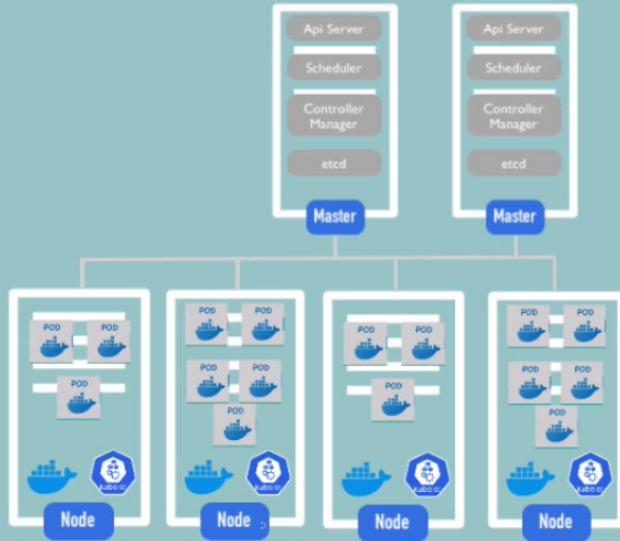


Master and Node processes  
run on **ONE** machine





## Test/Local Cluster Setup

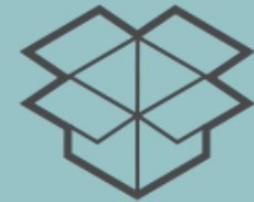
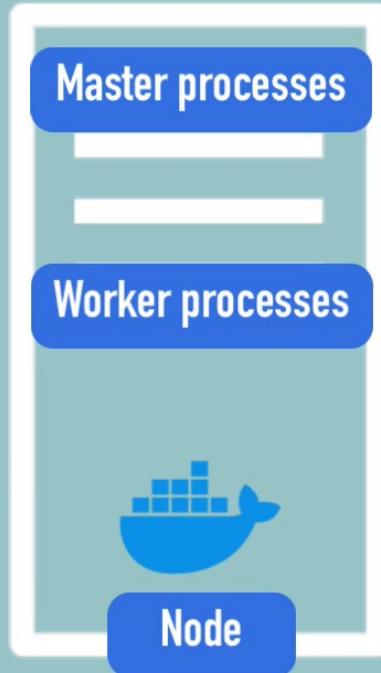


Docker pre-installed





## Test/Local Cluster Setup



Virtual Box

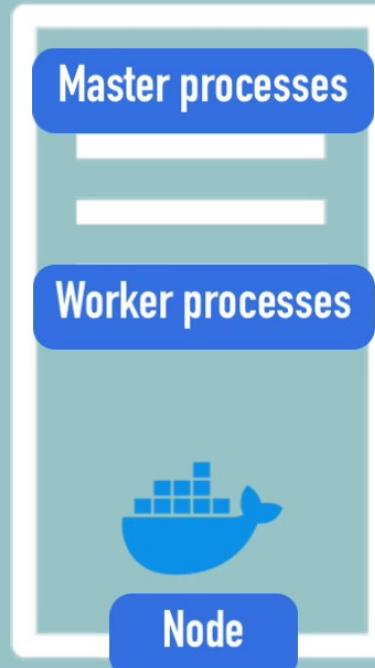




## Test/Local Cluster Setup



- ▶ creates Virtual Box on your laptop
- ▶ Node runs in that Virtual Box
- ▶ 1 Node K8s cluster
- ▶ for testing purposes



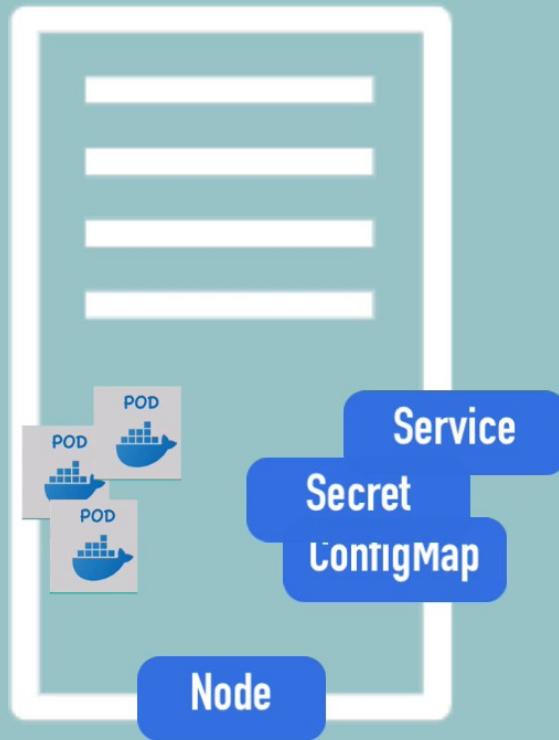
Virtual Box



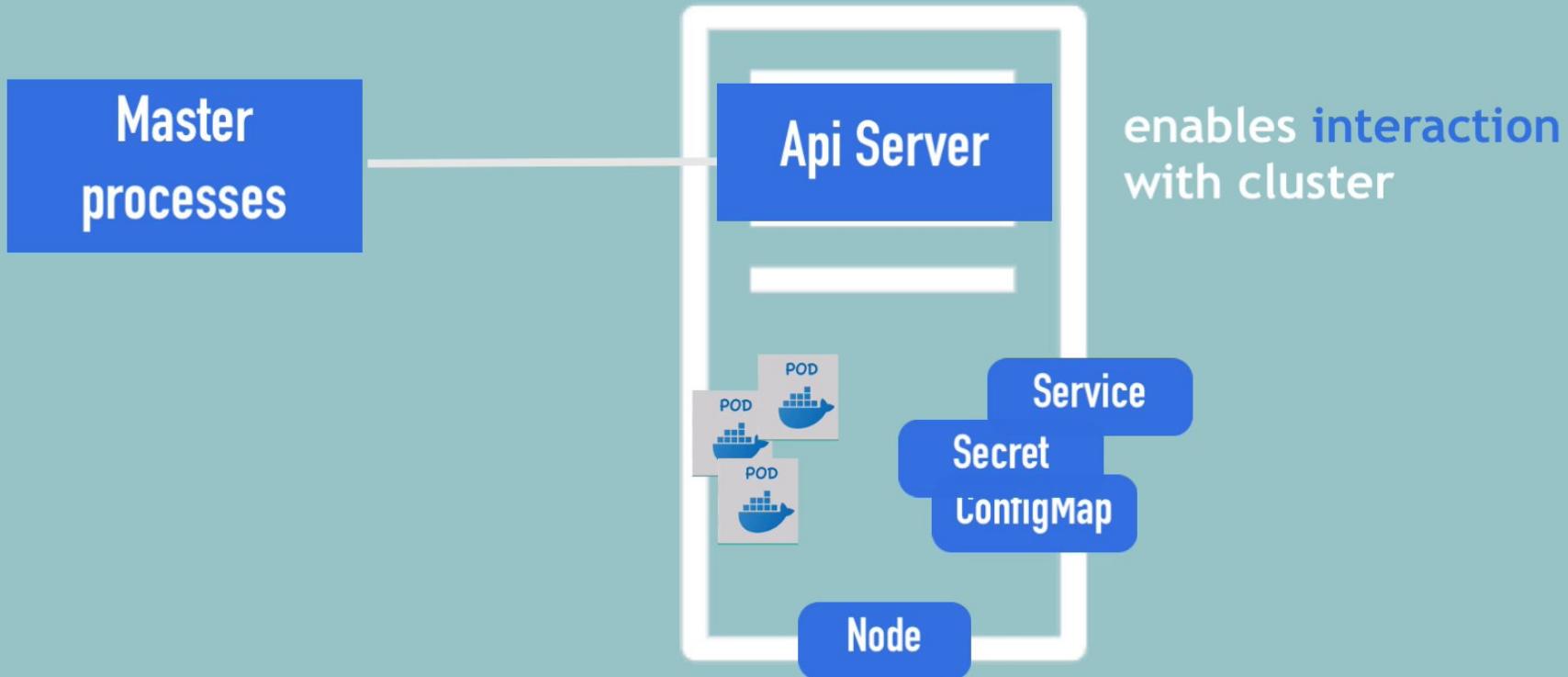
What is **kubectl** ?

# What is kubectl?

Command line tool for K8s cluster

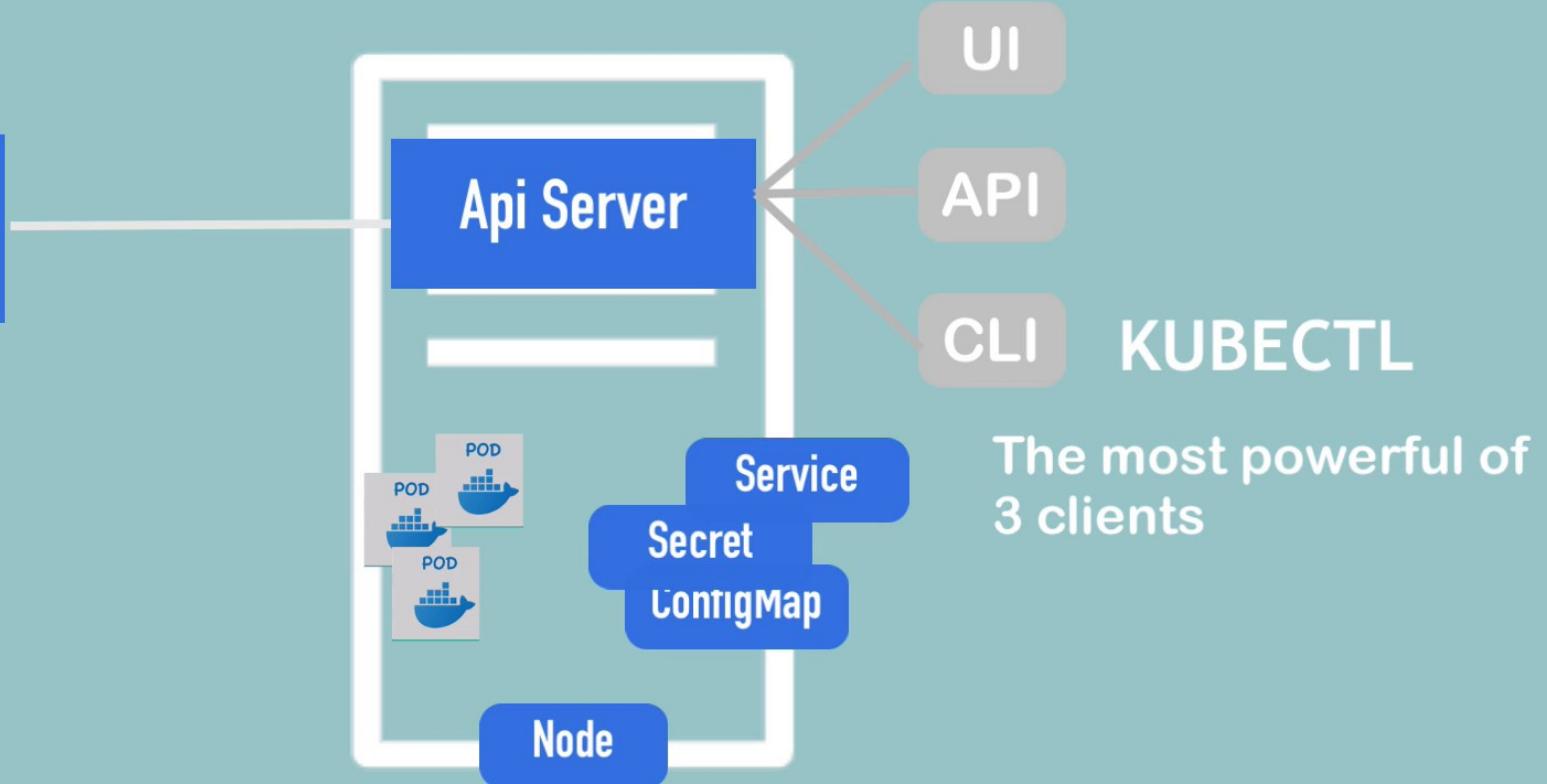


# What is kubectl?



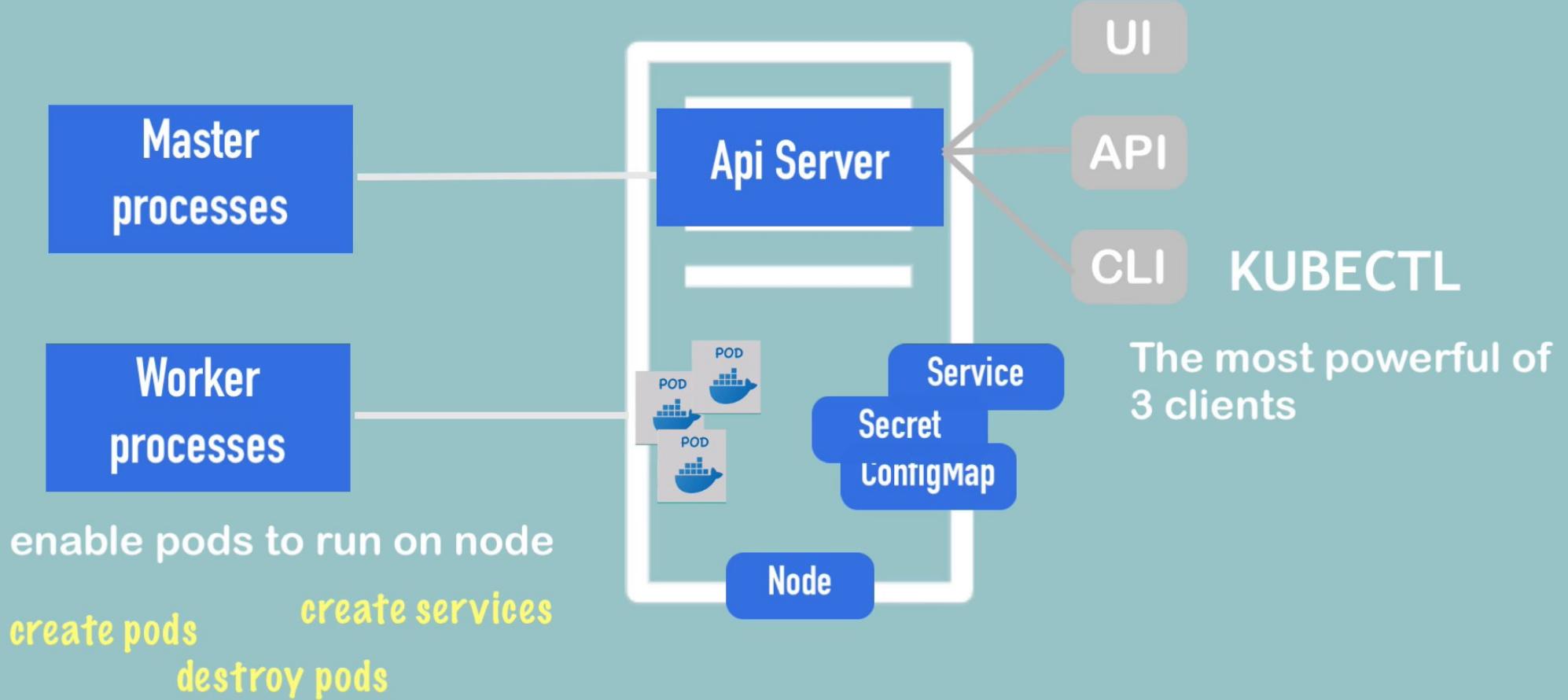
# What is kubectl?

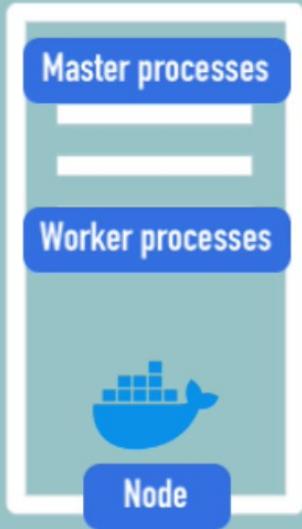
Master  
processes



The most powerful of  
3 clients

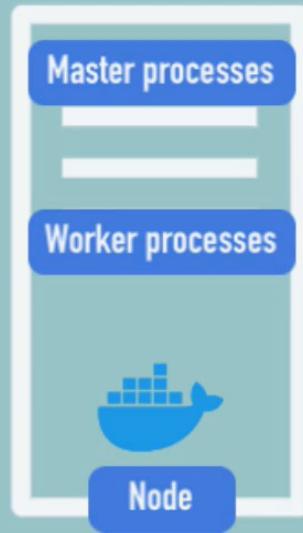
# What is kubectl?



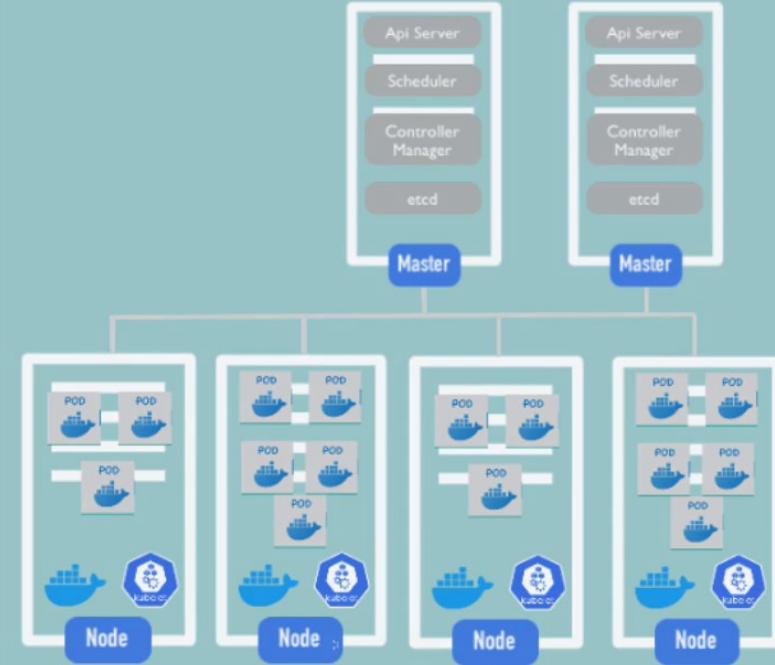


## Minikube cluster

KUBECTL



Minikube cluster



Cloud cluster

KUBECTL

# Installation

and

create



**minikube**

cluster



# Installation

Installation Guide  
Links for your OS:

**<https://kubernetes.io/docs/tasks/tools/install-minikube/>**

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



Screenshot of the Kubernetes documentation page for installing Minikube.

The URL in the address bar is <https://kubernetes.io/docs/tasks/tools/install-minikube/>.

The page title is "Install Minikube".

The main content area is titled "Tasks" and includes the following sections:

- ▼ Install Tools
  - ▶ Install and Set Up kubectl
  - ▶ Install Minikube** (highlighted)
  - ▶ Administer a Cluster
- ▶ Configure Pods and Containers
- ▶ Manage Kubernetes Objects
- ▶ Inject Data Into Applications
- ▶ Run Applications
- ▶ Run Jobs
- ▶ Access Applications in a Cluster
- ▶ Monitoring, Logging, and Debugging
- ▶ Extend Kubernetes
- ▶ TLS
- ▶ Federation
- ▶ Manage Cluster Daemons
- ▶ Install Service Catalog
- ▶ Network

The "Install Minikube" link is highlighted with a light gray background.

The "Before you begin" section is visible on the right side of the page.

At the bottom, there are three buttons for different operating systems: "Linux", "macOS", and "Windows".

A note at the bottom states: "To check if virtualization is supported on Linux, run the following command and verify that the output is non-empty:"



## Installation

**! Virtualization on your machine needed!**



# Installation

## Step 1: Install Hypervisor

### Installing minikube

Linux

macOS

Windows

#### Install kubectl

Make sure you have kubectl installed. You can install kubectl according to the instructions in [Install and Set Up kubectl](#).

#### Install a Hypervisor

If you do not already have a hypervisor installed, install one of these now:

- [KVM](#), which also uses QEMU
- [VirtualBox](#)

<https://nextgentips.com/2021/12/24/how-to-install-and-configure-minikube-on-ubuntu-21-10/>

wait Experimental: Wait for a specific condition on one or many resources.  
convert Convert config files between different API versions  
kustomize Build a kustomization target from a directory or a remote url.

#### Settings Commands:

label Update the labels on a resource  
annotate Update the annotations on a resource  
completion Output shell completion code for the specified shell (bash or zsh)

#### Other Commands:

api-resources Print the supported API resources on the server  
api-versions Print the supported API versions on the server, in the form of "group/version"  
config Modify kubeconfig files  
plugin Provides utilities for interacting with plugins.  
version Print the client and server version information

#### Usage:

kubectl [flags] [options]

Use "kubectl <command> --help" for more information about a given command.  
Use "kubectl options" for a list of global command-line options (applies to all

=> minikube

Bash completion has been installed to:  
/usr/local/etc/bash\_completion.d

zsh completions have been installed to:

/usr/local/share/zsh/site-functions

[~]\$ kubectl

kubectl controls the Kubernetes cluster manager.

Find more information at:

<https://kubernetes.io/docs/reference/kubectl/overview/>

Basic Commands (Beginner):

create	Create a resource from a file or from stdin.
expose	Take a replication controller, service, deployment or pod and expose it as a new Kubernetes Service
run	Run a particular image on the cluster
set	Set specific features on objects

Basic Commands (Intermediate):

explain	Documentation of resources
get	Display one or many resources
edit	Edit a resource on the server
delete	Delete resources by filenames, stdin, resources and names, or

Usage:

```
kubectl [flags] [options]
```

Use "kubectl <command> --help" for more information about a given command.

Use "kubectl options" for a list of global command-line options (applies to all commands).

```
[~]$ minikube
```

Minikube is a CLI tool that provisions and manages single-node Kubernetes clusters optimized for development workflows.

#### Basic Commands:

start	Starts a local kubernetes cluster
status	Gets the status of a local kubernetes cluster
stop	Stops a running local kubernetes cluster
delete	Deletes a local kubernetes cluster
dashboard	Access the kubernetes dashboard running within the minikube cluster

#### Images Commands:

docker-env	Sets up docker env variables; similar to '\$(docker-machine env)'
cache	Add or delete an image from the local cache.

#### Configuration and Management Commands:



Use "kubectl <command> --help" for more information about a given command.  
Use "kubectl options" for a list of global command-line options (applies to all commands).

[~]\$ minikube

Minikube is a CLI tool that provisions and manages single-node Kubernetes clusters optimized for development workflows.

#### Basic Commands:

start	Starts a local kubernetes cluster
status	Gets the status of a local kubernetes cluster
stop	Stops a running local kubernetes cluster
delete	Deletes a local kubernetes cluster
dashboard	Access the kubernetes dashboard running within the minikube cluster

#### Images Commands:

docker-env	Sets up docker env variables; similar to '\$(docker-machine env)'
cache	Add or delete an image from the local cache.

#### Configuration and Management Commands:

addons	Modify minikube's kubernetes addons
config	Modify minikube config
profile	Profile gets or sets the current minikube profile



```
[~]$ minikube start --vm-driver=hyperkit
    minikube v1.6.2 on Darwin 10.14.1
    Selecting 'hyperkit' driver from user configuration (alternates: [])
    Tip: Use 'minikube start -p <name>' to create a new cluster, or 'minikube delete' to
delete this one.
    Starting existing hyperkit VM for "minikube" ...
    Waiting for the host to be provisioned ...
    Preparing Kubernetes v1.17.0 on Docker '19.03.5' ...
    Launching Kubernetes ...
    Done! kubectl is now configured to use "minikube"
[~]$
```

Docker is pre-installed!

```
[~]$ minikube start --vm-driver=hyperkit
  minikube v1.6.2 on Darwin 10.14.1
  Selecting 'hyperkit' driver from user configuration (alternates: [])
  Tip: Use 'minikube start -p <name>' to create a new cluster, or 'minikube delete' to
delete this one.
  Starting existing hyperkit VM for "minikube" ...
  Waiting for the host to be provisioned ...
  Preparing Kubernetes v1.17.0 on Docker '19.03.5' ...
  Launching Kubernetes ...
  Done! kubectl is now configured to use "minikube"
[~]$ kubectl get nodes
NAME      STATUS    ROLES      AGE      VERSION
minikube  Ready     master     21h      v1.17.0
[~]$ minikube status
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
[~]$
```



```
>Selecting 'hyperkit' driver from user configuration (alternates: [])
Tip: Use 'minikube start -p <name>' to create a new cluster, or 'minikube delete' to
delete this one.
Starting existing hyperkit VM for "minikube" ...
Waiting for the host to be provisioned ...
Preparing Kubernetes v1.17.0 on Docker '19.03.5' ...
Launching Kubernetes ...
Done! kubectl is now configured to use "minikube"
[~]$ kubectl get nodes
NAME     STATUS   ROLES    AGE    VERSION
minikube  Ready    master   21h    v1.17.0
[~]$ minikube status
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
[~]$ kubectl version
Client Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.1", GitCommit:"d224
76cd0730bac2b0e357d144171ed74192d6", GitTreeState:"clean", BuildDate:"2020-01-15T15:50:
25Z", GoVersion:"go1.13.6", Compiler:"gc", Platform:"darwin/amd64"}
Server Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.0", GitCommit:"7013
2b0f130acc0bed193d9ba59dd186f0e634cf", GitTreeState:"clean", BuildDate:"2019-12-07T21:12:
17Z", GoVersion:"go1.13.4", Compiler:"gc", Platform:"linux/amd64"}
[~]$
```



>Selecting 'hyperkit' driver from user configuration (alternates: [])  
Tip: Use 'minikube start -p <name>' to create a new cluster, or 'minikube delete' to  
delete this one.

Starting existing hyperkit VM for "minikube" ...

Waiting for the host to be provisioned ...

Preparing Kubernetes v1.17.0 on Docker '19.03.5' ...

Launch

Done!

[~]\$ kubec

NAME

minikube

[~]\$ minikube status

host: Running

kubelet: R

apiserver:

kubeconfig

[~]\$ kubec

Client Ver

GitCommit:"d224  
476cd0730baca2b6e357d144171ed74192d6", GitTreeState:"clean", BuildDate:"2020-01-15T15:50:  
25Z", GoVersion:"go1.13.6", Compiler:"gc", Platform:"darwin/amd64"}

Server Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.0", GitCommit:"7013  
2b0f130acc0bed193d9ba59dd186f0e634cf", GitTreeState:"clean", BuildDate:"2019-12-07T21:12:  
17Z", GoVersion:"go1.13.4", Compiler:"gc", Platform:"linux/amd64"}

[~]\$

## Kubectl CLI

...for configuring the Minikube cluster

## Minikube CLI

...for start up/deleting the cluster

## Main Kubectl

## Commands





# Basic kubectl commands



# Basic kubectl commands

Create and debug Pods  
in a minikube cluster



## CRUD commands

[Create deployment](#)

`kubectl create deployment [name]`

[Edit deployment](#)

`kubectl edit deployment [name]`

[Delete deployment](#)

`kubectl delete deployment [name]`

## Status of different K8s components

`kubectl get nodes | pod | services | replicaset | deployment`

## Debugging pods

[Log to console](#)

`kubectl logs [pod name]`

[Get Interactive Terminal](#)

`kubectl exec -it [pod name] -- bin/bash`

**Get status of different components**

**Create and Edit a Pod**

[~]\$

~ -- bash — 89x24

# Pre-requisites:

- minikube installed
- kubectl installed

```
[~]$ kubectl get nodes
NAME      STATUS    ROLES      AGE      VERSION
minikube  Ready     master     25h      v1.17.0
[~]$
```

kubectl get nodes

```
[~]$ kubectl get pod  
No resources found in default namespace.  
[~]$ kubectl get services  
NAME          TYPE      CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE  
kubernetes   ClusterIP  10.96.0.1    <none>        443/TCP   25h  
[~]$
```

value

cronjob	Create a cronjob with the specified name.
deployment	Create a deployment with the specified name.
job	Create a job with the specified name.
namespace	Create a namespace with the specified name
poddisruptionbudget	Create a pod disruption budget with the specified name.
priorityclass	Create a priorityclass with the specified name.
quota	Create a quota with the spec
role	Create a role with single ru
rolebinding	Create a RoleBinding for a particular role or clusterrole
secret	Create a secret using specified subcommand
service	Create a service using specified subcommand.
serviceaccount	Create a service account with the specified name

**kubectl create ..**

Options:

--allow-missing-template-keys=true: If true, ignore any errors in templates when a field or map key is missing in the template. Only applies to golang and jsonpath output formats.

--dry-run=false: If true, only print the object that would be sent, without sending it.

--edit=false: Edit the API resource before creating

-f, --filename=[]: Filename, directory, or URL to files to use to create the resource

-k, --kustomize='': Process the kustomization directory. This flag can't be



# Pod is the smallest unit

BUT, you are creating...



## Deployment - abstraction over Pods

Usage:

```
kubectl create deployment NAME --image=image [--dry-run] [options]
```

--allow-missing-template-keys=true: If true, ignore any errors in templates when a field or map key is missing in the template. Only applies to golang and jsonpath output formats.

--dry-run=false: If true, only print the object that would be sent, without sending it.

--generator='': The name of the API generator to use.

--image=[]: Image name to run.

-o, --output='': Output format. One of: json|yaml|name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-file.

--save-config=false: If true, the configuration of current object will be saved in its annotation. Otherwise, the annotation will be unchanged. This flag is useful when you want to perform kubectl apply on this object in the future.

--template='': Template string or path to template file to use when -o=go-template, -o=go-template-file. The template format is golang templates [<http://golang.org/pkg/text/template/#pkg-overview>].

--validate=true: If true, use a schema to validate the input before sending it

Usage:

kubectl create deployment NAME --image=image [--dry-run] [options]

Use "kubectl options" for a list of global command-line options (applies to all commands)

.

```
[~]$ kubectl create deployment nginx-depl --image=nginx
```

want to perform kubectl apply on this object in the future.

--template='': Template string or path to template file to use when -o=go-template, -o=go-template-file. The template format is golang templates [<http://golang.org/pkg/text/template/#pkg-overview>].

--validate=true: If true, use a schema to validate the input before sending it

## Usage:

```
kubectl create deployment NAME --image=image [--dry-run] [options]
```

Use "kubectl options" for a list of global command-line options (applies to all commands)

.

```
[~]$ 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
```

```
[~]$ kubectl get pod
NAME                           READY   STATUS      RESTARTS   AGE
nginx-depl-7d9447675c-j9j8k  1/1    Running     0          54s
```

```
[~]$
```



```
kubectl create deployment nginx-depl --image=nginx
```

- blueprint for creating pods
- most basic configuration for deployment  
(name and image to use)
- rest defaults

```
/template/#pkg-overview].
```

```
--validate=true: If true, use a schema to validate the input before sending it
```

Usage:

```
kubectl create deployment NAME --image=image [--dry-run] [options]
```

Use "kubectl options" for a list of global command-line options (applies to all commands)

.

```
[~]$ 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

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-depl-7d9447675c-j9j8k	1/1	Running	0	54s

```
[~]$ kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
nginx-depl-7d9447675c	1	1	1	98s

```
[~]$
```

/template/#pkg-overview].

--validate=true: If true, use a schema to validate the input before sending it

Usage:

```
kubectl create deployment NAME --image=image [--dry-run] [options]
```

Use "kubectl options" for a list of global command-line options (applies to all commands)

.

```
[~]$ kubectl create deployment nginx-depl --image=nginx
```

```
deployment.apps/nginx-depl created
```

```
[~]$ kubectl get deployment
```

NAME	READY	UP-TO-DATE	AVAI
nginx-depl	0/1	1	0

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-depl-7d9447675c-j9j8k	0/1	ContainerCreating	0	31s

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-depl-7d9447675c-j9j8k	1/1	Running	0	54s

```
[~]$ kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
nginx-depl-7d9447675c	1	1	1	98s

```
[~]$
```

Replicaset is managing  
the replicas of a Pod

## Examples:

```
# Create a new deployment named my-dep that runs the busybox image.  
kubectl create deployment my-dep --image=busybox
```

## Options:

--allow-missing-template-keys=true: If true, ignore any errors in templates when a field or map key is missing in the template. Only applies to golang and jsonpath output formats.

--dry-run=false: If true, only print the object that would be sent, without sending it.

--generator='': The name of the API generator to use.

--image=[]: Image name to run.

-o, --output='': Output format. One of: json|yaml|name|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-file.

--save-config=false: If true, the configuration of current object will be saved in its annotation. Otherwise, the annotation will be unchanged. This flag is useful when you want to perform kubectl apply on this object in the future.

--template='': Template string or path to template file to use when -o=go-template, -o=go-template-file. The template format is golang templates [<http://golang.org/pkg/text/template/#pkg-overview>].

--validate=true: If true, use a schema to validate the input before sending it

## Usage:

# Layers of Abstraction



Deployment manages a ..



ReplicaSet manages a ..



Pod is an abstraction of ..



Container

# Layers of Abstraction

Everything below Deployment is handled by Kubernetes

```
[~]$ kubectl get deployment  
NAME READY UP-TO-DATE AVAILABLE AGE  
nginx-depl 1/1 1 1 12m
```

```
[~]$ kubectl get pod  
NAME READY STATUS RESTARTS AGE  
nginx-depl-7d9447675c-j9j8k 1/1 Running 0 12m
```

```
[~]$ kubectl get replicaset  
NAME DESIRED CURRENT READY AGE  
nginx-depl-7d9447675c 1 1 1 12m
```

```
[~]$ kubectl edit deployment nginx-
```

**kubectl edit deployment [name]**

```
# Please edit the object below. Lines beginning with a '#' will be ignored,  
# and an empty file will abort the edit. If an error occurs while saving this file will b  
e  
# reopened with the relevant failures.  
#  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  annotations:  
    deployment.kubernetes.io/revision: "1"  
  creationTimestamp: "2020-01-23T11:20:43Z"  
  generation: 3  
  labels:  
    app: nginx-depl  
  name: nginx-depl  
  namespace: default  
  resourceVersion: "55942"  
  selfLink: /apis/apps/v1/namespaces/default/deployments/nginx-depl  
  uid: e6bf6b5b-d56a-4a99-b85d-9c5a56c46113  
spec:  
  progressDeadlineSeconds: 600  
  replicas: 1  
  revisionHistoryLimit: 10  
"/var/folders/y3/bvgmrxg950x0f1z4zt3pby3c0000gn/T/kubectl-edit-808e9.yaml" 67L, 1866C
```

**Auto-generated configuration file  
with default values**

```
Terminal Shell Edit View Window Help ~ — vi ~ kubectl edit deployment nginx-depl — 89x24
creationTimestamp: null
labels:
  app: nginx-depl
spec:
  containers:
  - image: nginx:1.16█
    imagePullPolicy: Always
    name: nginx
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  terminationGracePeriodSeconds: 30
status:
  availableReplicas: 1
conditions:
- lastTransitionTime: "2020-01-23T11:20:43Z"
  lastUpdateTime: "2020-01-23T11:21:17Z"
  message: ReplicaSet "nginx-depl-7d9447675c" has successfully progressed.
  reason: NewReplicaSetAvailable
-- INSERT --
```

```
Terminal Shell Edit View Window Help ~ -- bash -- 89x24 33% [Thu 12:36 Nana Janashia] ⓘ
nginx-depl 1/1 1 1 12m
[~]$ kubectl get pod
NAME READY STATUS RESTARTS AGE
nginx-depl-7d9447675c-j9j8k 1/1 Running 0 12m
[~]$ kubectl get replicaset
NAME DESIRED CURRENT READY AGE
nginx-depl-7d9447675c 1 1 1 12m
[~]$ kubectl edit deployment nginx-depl
deployment.apps/nginx-depl edited
[~]$ kubectl get deployment
NAME READY UP-TO-DATE AVAILABLE AGE
nginx-depl 1/1 1 1 15m
[~]$ kubectl get pod
NAME READY STATUS RESTARTS AGE
nginx-depl-66859c8f65-vfjjk 1/1 Running 0 25s
nginx-depl-7d9447675c-j9j8k 0/1 Terminating 0 15m
[~]$ kubectl get pod
NAME READY STATUS RESTARTS AGE
nginx-depl-66859c8f65-vfjjk 1/1 Running 0 41s
[~]$ kubectl get replicaset
NAME DESIRED CURRENT READY AGE
nginx-depl-66859c8f65 1 1 1 59s
nginx-depl-7d9447675c 0 0 0 15m
[~]$
```



# Debugging pods

```
Terminal Shell Edit View Window Help 📦 ~ — bash — 89x24  
[~]$ kubectl get deployment  
NAME READY UP-TO-DATE AVAILABLE AGE  
nginx-depl 1/1 1 1 21m  
[~]$ kubectl get pod  
NAME READY STATUS RESTARTS AGE  
nginx-depl-66859c8f65-vfjjk 1/1 Running 0 6m29s  
[~]$ kubectl get replicaset  
NAME DESIRED CURRENT READY AGE  
nginx-depl-66859c8f65 1 1 1 6m32s  
nginx-depl-7d9447675c 0 0 0 21m  
[~]$ kubectl logs nginx-depl-66859c8f65-vfjjk  
[~]$
```

**kubectl logs [pod name]**

Terminal Shell Edit View Window Help

~ -- bash -- 89x24

NAME READY UP-TO-DATE AVAILABLE AGE  
nginx-depl 1/1 1 1 21m

[~]\$ kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	6m29s

[~]\$ kubectl get replicaset

NAME	DESIRED	CURRENT	READY	AGE
nginx-depl-66859c8f65	1	1	1	6m32s
nginx-depl-7d9447675c	0	0	0	21m

[~]\$ kubectl logs nginx-depl-66859c8f65-vfjjk

[~]\$ kubectl create deployment mongo-depl --image=mongo

deployment.apps/mongo-depl created

[~]\$ kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	0/1	ContainerCreating	0	6s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	8m16s

[~]\$ kubectl logs mongo-depl-67f895857c-fkspm

Error from server (BadRequest): container "mongo" in pod "mongo-depl-67f895857c-fkspm" is waiting to start: ContainerCreating

[~]\$ kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	0/1	ContainerCreating	0	35s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	8m45s

[~]\$

NAME	DESIRED	CURRENT	READY	AGE
nginx-depl-66859c8f65	1	1	1	6m32s
nginx-depl-7d9447675c	0	0	0	21m

```
[~]$ kubectl logs nginx-depl-66859c8f65-vfjjk
```

```
[~]$ kubectl create deployment mongo-depl --image=mongo  
deployment.apps/mongo-depl created
```

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	0/1	ContainerCreating	0	3s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	8m16s

```
[~]$ kubectl logs mongo-depl-67f895857c-fkspm
```

```
Error from server (BadRequest): container "mongo" in pod "mongo-depl-67f895857c-fkspm" is  
waiting to start: ContainerCreating
```

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	0/1	ContainerCreating	0	35s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	8m45s

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	0/1	ContainerCreating	0	76s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	9m26s

```
[~]$ kubectl describe mongo-depl-67f895857c-fkspm
```

```
error: the server doesn't have a resource type "mongo-depl-67f895857c-fkspm"
```

```
[~]$ kubectl describe pod mongo-depl-67f895857c-fkspm
```

## kubectl describe pod [pod name]

Type Status  
Initialized True  
Ready True  
ContainersReady True  
PodScheduled True

## Volumes:

default-token-z2jgc:

Type: Secret (a volume populated by a Secret)  
SecretName: default-token-z2jgc  
Optional: false  
QoS Class: BestEffort  
Node-Selectors: <none>  
Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s  
node.kubernetes.io/unreachable:NoExecute for 300s

## Events:

Type	Reason	Age	From	Message
Normal	Scheduled	<unknown>	default-scheduler	Successfully assigned default/mongo-de
Normal	Pulling	105s	kubelet, minikube	Pulling image "mongo"
Normal	Pulled	13s	kubelet, minikube	Successfully pulled image "mongo"
Normal	Created	13s	kubelet, minikube	Created container mongo
Normal	Started	13s	kubelet, minikube	Started container mongo

PodScheduled True

Volumes:

default-token-z2jgc:

Type: Secret (a volume populated by a Secret)

SecretName: default-token-z2jgc

Optional: false

QoS Class: BestEffort

Node-Selectors: <none>

Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s

node.kubernetes.io/unreachable:NoExecute for 300s

Events:

Type	Reason	Age	From	Message
Normal	Scheduled	<unknown>	default-scheduler	Successfully assigned default/mongo-depl-67f895857c-fkspm to minikube
Normal	Pulling	105s	kubelet, minikube	Pulling image "mongo"
Normal	Pulled	13s	kubelet, minikube	Successfully pulled image "mongo"
Normal	Created	13s	kubelet, minikube	Created container mongo
Normal	Started	13s	kubelet, minikube	Started container mongo

[~]\$ kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	1/1	Running	0	2m12s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	10m

[~]\$ kubectl logs mongo-depl-67f895857c-fkspm

```
[~]$ kubectl logs mongo-depl-67f895857c-fkspm
2020-01-23T11:45:15.008+0000 I CONTROL [main] Automatically disabling TLS 1.0, to force
-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-01-23T11:45:15.015+0000 I CONTROL [initandlisten] MongoDB starting : pid=1 port=27
017 dbpath=/data/db 64-bit host=mongo-depl-67f895857c-fkspm
2020-01-23T11:45:15.015+0000 I CONTROL [initandlisten] db version v4.2.2
2020-01-23T11:45:15.015+0000 I CONTROL [initandlisten] git version: a0bbbff6ada159e1929
8d37946ac8dc4b497eadf
2020-01-23T11:45:15.015+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.1.1
11 Sep 2018
2020-01-23T11:45:15.016+0000 I CONTROL [initandlisten] allocator: tcmalloc
2020-01-23T11:45:15.016+0000 I CONTROL [initandlisten] modules: none
2020-01-23T11:45:15.016+0000 I CONTROL [initandlisten] build environment:
2020-01-23T11:45:15.016+0000 I CONTROL [initandlisten] distmod: ubuntu1804
2020-01-23T11:45:15.016+0000 I CONTROL [initandlisten] distarch: x86_64
2020-01-23T11:45:15.016+0000 I CONTROL [initandlisten] target_arch: x86_64
2020-01-23T11:45:15.016+0000 I CONTROL [initandlisten] options: { net: { bindIp: "*" } }
}
2020-01-23T11:45:15.016+0000 I STORAGE [initandlisten]
2020-01-23T11:45:15.017+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS file system
is strongly recommended with the WiredTiger storage engine
2020-01-23T11:45:15.017+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-01-23T11:45:15.017+0000 I STORAGE [initandlisten] wiredtiger_open config: create,c
```

```
Terminal Shell Edit View Window Help root@mongo-depl-67f895857c-fkspm: / — kubectl exec -it mongo-depl-67f895857c-fkspm -- bin/bash — 89x24  
[~]$ kubectl get pod  
NAME READY STATUS RESTARTS AGE  
mongo-depl-67f895857c-fkspm 1/1 Running 0 3m5s  
nginx-depl-66859c8f65-vfjjk 1/1 Running 0 11m  
[~]$ kubectl exec -it mongo-depl-67f895857c-fkspm -- bin/bash  
root@mongo-depl-67f895857c-fkspm:/#
```

kubectl exec -it [pod name] -- bin/bash

[~]\$ kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	1/1	Running	0	3m5s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	11m

[~]\$ kubectl exec -it mongo-depl-67f895857c-fkspm -- bin/bash

root@mongo-depl-67f895857c-fkspm:/# ls

bin	dev	home	lib64	opt	run	sys	var
boot	docker-entrypoint-initdb.d	js-yaml.js	media	proc	sbin	tmp	
data	etc	lib	mnt	root	srv	usr	

root@mongo-depl-67f895857c-fkspm:/# exit

exit

[~]\$

- Delete deployment
- Apply configuration file

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	1/1	Running	0	3m5s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	11m

```
[~]$ kubectl exec -it mongo-depl-67f895857c-fkspm -- bin/bash
```

```
[root@mongo-depl-67f895857c-fkspm:/# ls
```

```
bin dev home lib64 opt run sys var  
boot docker-entrypoint-initdb.d js-y lib  
data etc
```

```
[root@mongo-depl-67f895857c-fkspm:/# exit
```

```
exit
```

```
[~]$ kubectl get deplyoment
```

```
error: the server doesn't have a resource type "deplyoment"
```

```
[~]$ kubectl get deployment
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
mongo-depl	1/1	1	1	5m55s
nginx-depl	1/1	1	1	28m

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	1/1	Running	0	6m
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	14m

```
[~]$ kubectl delete
```

kubectl delete deployment [name]

```
Terminal Shell Edit View Window Help 📦 ~ -- bash -- 89x24 44% [⚡] Thu 12:50 Nana Janashia ⚡ ⌂
error: the server doesn't have a resource type "deployment"
[~]$ kubectl get deployment
NAME READY UP-TO-DATE AVAILABLE AGE
mongo-depl 1/1 1 1 5m55s
nginx-depl 1/1 1 1 28m
[~]$ kubectl get pod
NAME READY STATUS RESTARTS AGE
mongo-depl-67f895857c-fkspm 1/1 Running 0 6m
nginx-depl-66859c8f65-vfjjk 1/1 Running 0 14m
[~]$ kubectl delete deployment mongo-depl
deployment.apps "mongo-depl" deleted
[~]$ kubectl get pod
NAME READY STATUS RESTARTS AGE
mongo-depl-67f895857c-fkspm 0/1 Terminating 0 6m29s
nginx-depl-66859c8f65-vfjjk 1/1 Running 0 14m
[~]$ kubectl get replicaset
NAME DESIRED CURRENT READY AGE
nginx-depl-66859c8f65 1 1 1 14m
nginx-depl-7d9447675c 0 0 0 29m
[~]$ kubectl delete deployment nginx-depl
deployment.apps "nginx-depl" deleted
[~]$ kubectl get replicaset
No resources found in default namespace.
[~]$
```

```
error: the server doesn't have a resource type "deployment"
```

```
[~]$ kubectl get deployment
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
mongo-depl	1/1	1	1	5m55s
nginx-depl	1/1	1	1	28m

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	1/1	Running	0	6m
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	14m

```
[~]$ kubectl delete deployment mongo-depl
```

```
deployment.apps "mongo-depl" deleted
```

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	0/1	Terminating	0	6m29s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	14m

```
[~]$ kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
nginx-depl-66859c8f65	1	1	1	14m
nginx-depl-7d9447675c	0	0	0	29m

```
[~]$ kubectl delete deployment nginx-depl
```

```
deployment.apps "nginx-depl" deleted
```

```
[~]$ kubectl get replicaset
```

```
No resources found in default namespace.
```

```
[~]$ kubectl create deployment name image option1 option2
```

```
error: the server doesn't have a resource type "deployment"
```

```
[~]$ kubectl get deployment
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
mongo-depl	1/1	1	1	5m55s
nginx-depl	1/1	1	1	28m

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	1/1	Running	0	6m29s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	14m

```
[~]$ kubectl delete deployment mongo-depl
```

```
deployment.apps "mongo-depl" deleted
```

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	0/1	Terminating	0	6m29s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	14m

```
[~]$ kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
nginx-depl-66859c8f65	1	1	1	14m
nginx-depl-7d9447675c	0	0	0	29m

```
[~]$ kubectl delete deployment nginx-depl
```

```
deployment.apps "nginx-depl" deleted
```

```
[~]$ kubectl get replicaset
```

```
No resources found in default namespace.
```

```
[~]$ kubectl apply -f
```

**kubectl apply -f [file name]**

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
mongo-depl	1/1	1	1	5m55s
nginx-depl	1/1	1	1	28m

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	1/1	Running	0	6m
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	14m

```
[~]$ kubectl delete deployment mongo-depl
```

```
deployment.apps "mongo-depl" deleted
```

```
[~]$ kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-depl-67f895857c-fkspm	0/1	Terminating	0	6m29s
nginx-depl-66859c8f65-vfjjk	1/1	Running	0	14m

```
[~]$ kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
nginx-depl-66859c8f65	1	1	1	14m
nginx-depl-7d9447675c	0	0	0	29m

```
[~]$ kubectl delete deployment nginx-depl
```

```
deployment.apps "nginx-depl" deleted
```

```
[~]$ kubectl get replicaset
```

```
No resources found in default namespace.
```

```
[~]$ #kubectl apply -f nginx-deployment.yaml
```

```
[~]$ touch nginx-deployment.yaml
```

```
[~]$ vim 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
```

~  
~  
:wq

```
[~]$ vim 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
[~]$ vim nginx-deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.16
          ports:
            - containerPort: 80
```

~  
~

-- INSERT --

```
[~]$ vim 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
[~]$ vim nginx-deployment.yaml
[~]$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment configured
[~]$
```

K8s knows when to create or update deployment

```
[~]$ vim 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
[~]$ vim nginx-deployment.yaml
[~]$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment configured
[~]$ kubectl get deployment
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2      2           2          115s
[~]$ kubectl get pod
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-594cc45b78-ncs97   1/1     Running   0          39s
nginx-deployment-594cc45b78-pq5dx   1/1     Running   0          2m5s
[~]$
```

# Summarize kubectl commands

## CRUD commands

Create deployment

`kubectl create deployment [name]`

Edit deployment

`kubectl edit deployment [name]`

Delete deployment

`kubectl delete deployment [name]`

## Status of different K8s components

`kubectl get nodes | pod | services | replicaset | deployment`

## Debugging pods

Log to console

`kubectl logs [pod name]`

Get Interactive Terminal

`kubectl exec -it [pod name] -- bin/bash`

## Debugging pods

[Log to console](#)

[Get Interactive Terminal](#)

[Get info about pod](#)

`kubectl logs [pod name]`

`kubectl exec -it [pod name] -- bin/bash`

`kubectl describe pod [pod name]`

## Use configuration file for CRUD

[Apply a configuration file](#)

[Delete with configuration file](#)

`kubectl apply -f [file name]`

`kubectl delete -f [file name]`



# YAML Configuration File in Kubernetes



# YAML Configuration File in Kubernetes

## Overview:



- The 3 parts of configuration file
- Connecting Deployments to Service to Pods
- Demo



# YAML Configuration File in Kubernetes

## Overview:

- The 3 parts of configuration file
- Connecting Deployments to Service to Pods
- Demo

```
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
```

**3 PARTS**  
of a K8s configuration file

! nginx-deployment.yaml x

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

22

...

! nginx-service.yaml x

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

12

...



# Each configuration file has 3 parts

## I) metadata

### Deployment

```
! nginx-deployment.yaml ✘  
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    +  
11    +  
12    +  
13    +  
14    +  
15    +  
16    +  
17    +  
18    +  
19    +  
20    +  
21    +  
22
```

### Service

```
! nginx-service.yaml ✘  
1   apiVersion: v1  
2   kind: Service  
3   metadata:  
4     name: nginx-service  
5   spec:  
6     selector: ...  
7     ports: ...  
8     +  
9     +  
10    +  
11    +  
12    +  
13    +  
14    +  
15    +  
16    +  
17    +  
18    +  
19    +  
20    +  
21    +  
22
```

# Each configuration file has 3 parts

## Deployment

```
! nginx-deployment.yaml ✘  
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      +  
11      +  
12      +  
13      +  
14      +  
15      +  
16      +  
17      +  
18      +  
19      +  
20      +  
21      +  
22      +
```

## 1) metadata

## 2) specification

## Service

```
! nginx-service.yaml ✘  
1   apiVersion: v1  
2   kind: Service  
3   metadata:  
4     name: nginx-service  
5     + spec:  
6       selector: ...  
7       + ports: ...  
8       +  
9       +  
10      +  
11      +  
12      +  
13      +  
14      +  
15      +  
16      +  
17      +  
18      +  
19      +  
20      +  
21      +  
22      +
```

# Each configuration file has 3 parts

Deployment

! nginx-deployment.yaml ×

```
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
11
12
```

1) metadata

2) specification

Service

! nginx-service.yaml ×

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: nginx-service
5  spec:
6    selector: ...
7    ports: ...
8
9
10
11
12
```

```
! nginx-deployment.yaml ✘
```

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5  +  labels: ...
6
7  spec:
8    replicas: 2
9    selector: ...
10   +  template: ...
```

```
! nginx-service.yaml ✘
```

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

Attributes of "spec" are specific to the **kind**!

Each configuration file has **3** parts

- 1) metadata
- 2) specification
- 3) status

Automatically generated and added by Kubernetes!

```
! nginx-deployment.yaml ✘

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5    labels: ...
6
7  spec:
8    replicas: 2
9    selector: ...
10   template: ...
```

## 3rd part: status

Desired?  Actual?

```
! nginx-deployment.yaml ✘

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5  +  labels: ...
6
7  spec:
8    replicas: 2
9  +  selector: ...
10 +  template: ...
```

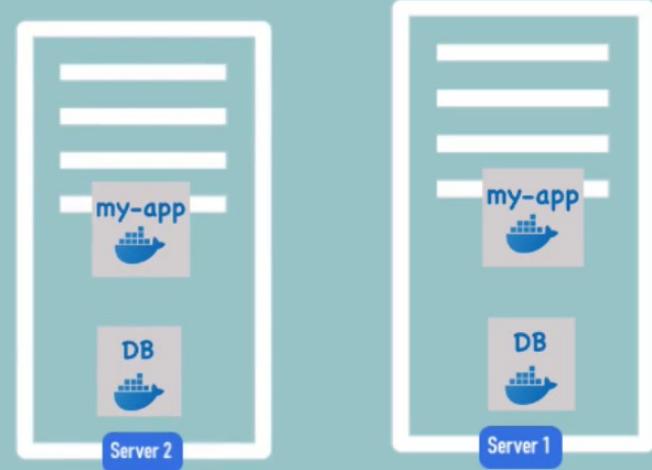
## 3rd part: status

Desired?   =   Actual?

```
! nginx-deployment.yaml ✘
1   apiVersion: apps/v1
2   kind: Deployment
3   metadata:
4     name: nginx-deployment
5     labels: ...
6
7   spec:
8     replicas: 2
9
10  selector: ...
11
12  template: ...
13
14
15
16
17
18
19
20
21
22
status:
  availableReplicas: 1
  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-7d64f4b...
      reason: MinimumReplicasAvailable
      status: "True"
      type: Available
```

# K8s updates state continuously!

```
status:  
  availableReplicas: 1  
  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-7d64f4b"  
    reason: NewReplicaSetAvailable  
    status: "True"  
    type: Progressing  
  observedGeneration: 1  
  readyReplicas: 1  
  replicas: 1  
  updatedReplicas: 1
```



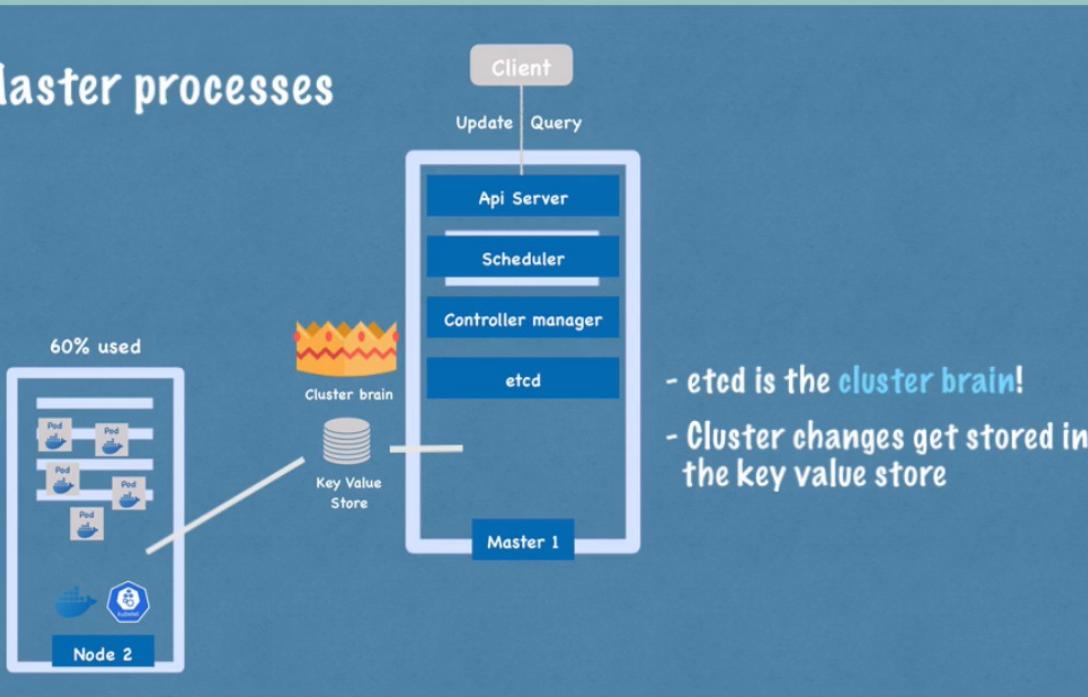
```
! nginx-deployment.yaml ✘  
1  apiVersion: apps/v1  
2  kind: Deployment  
3  metadata:  
4    name: nginx-deployment  
5    labels: ...  
7  spec:  
8    replicas: 2  
9    selector: ...  
12   template: ...  
22
```

# Where does K8s get this status data?

```
status:  
  availableReplicas: 1  
  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-7d64f4b"  
      reason: NewReplicaSetAvailable  
      status: "True"  
      type: Progressing  
  observedGeneration: 1  
  readyReplicas: 1  
  replicas: 1  
  updatedReplicas: 1
```

# Where does K8s get this status data?

Master processes



Etcd holds the current status of any K8s component!

# Format of configuration file



# YAML configuration files

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

- "human friendly data serialization standard for all programming languages"
- syntax: strict indentation!

yaml validator - Google-Suche

google.com/search?q=yaml+validator&rlz=1C5CHFA\_enAT835AT835&oq=ya&aqs=chro...

Alle Bilder News Bücher Videos Mehr Einstellungen Suchfilter

Ungefähr 449 000 Ergebnisse (0,31 Sekunden)

www.yamllint.com ▾ Diese Seite übersetzen

## YAMLLint - The YAML Validator

Validate and Verify your YAML documents, optimized for Ruby on Rails.

codebeautify.org › yaml-validator ▾ Diese Seite übersetzen

## Best YAML Validator Online - Code Beautify

Free YAML Validator is a web based validator and re-formatter for YAML. Also known as YAML Lint.

onlinyamltools.com › validate-yaml ▾ Diese Seite übersetzen

## Validate YAML - Online YAML Tools

Yaml validator. World's simplest yaml tool. Quickly check Yet Another Markup Language syntax for errors. Enter your YAML in the input box below and you'll ...

Andere suchten auch nach

- yaml schema
- fix yaml file
- yaml debug online
- yaml syntax
- yaml validator
- php yaml parser online

JF jsonformatter.org › yaml-validator ▾ Diese Seite übersetzen

## Best YAML Validator Online - JSON Formatter

YAML Validator is a web based validator and re-formatter for YAML.

https://codebeautify.org/yaml-validator

Evernote

Sign in to Web Clipper to see Related Results

Code Editors also have plugins  
for YAML syntax validation

# YAML configuration files

```
! nginx-deployment.yaml ✘  
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  +  
11  +  
12  +  
22
```

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

# Blueprint for pods

(Template)

## Layers of Abstraction



Deployment manages a ..



ReplicaSet manages a ..



Pod is an abstraction of ..



Deployment manage Pods

! nginx-deployment.yaml x



! nginx-service.yaml x



```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5  +  labels: ...
6
7  spec:
8    replicas: 2
9  +  selector: ...
10 + template: ...
11
12
13
14
15
16
17
18
19
20
21
22
```

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: nginx-service
5  spec:
6    selector: ...
7
8    ports: ...
9
10
11
12
```





! nginx-deployment.yaml x



```
4   name: nginx-deployment
5   + labels: ...
7   spec:
8     replicas: 2
9   + selector: ...
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-service.yaml x



```
1   apiVersion: v1
2   kind: Service
3   metadata:
4     name: nginx-service
5   spec:
6   + selector: ...
8   + ports: ...
12
```



```
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
14   spec:
15     containers:
16       - name: nginx
17         image: nginx:1.16
18         ports:
19           - containerPort: 8080
```

# Template

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

port?

name?

image?

# Connecting components

## (Labels & Selectors & Ports)

# Deployment

# Labels & Selectors

```
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: ...
```

## Service

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: nginx-service
5  spec:
6    selector:
7      app: nginx
8    > ports: ...
12
```

# Deployment

## Labels & Selectors

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

## Service

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

# Deployment

# Labels & Selectors

```
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: ...
```

## Service

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

# Deployment

## Connecting Deployment to Pods

```
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:
```

- any key-value pair for component

# Deployment

## Connecting Deployment to Pods

```
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: ...
```

- Pods get the label through the template blueprint

# 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 >   spec: ...
```

## Connecting Deployment to Pods

- Pods get the label through the template blueprint
- This label is matched by the selector

```
  selector:
    matchLabels:
      app: nginx
```

# Connecting Services to Deployments

```
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: ...
```

Service

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: nginx-service
5  spec:
6    selector:
7      app: nginx
8    ports: ...
12
```

# Connecting Services to Deployments

```
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: ...
```

Service

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: nginx-service
5  spec:
6    selector:
7      app: nginx
8    > ports: ...
12
```

! nginx-deployment.yaml x



```
/   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
```

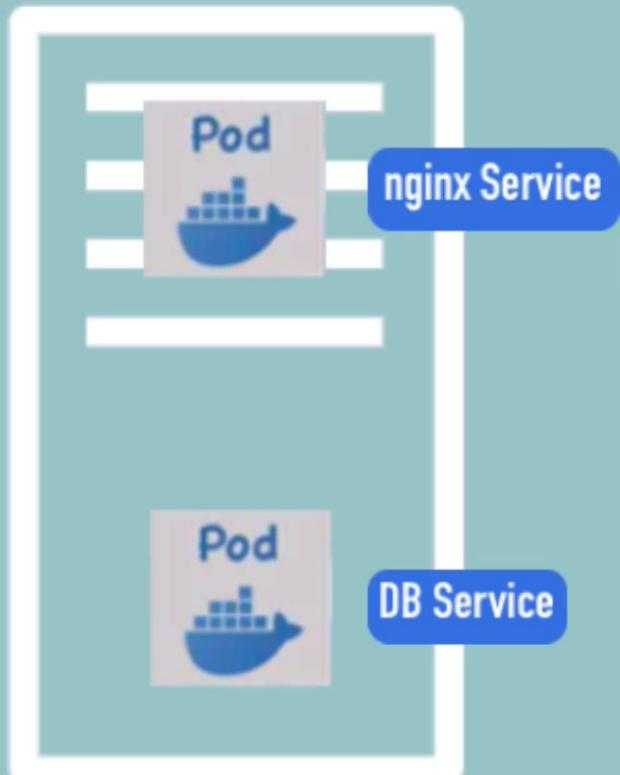
! nginx-service.yaml x



```
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
```



# Ports in Service and Pod



```
ports:  
- protocol: TCP  
port: 80  
targetPort: 8080
```

DB Service

↓  
port: 80

nginx Service

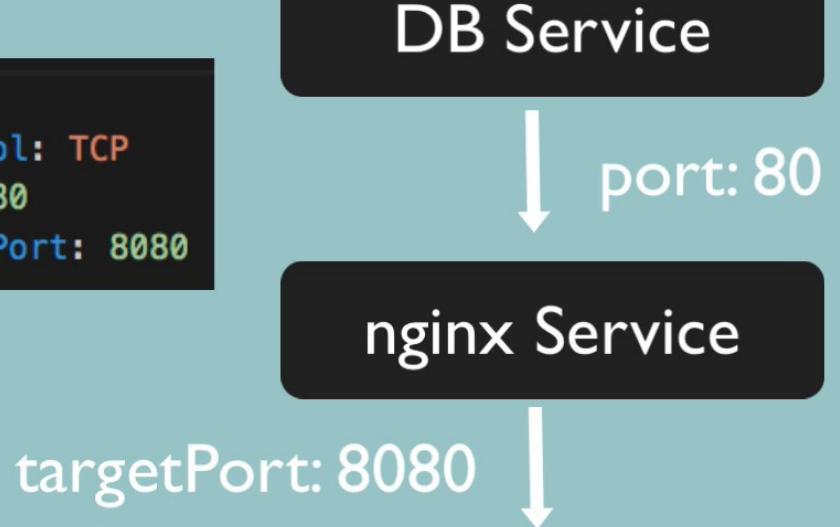
targetPort: 8080

↓  
Pod

# Ports in Service and 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  spec:
12    containers:
13      - name: nginx
14        image: nginx:1.16
15        ports:
16          - containerPort: 8080
```

```
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```



DB Service

port: 80

nginx Service

targetPort: 8080

Pod

! nginx-deployment.yaml x



```
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
```

! nginx-service.yaml x



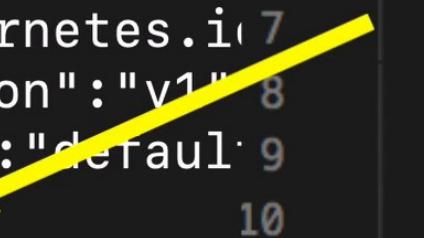
```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: nginx-service
5 spec:
6   selector:
7     app: nginx
8   ports:
9     - protocol: TCP
10    port: 80
11    targetPort: 8080
```



```
[Documents]$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
[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]$ kubectl describe service nginx-servic
```

```
[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]$ kubectl describe service nginx-service
Name:           nginx-service
Namespace:      default
Labels:         <none>
Annotations:   kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"nginx-service","namespace":"default"},"spec":{"ports":[{"port":80...
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
Session Affinity: None
Events:         <none>
[Documents]$
```

```
[Documents]$ kubectl get service
NAME           TYPE      CLUSTER-IP   AGE
kubernetes     ClusterIP 10.96.0.1    2d
nginx-service  ClusterIP 10.96.25.229 19s
[Documents]$ kubectl describe service nginx
Name:           nginx-service
Namespace:      default
Labels:          <none>
Annotations:    kubectl.kubernetes.io/selector={"apiVersion":"v1","kind":"Service","name":"nginx-service","namespace":"default","resourceVersion":7,"selfLink":"/api/v1/namespaces/default/services/nginx-service","uid":8}
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
Session Affinity: None
Events:          <none>
[Documents]$
```



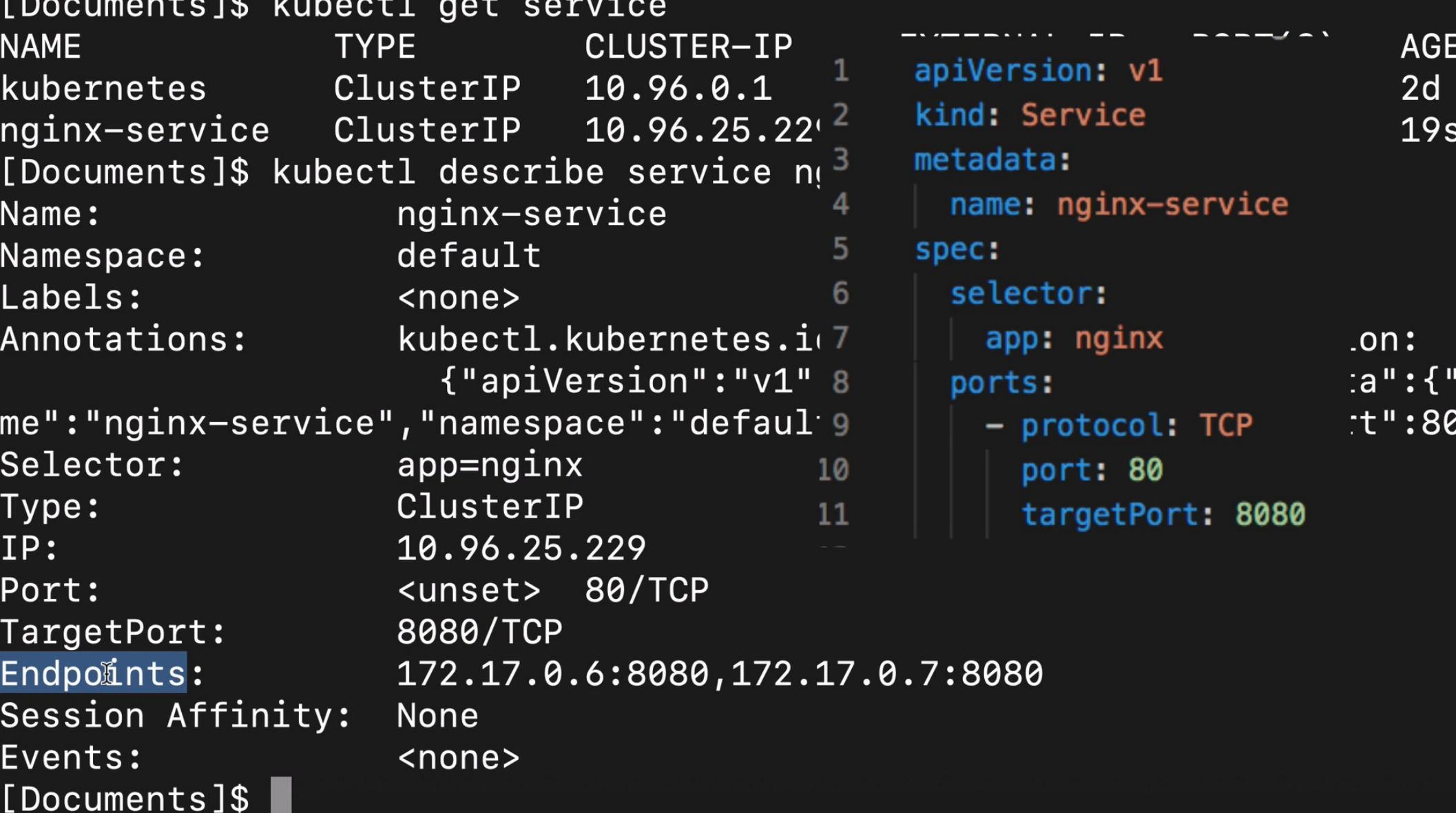
```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

```
[Documents]$ kubectl get service
NAME          TYPE      CLUSTER-IP   AGE
kubernetes    ClusterIP  10.96.0.1    2d
nginx-service ClusterIP  10.96.25.22  19s
[Documents]$ kubectl describe service nginx-service
Name:           nginx-service
Namespace:      default
Labels:          <none>
Annotations:    kubectl.kubernetes.io/selector={"apiVersion":"v1","kind":"Service","name":"nginx-service","namespace":"default","resourceVersion":9,"uid":8}
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
Session Affinity: None
Events:          <none>
[Documents]$
```



```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

```
[Documents]$ kubectl get service
NAME          TYPE      CLUSTER-IP   AGE
kubernetes    ClusterIP 10.96.0.1    2d
nginx-service ClusterIP 10.96.25.22 19s
[Documents]$ kubectl describe service nginx
Name:           nginx-service
Namespace:      default
Labels:          <none>
Annotations:    kubectl.kubernetes.io/selector={"apiVersion":"v1","kind":"Service","name":"nginx-service","namespace":"default","resourceVersion":10,"uid":10}
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
Session Affinity: None
Events:          <none>
[Documents]$
```



```
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
```

nginx-service ClusterIP 10.96.25.229 <none> 80/TCP 19s

```
[Documents]$ kubectl describe service nginx-service
Name:          nginx-service
Namespace:     default
Labels:        <none>
Annotations:   kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"nginx-service","namespace":"default"},"spec":{"ports":[{"port":80...}}
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
Session Affinity: None
Events:        > <none>
```

[Documents]\$ kubectl get pod -o wide

NAME	NOMINATED NODE	READY	STATUS	RESTARTS	AGE	IP	NODE
nginx-deployment-7d64f4b574-fklxj	ikube <none>	1/1	Running	0	2m17s	172.17.0.7	min
nginx-deployment-7d64f4b574-v7mwj	ikube <none>	1/1	Running	0	2m17s	172.17.0.6	min

```
[Documents]$
```



[Documents]\$ █

█

Status automatically generated?

```
[Documents]$ kubectl get deployment nginx-deployment -o yaml
```

```
restartPolicy: Always
schedulerName: default-scheduler
securityContext: {}
terminationGracePeriodSeconds: 30
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

```
securityContext: {}
terminationGracePeriodSeconds: 30
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
[Documents]$
```

! 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
```

...

! nginx-deployment-result.yaml ×

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   annotations:
5     deployment.kubernetes.io/revision: "1"
6     kubectl.kubernetes.io/last-applied-configuration: {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"name":"nginx-deployment","namespace":"default","resourceVersion":"96574","selfLink":"/apis/apps/v1/namespaces/default/deployments/nginx-deployment","uid":"e1075fa3-6468-43d0-83c0-63fede0dae51"}, "spec":{"progressDeadlineSeconds":600,"replicas":2,"revisionHistoryLimit":10,"selector":{"matchLabels":}}
```



```
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 pro
  reason: NewReplicaSetAvailable
  status: "True"
  type: Progressing
observedGeneration: 1
readyReplicas: 2
replicas: 2
updatedReplicas: 2
```

[Documents]\$ kubectl get deployment nginx-deployment -o yaml > nginx-deploy

ml

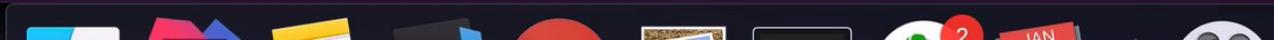
[Documents]\$ kubectl delete -f nginx-deployment.yaml

deployment.apps "nginx-deployment" deleted

[Documents]\$ kubectl delete -f nginx-service.yaml

service "nginx-service" deleted

[Documents]\$ █



! mongo.yaml

```
31      name: mongodb-secret
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: 30
```

! mongo-express.yaml

## 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**

**Port you need to put into browser**



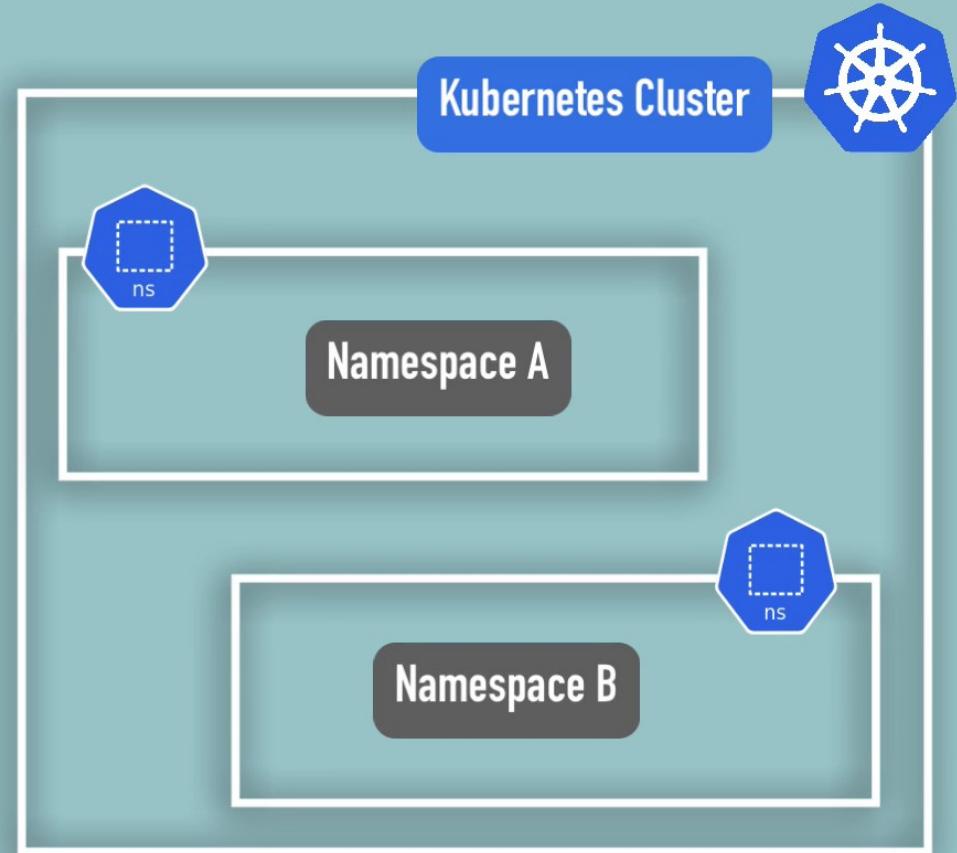
0 0 ▲ 0





# Kubernetes Namespaces explained

- **What is a Namespace?**
- **What are the use cases?**
- **How Namespaces work and how to use it?**

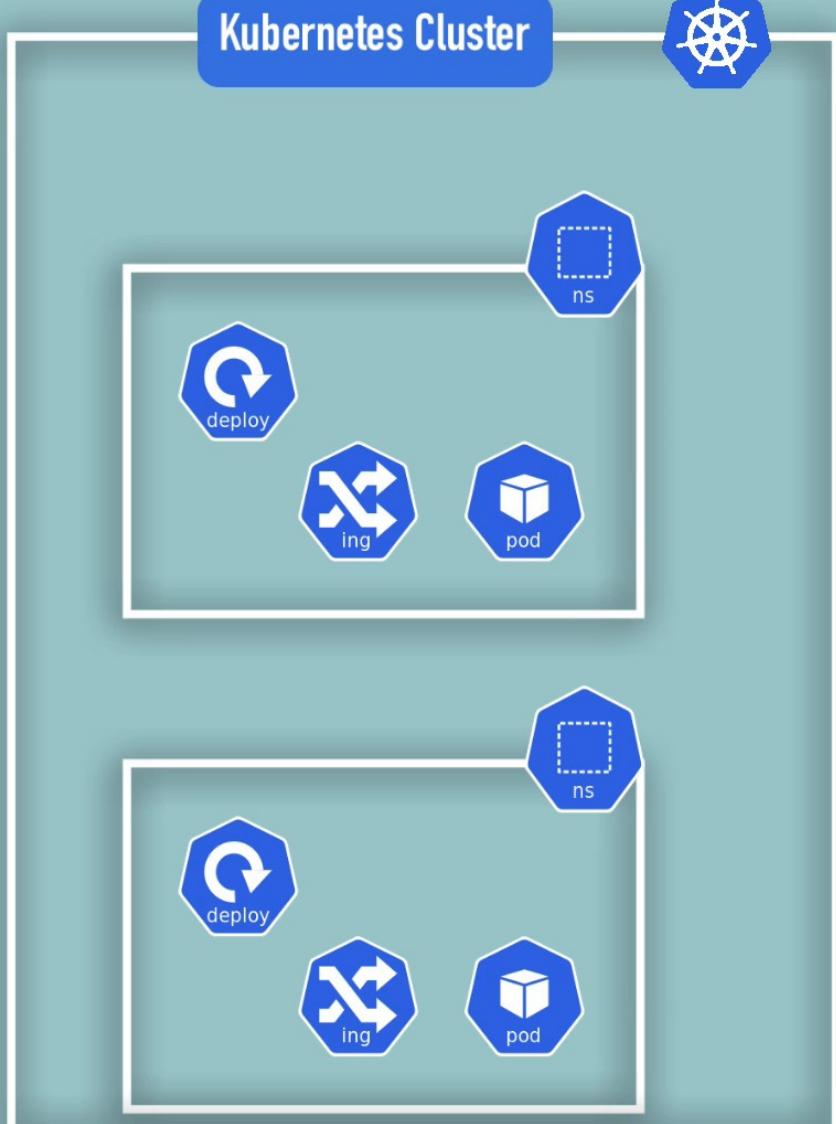




## What is a Namespace?

- Organise resources in namespaces
- Virtual cluster inside a cluster

Kubernetes Cluster





## 4 Namespaces per Default

Kubernetes Cluster



```
Terminal Shell Edit View Window Help
~/TEST-k8s-con
[[TEST-k8s-configuration]$ kubectl get namespace
NAME           STATUS   AGE
default        Active   6d2h
kube-node-lease Active   6d2h
kube-public    Active   6d2h
kube-system    Active   6d2h
kubernetes-dashboard Active  2m20s
[[TEST-k8s-configuration]$
```



## 4 Namespaces per Default

- **kubernetes-dashboard only with minikube**

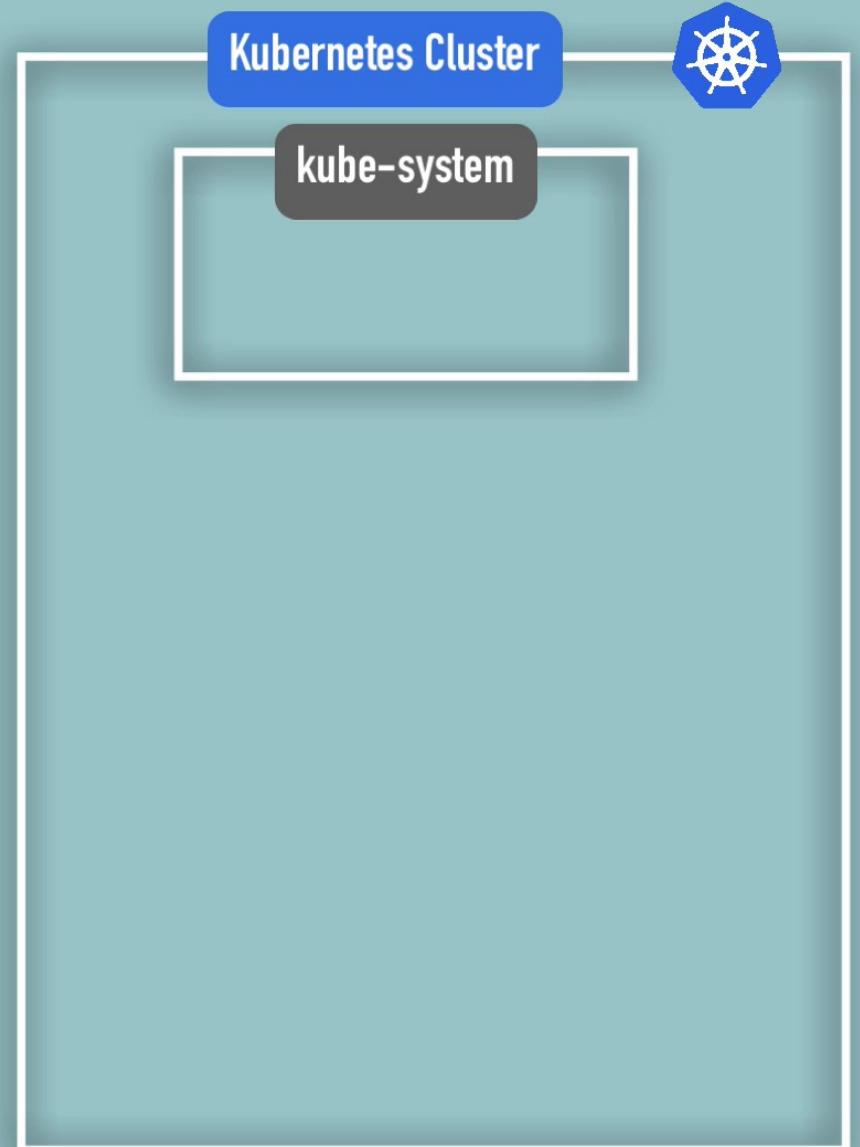
Kubernetes Cluster





## 4 Namespaces per Default

```
Terminal Shell Edit View Window Help
~/TEST-k8s-con
[TEST-k8s-configuration]$ kubectl get namespace
NAME          STATUS   AGE
default        Active   6d2h
kube-node-lease Active   6d2h
kube-public    Active   6d2h
kube-system    Active   6d2h
kubernetes-dashboard Active  2m20s
[TEST-k8s-configuration]$
```





## 4 Namespaces per Default

- **Do NOT create or modify  
in kube-system**
- **System processes**
- **Master and Kubectl process**

Kubernetes Cluster



kube-system





## 4 Namespaces per Default

```
Terminal Shell Edit View Window Help
[TEST-k8s-configuration]$ kubectl get namespace
NAME          STATUS   AGE
default        Active   6d2h
kube-node-local Active   6d2h
kube-public    Active   6d2h
kube-system    Active   6d2h
kubernetes-dashboard Active  2m20s
[TEST-k8s-configuration]$
```

Kubernetes Cluster



kube-system



kube-public



## 4 Namespaces per Default

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

Kubernetes Cluster



kube-system



kube-public



```
Terminal Shell Edit View Window Help
~/TEST-k8s-configuration — bash — 102x28
[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
oxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
[TEST-k8s-configuration]$
```



## 4 Namespaces per Default

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

Kubernetes Cluster



kube-system



kube-public



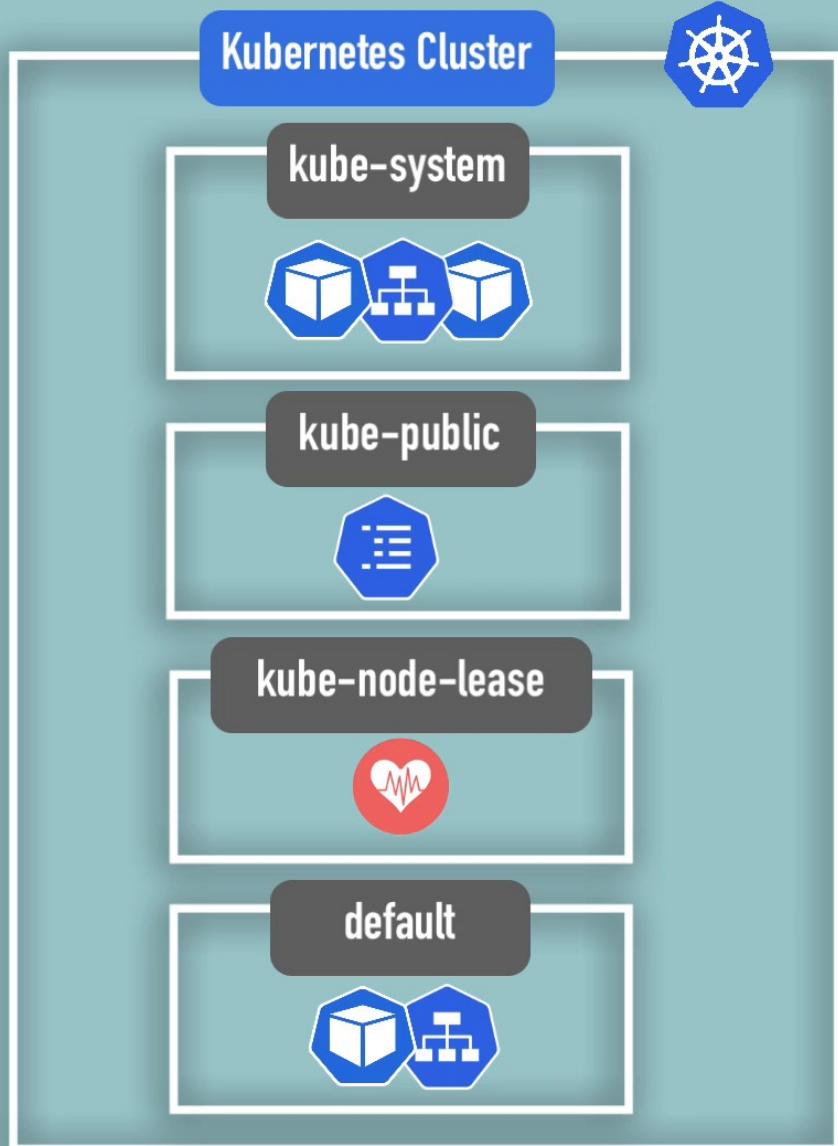
kube-node-lease





## Create a Namespace

- resources you create are located here





## Create a Namespace

```
Terminal Shell Edit View Window Help
~/TEST-k8s-configuration — bash — 102x28
[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/serv
oxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
[TEST-k8s-configuration]$ kubectl create namespace my-namespace
namespace/my-namespace created
[TEST-k8s-configuration]$ kubectl get namespace
NAME          STATUS   AGE
default        Active   6d2h
kube-node-lease Active   6d2h
kube-public    Active   6d2h
kube-system    Active   6d2h
kubernetes-dashboards Active  10...
my-namespace   Active   7m41s
[TEST-k8s-configuration]$
```



## Create a Namespace

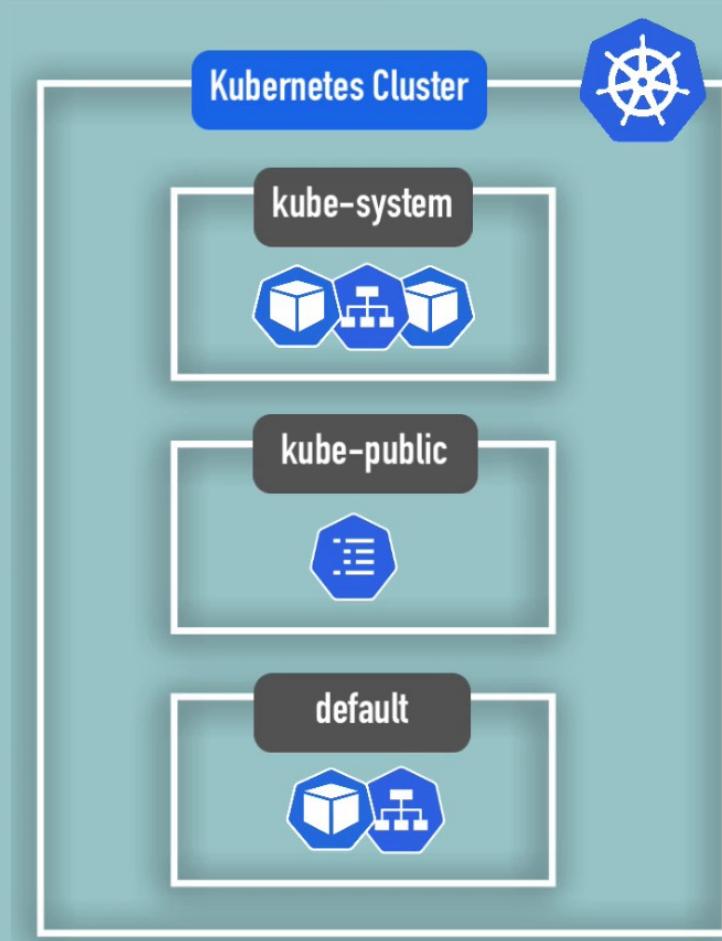
# Create a namespace with a configuration file

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



## What are Namespaces?

- Cluster inside a cluster
- Default Namespaces



# Why to use Namespaces? 🤔

1.

## Everything in one Namespace



Kubernetes Cluster



default



1.

## Everything in one Namespace



deployments  
replicasets  
services  
configmaps

Kubernetes Cluster



default



1.

## Everything in one Namespace



Kubernetes Cluster



default



No overview!



1.

## Resources grouped in Namespaces



Kubernetes Cluster



**Officially:**  
**Should not use**  
**for smaller**  
**projects**

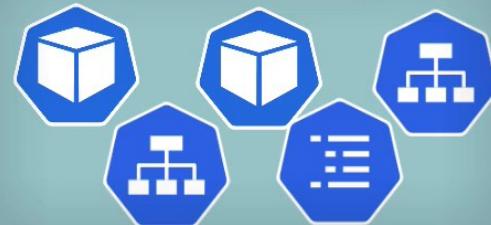
Database



Monitoring



Elastic Stack

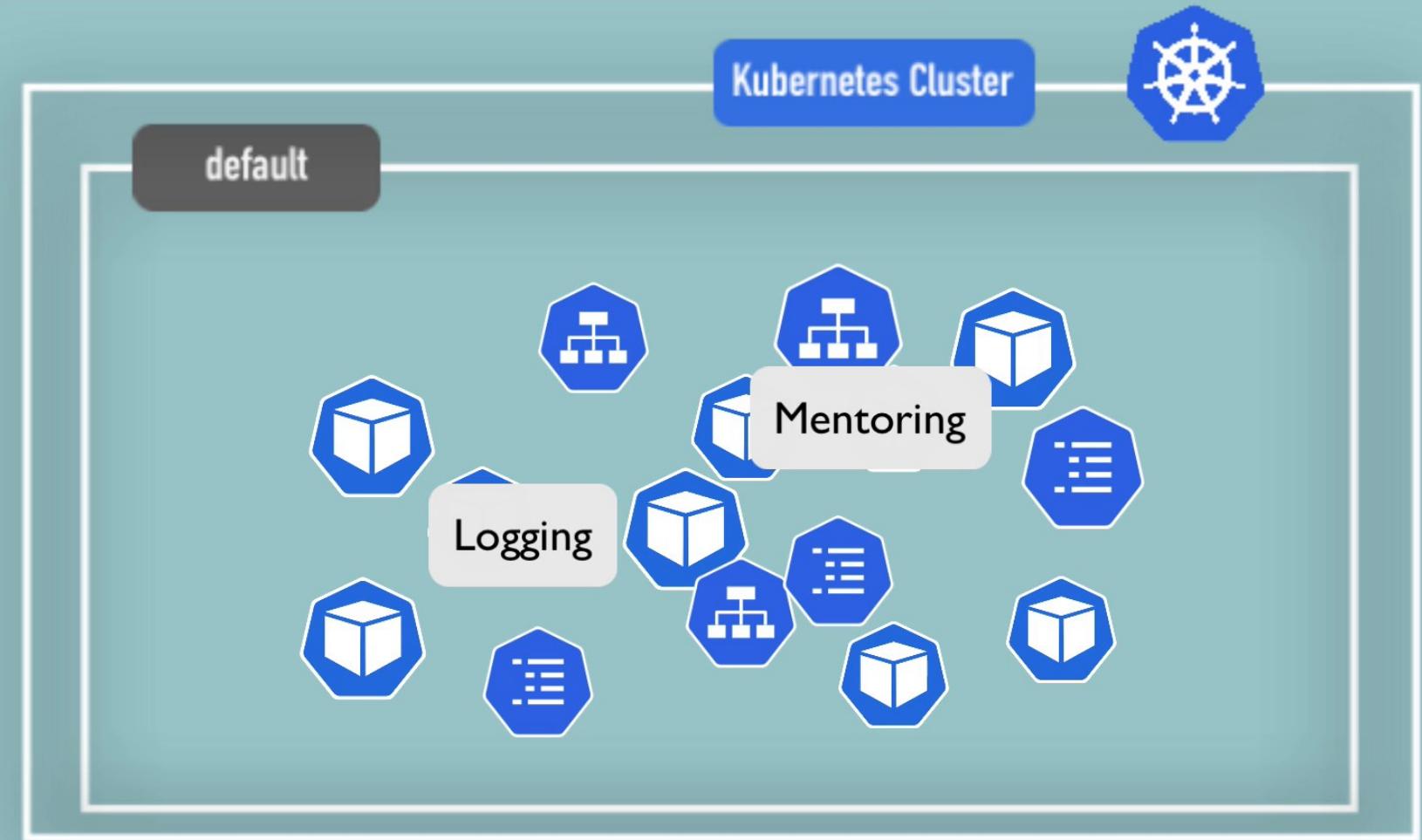


Nginx-Ingress



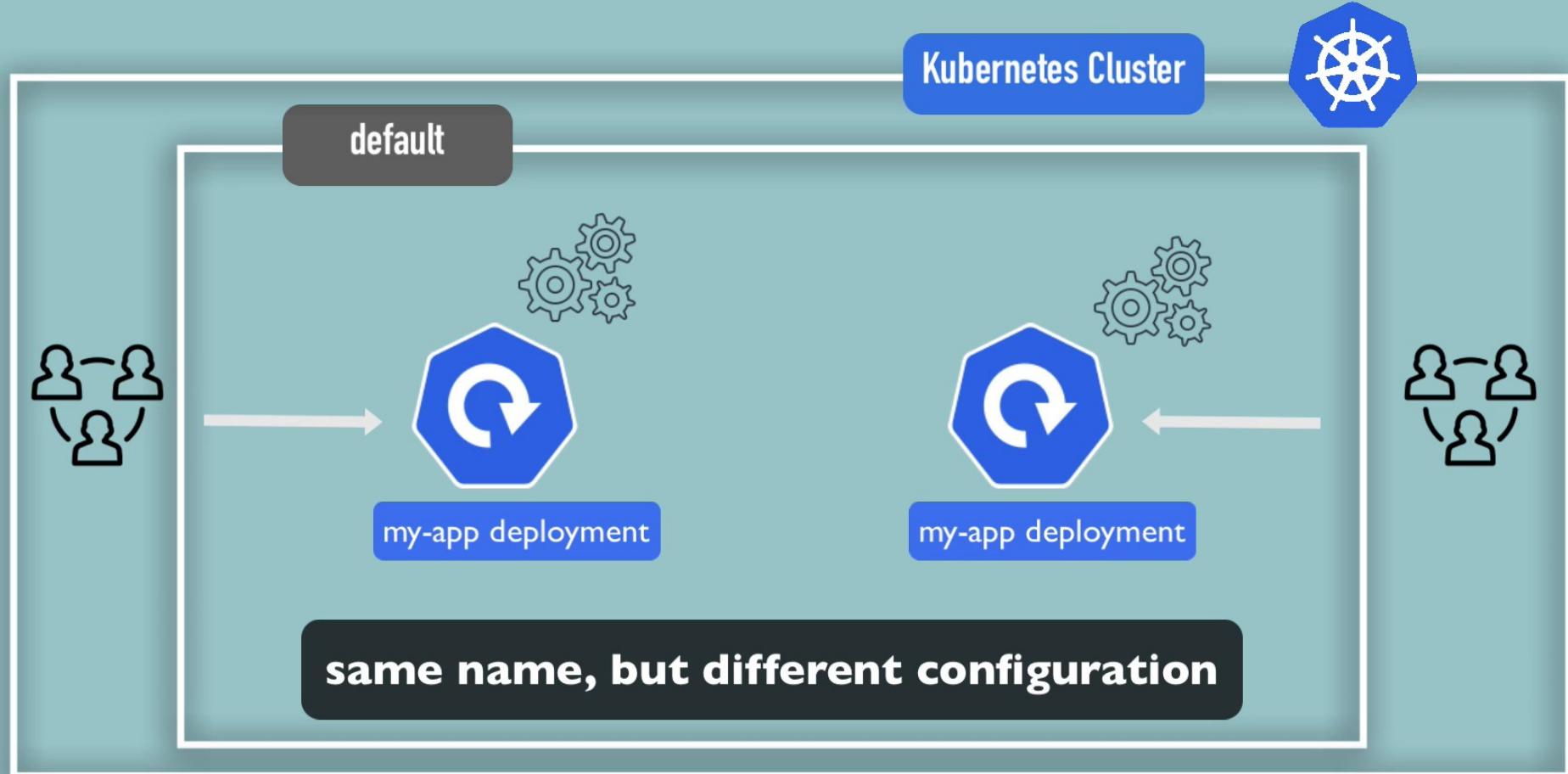
1.

## Resources grouped in Namespaces



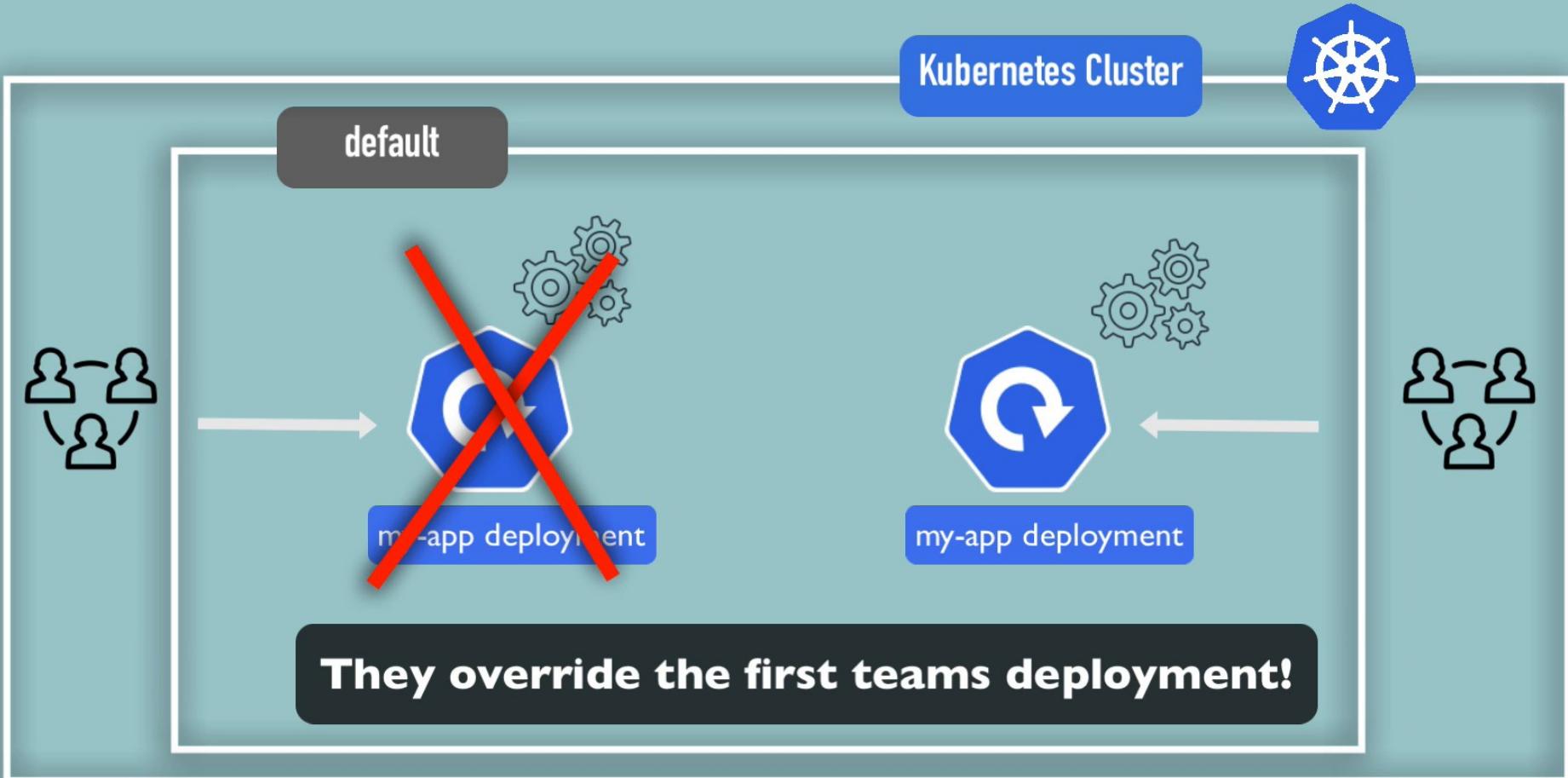
2.

## Conflicts: Many teams, same application



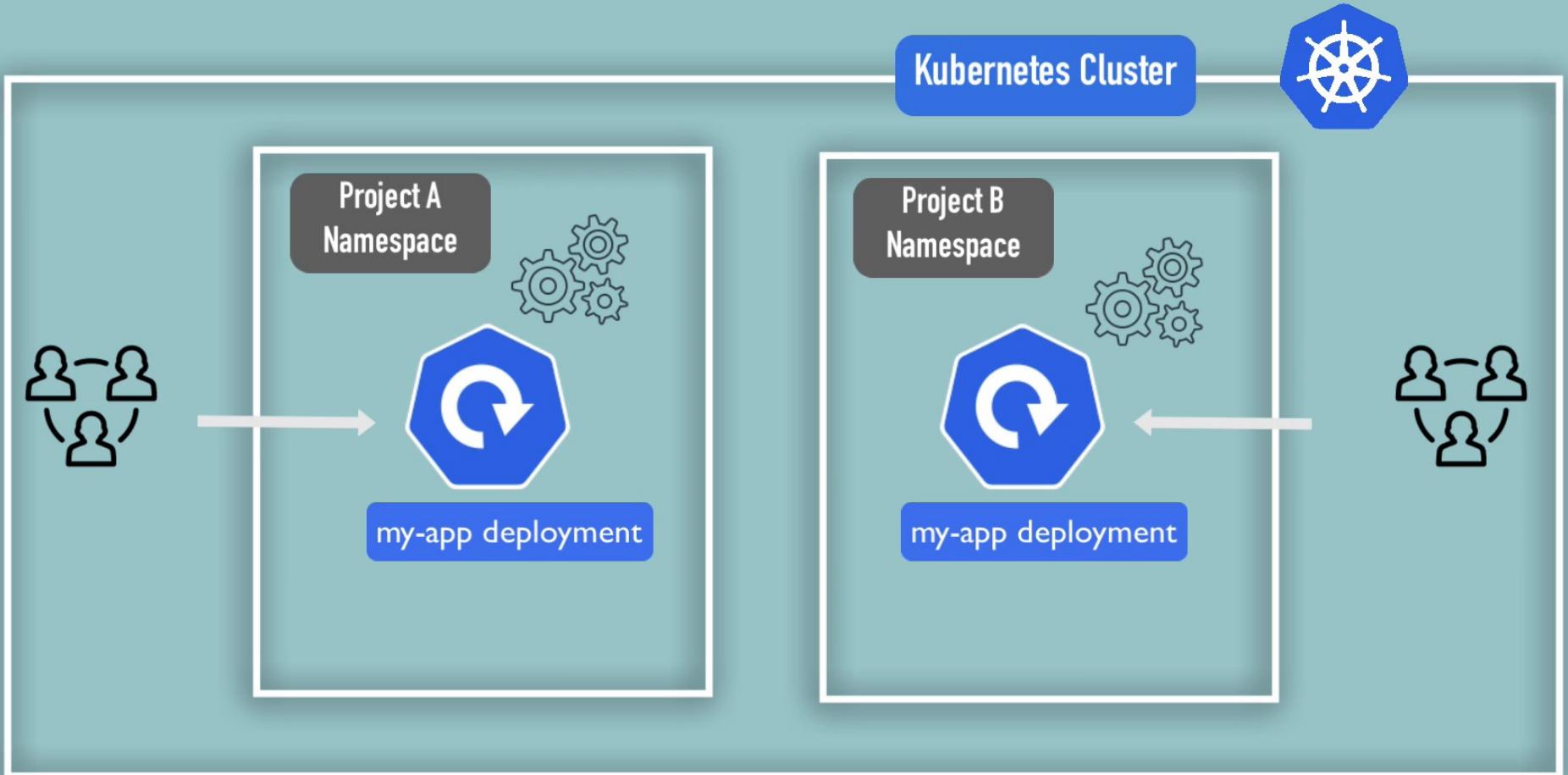
2.

## Conflicts: Many teams, same application



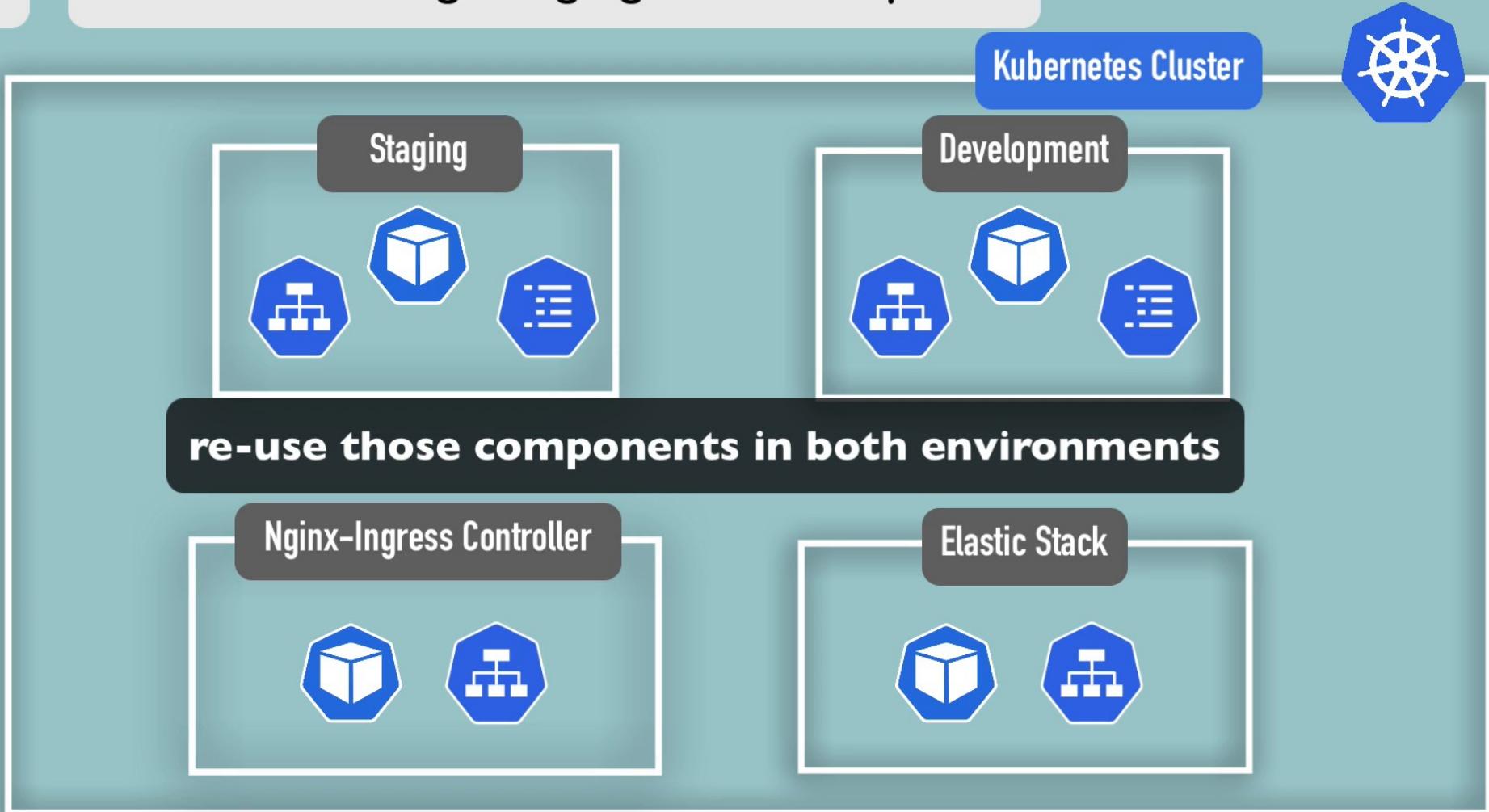
2.

## Conflicts: Many teams, same application



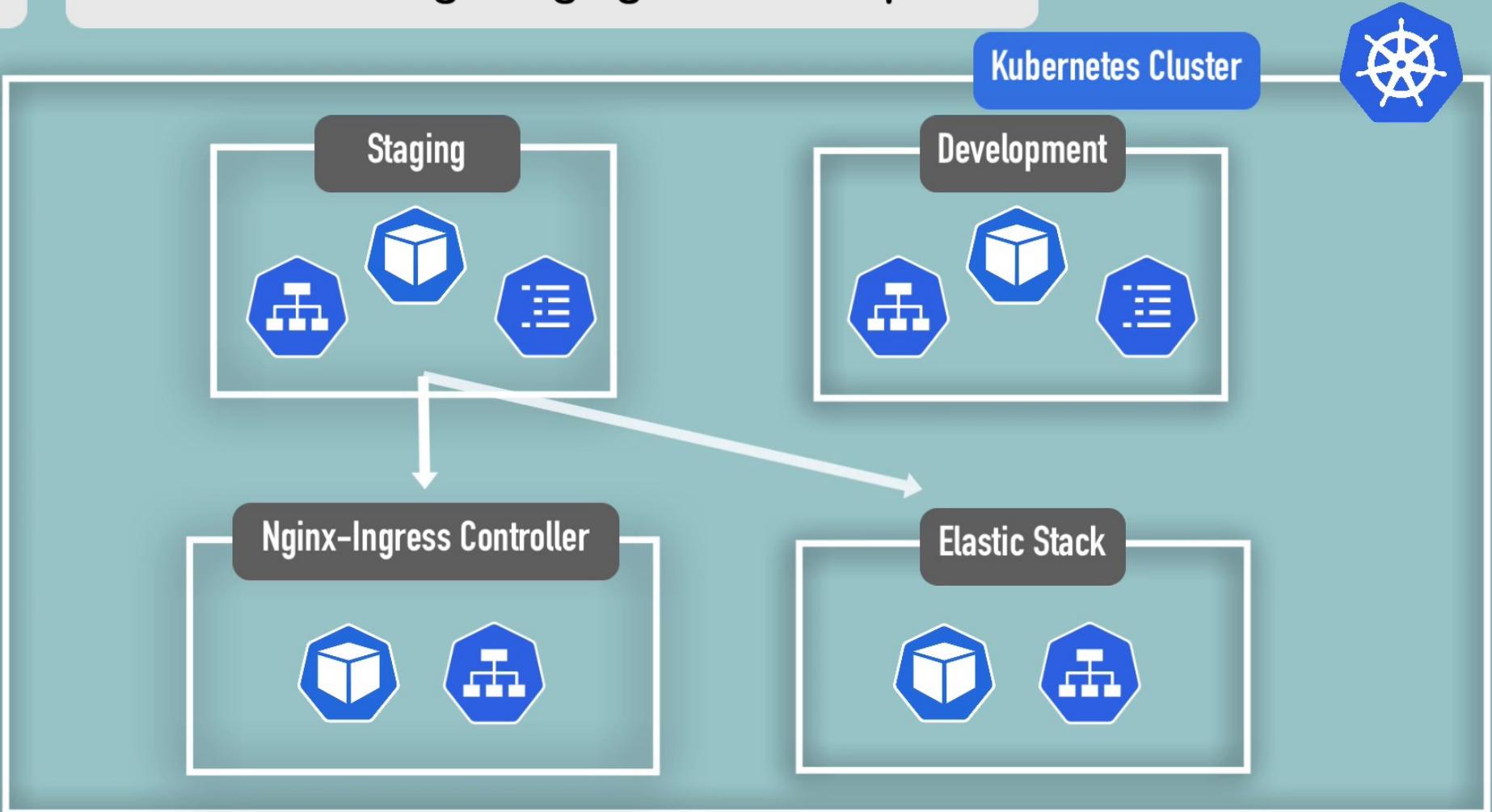
3.

## Resource Sharing: Staging and Development



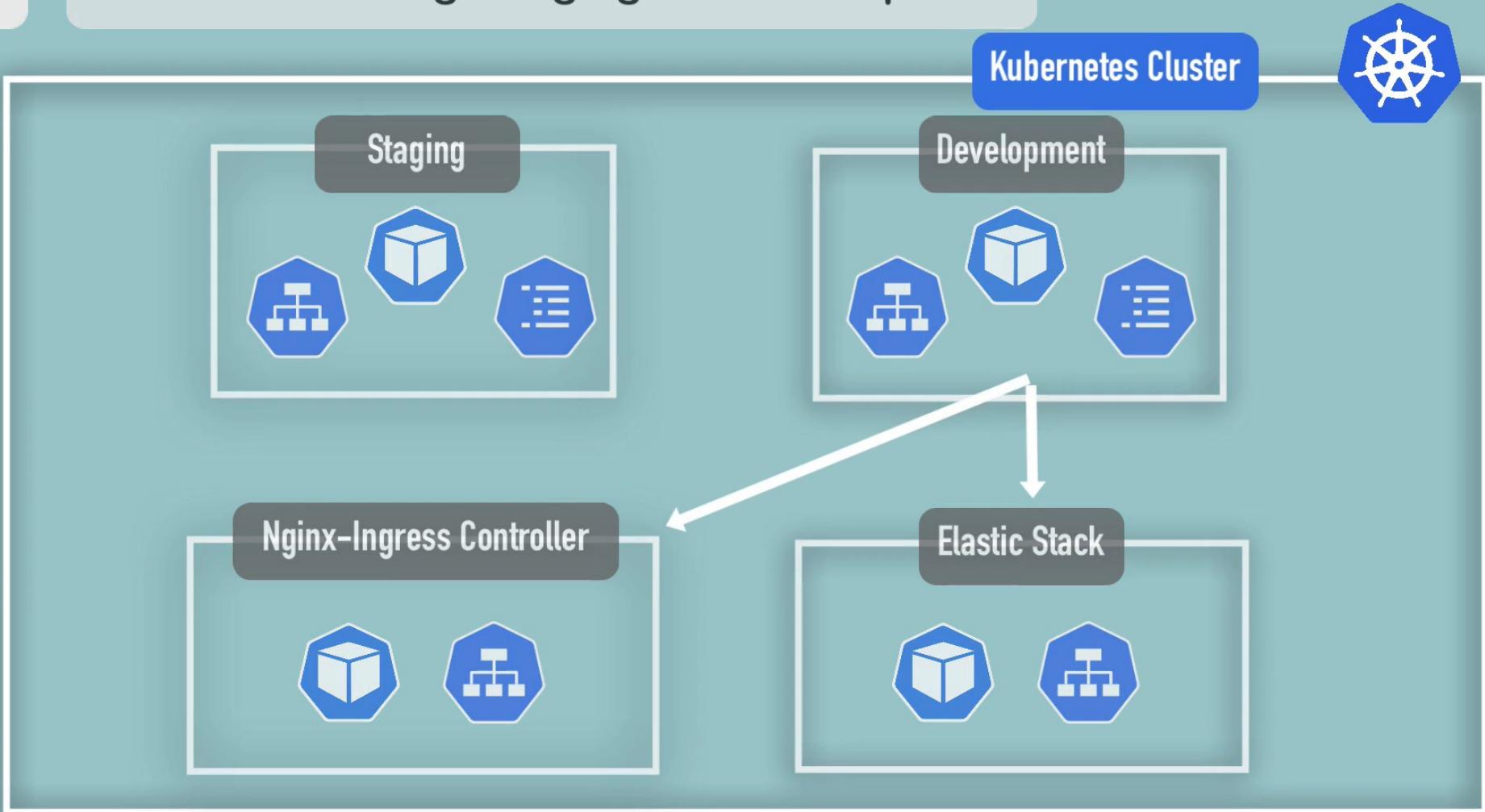
3.

### Resource Sharing: Staging and Development



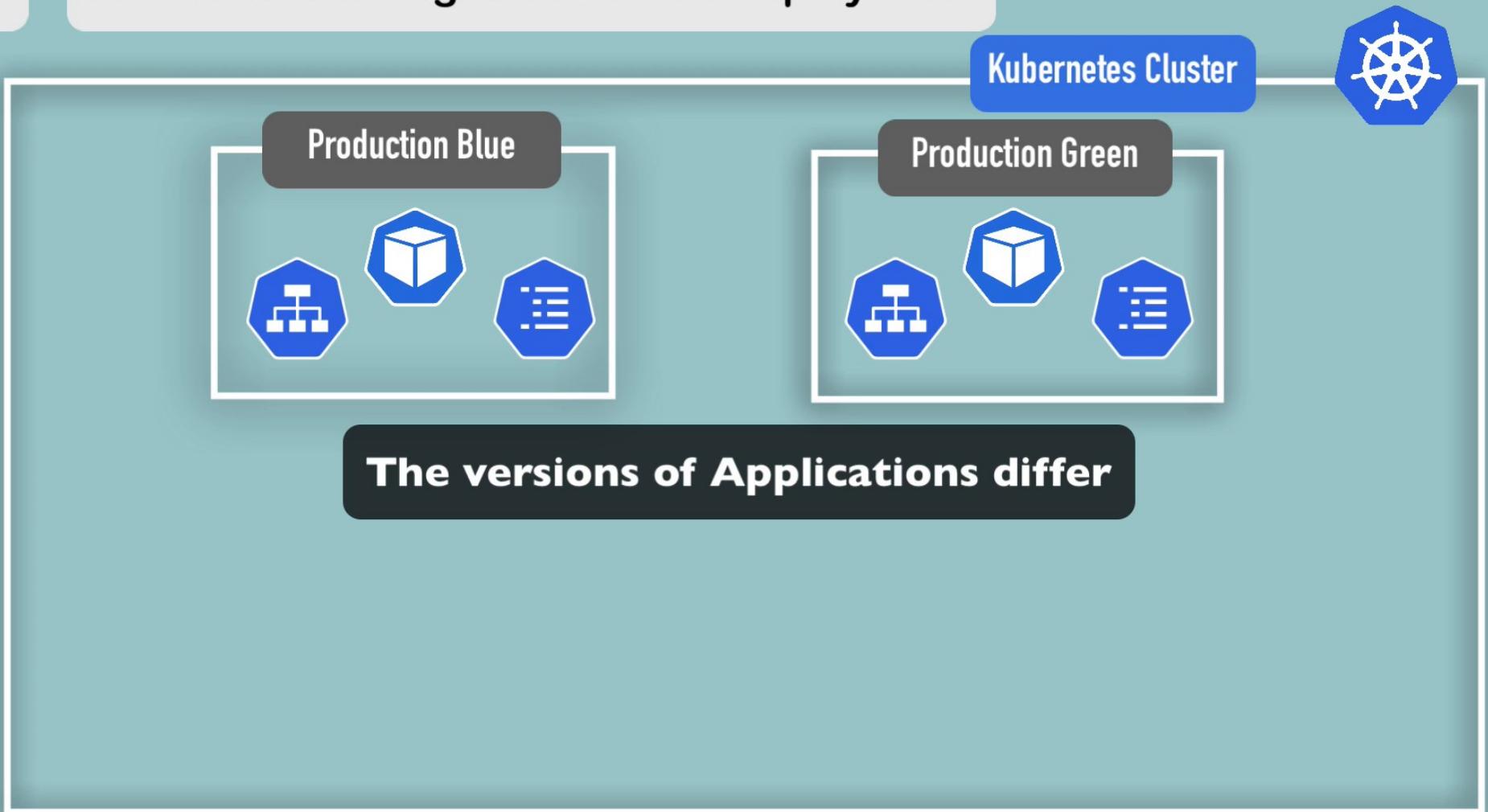
3.

## Resource Sharing: Staging and Development



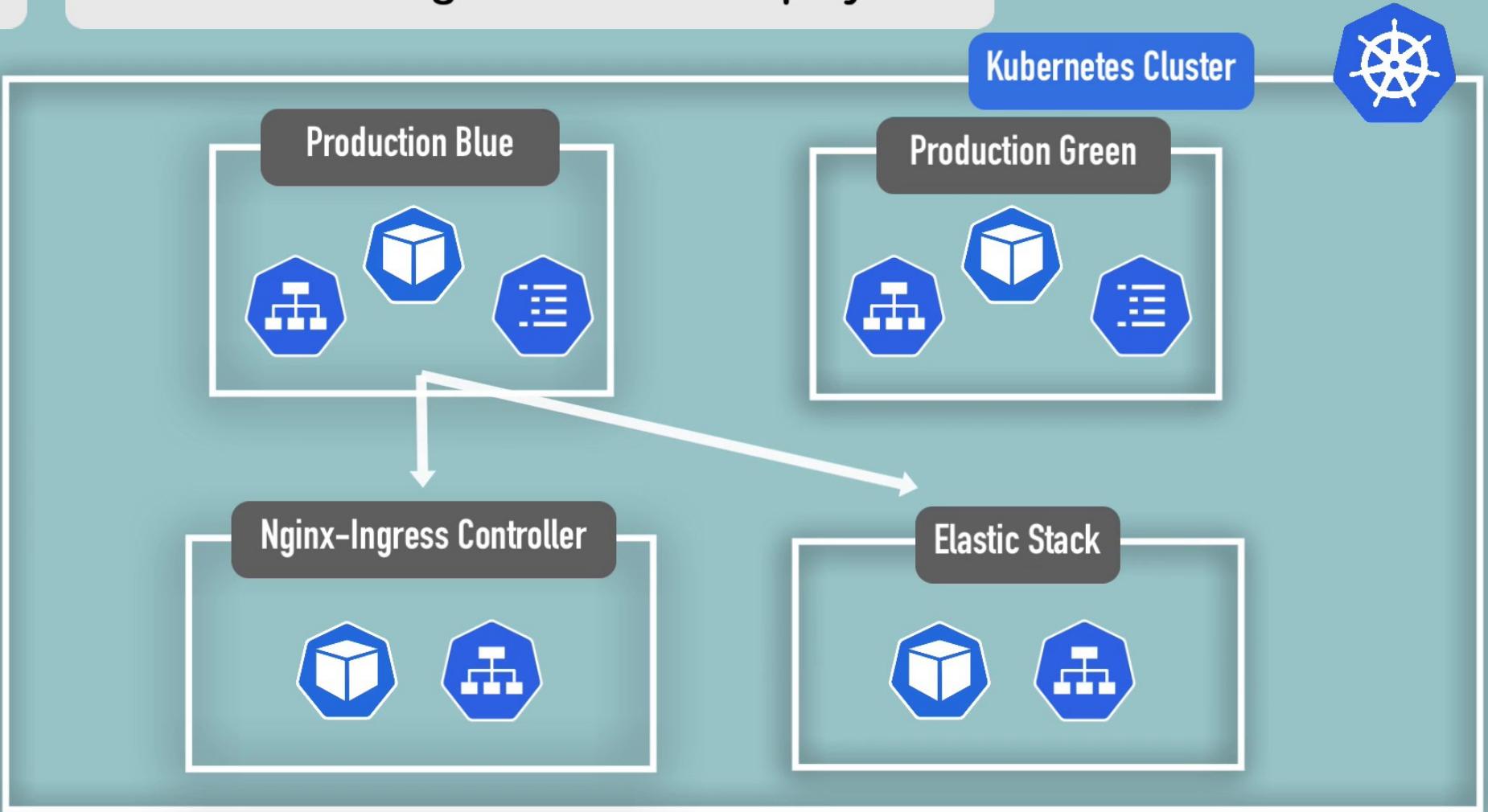
3.

## Resource Sharing: Blue/Green Deployment



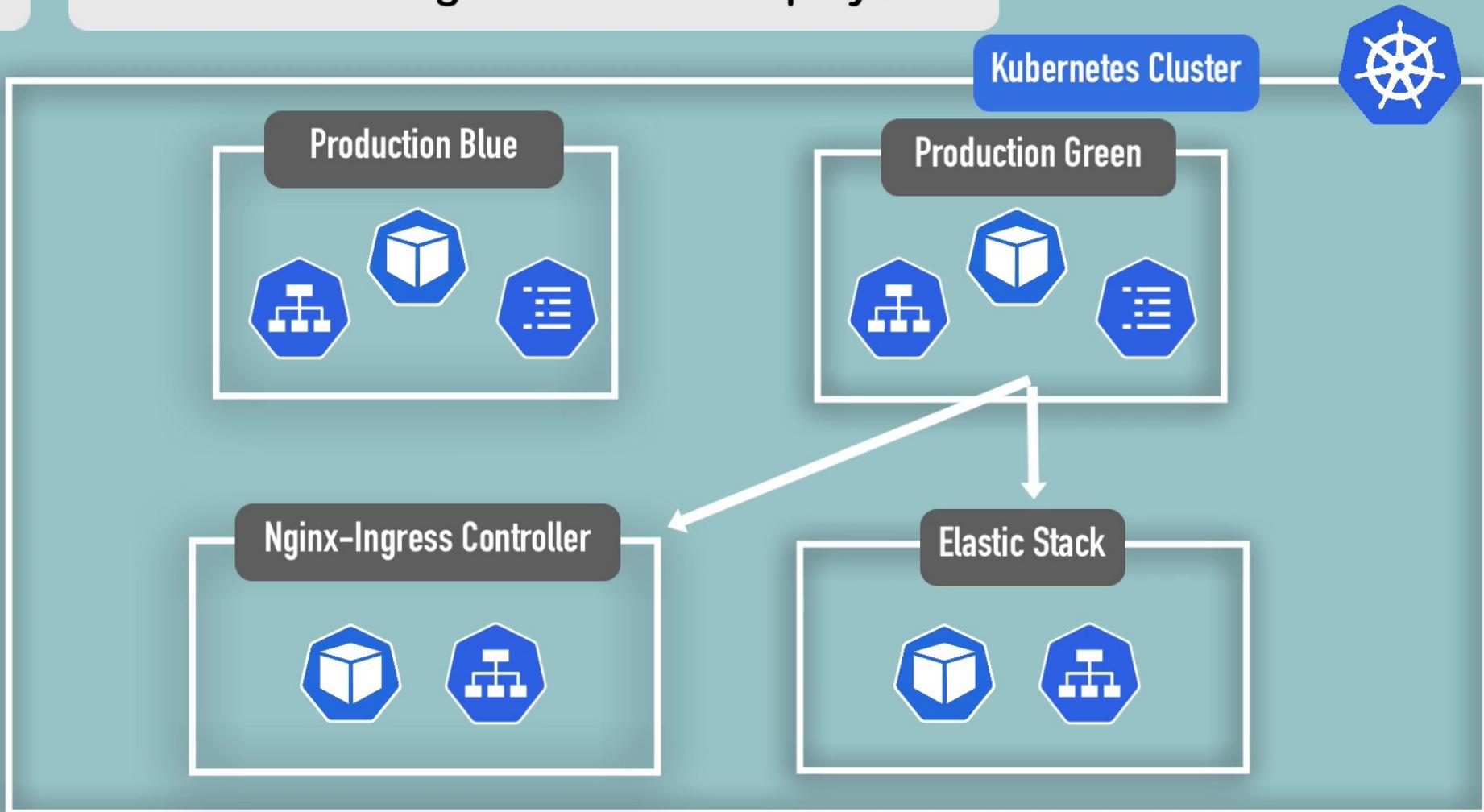
3.

## Resource Sharing: Blue/Green Deployment



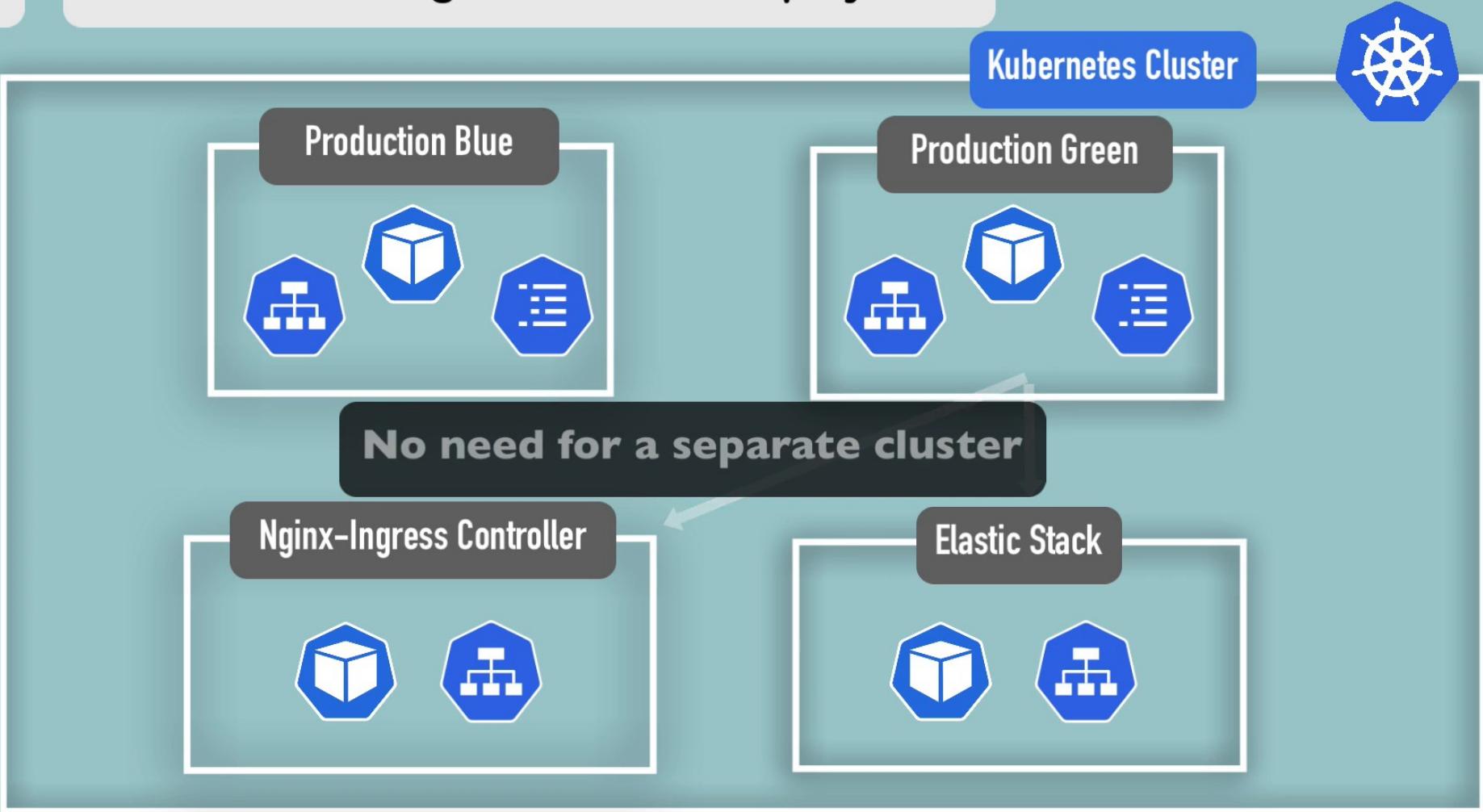
3.

## Resource Sharing: Blue/Green Deployment



3.

## Resource Sharing: Blue/Green Deployment

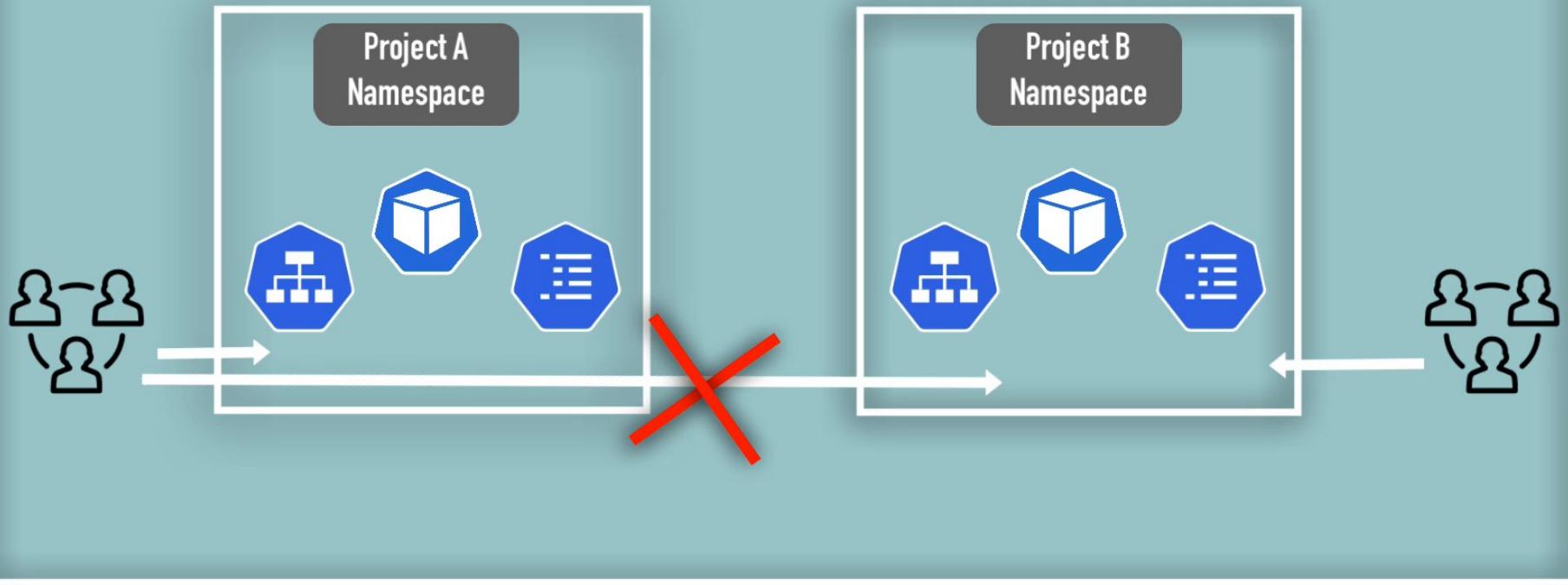


4.

## Access and Resource Limits on Namespaces

Each team has own, isolated environment

Kubernetes Cluster

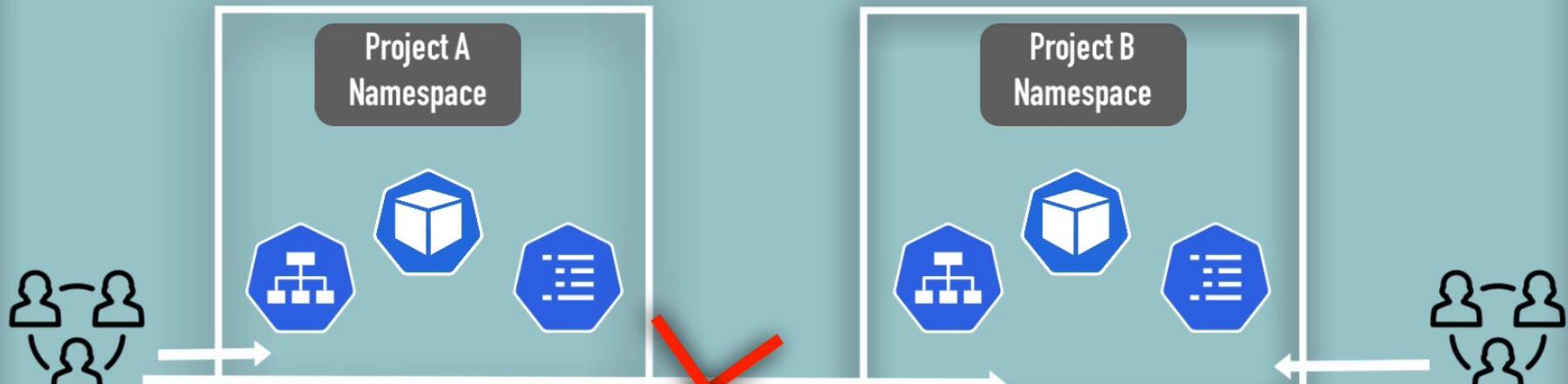


4.

## Access and Resource Limits on Namespaces

Limit: CPU, RAM, Storage per NS

Kubernetes Cluster



Resource Quota A

Resource Quota B



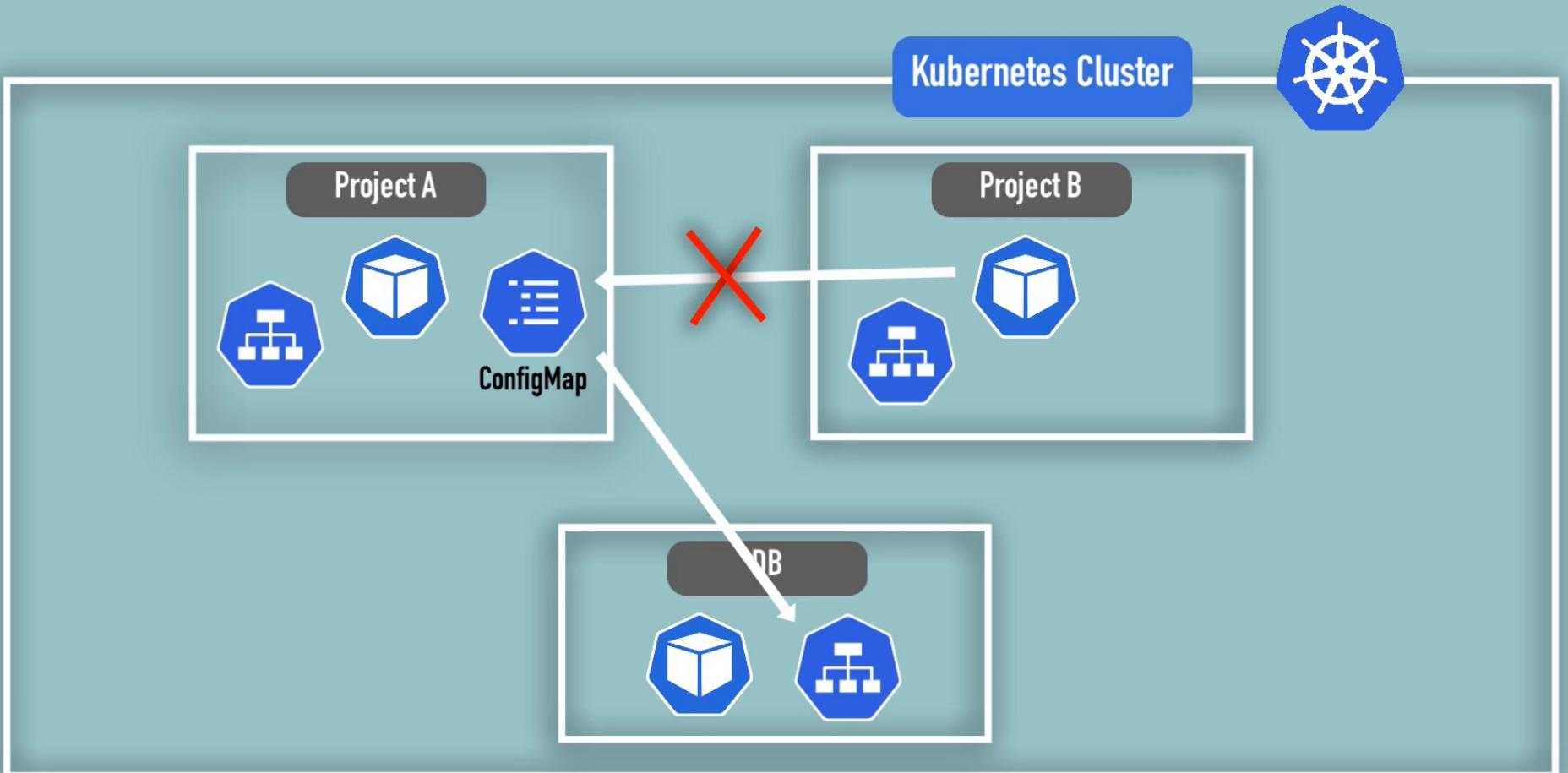
## Use Cases when to use Namespaces

- 1. Structure your components**
- 2. Avoid conflicts between teams**
- 3. Share services between different environments**
- 4. Access and Resource Limits on Namespaces Level**

# Characteristics of Namespaces?



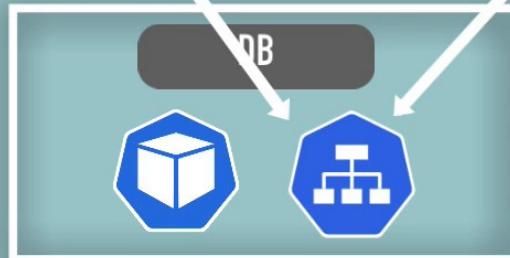
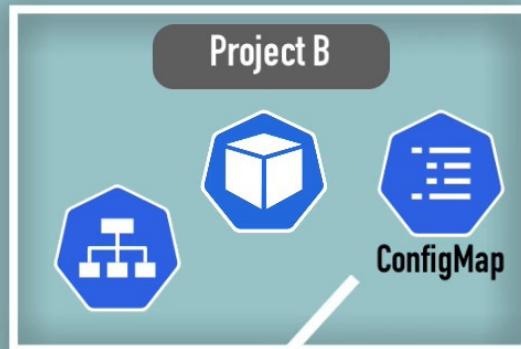
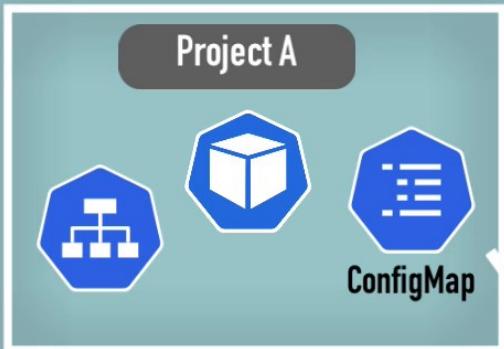
## You can't access most resources from another Namespace



You can't access most resources from another Namespace

Each NS must define own ConfigMap

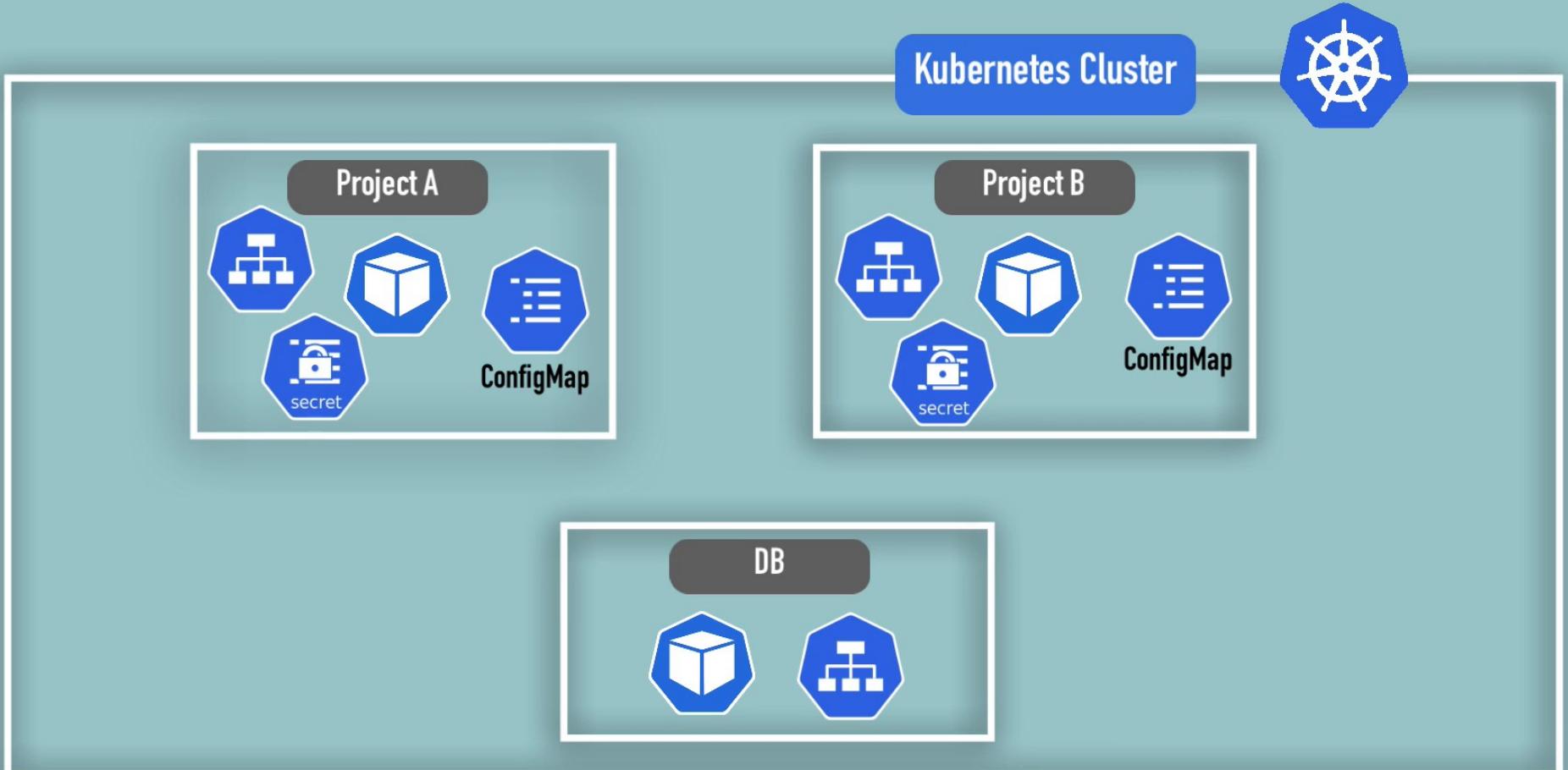
Kubernetes Cluster



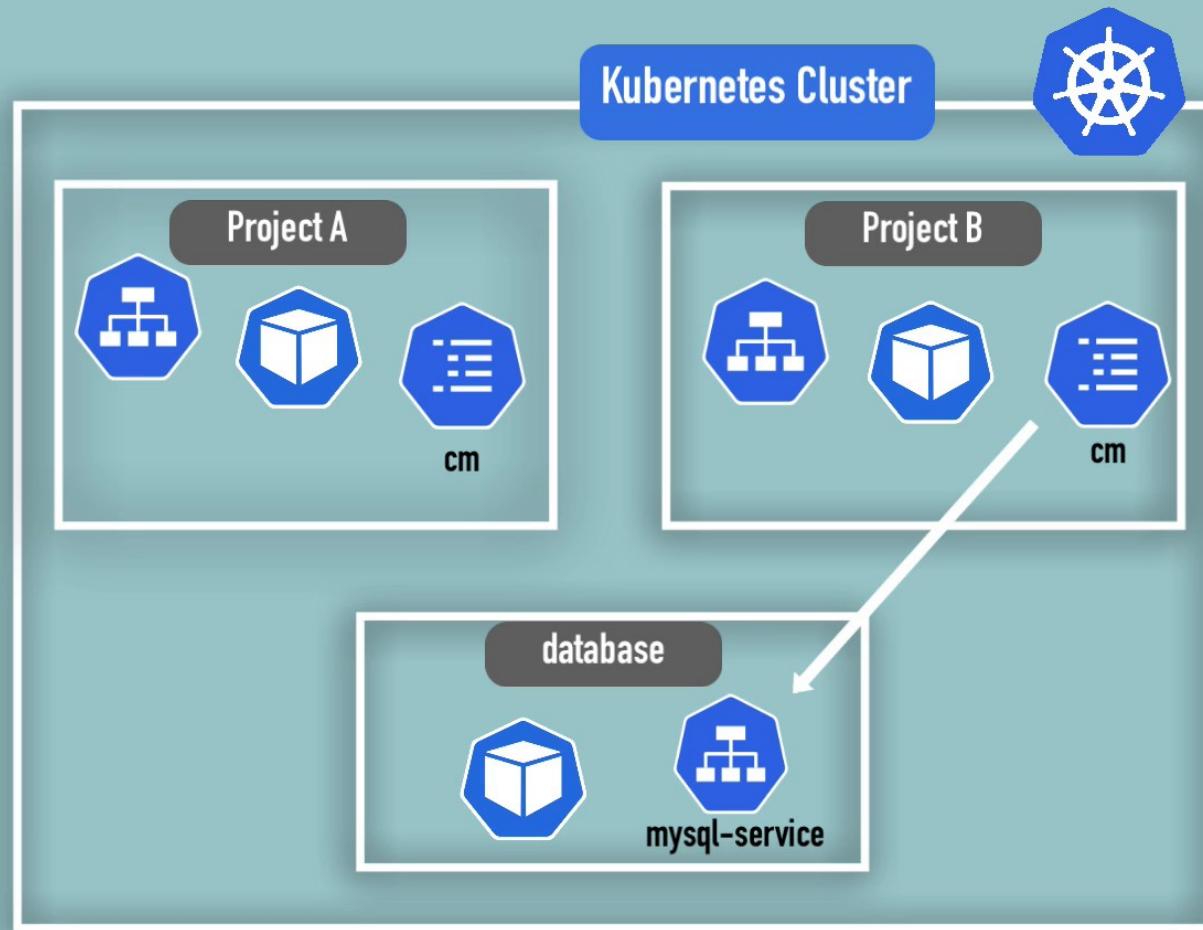
DB



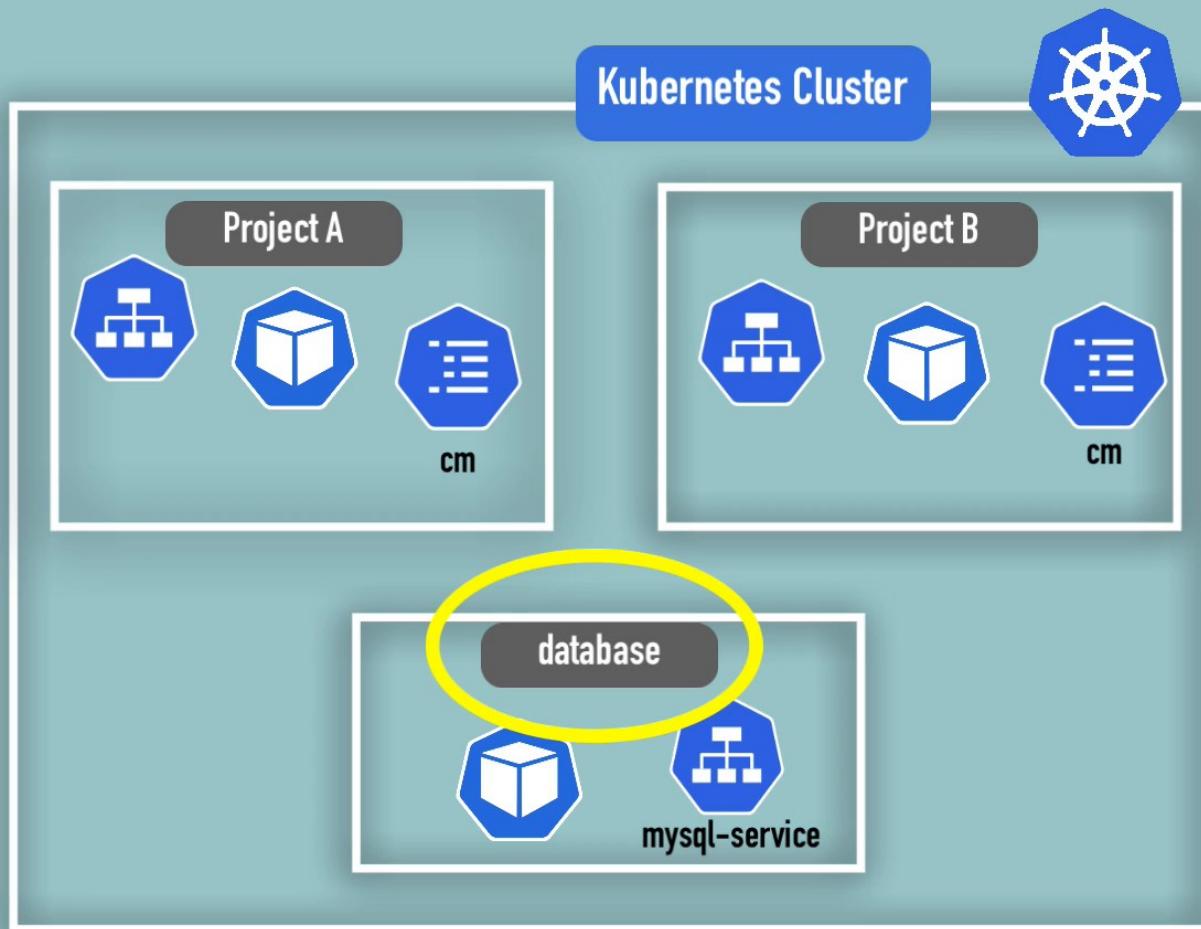
## You can't access most resources from another Namespace



## Access Service in another Namespace

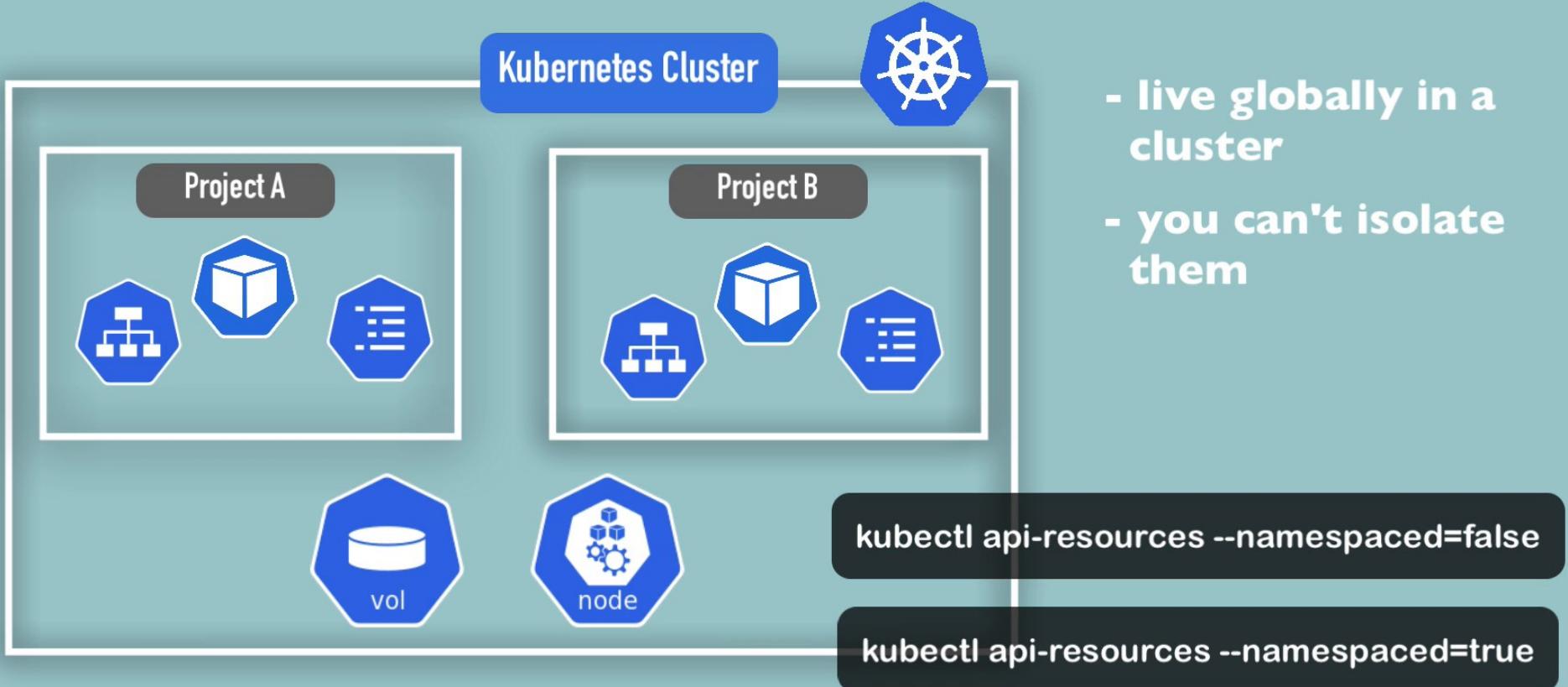


# Access Service in another Namespace



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

## Components, which can't be created within a Namespace





Create Components in Namespaces? 



## Create component in a Namespace

### No Namespace defined

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

**By default, components are created in a **default NS****



## Create component in a Namespace

### No Namespace defined

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-configmap
data:
  db_url: n
```

By default, components are

[NS]

```
[TEST-k8s-configuration]$ kubectl apply -f mysql-configmap.yaml
configmap/mysql-configmap created
[TEST-k8s-configuration]$ kubectl get configmap
NAME           DATA   AGE
mysql-configmap 1      10s
[TEST-k8s-configuration]$ kubectl get configmap -n my-nam
```



## Create component in a Namespace

### No Namespace defined

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-configmap
data:
  db_url: "mysql://root:password@127.0.0.1:3306/test"
[TEST-k8s-configuration]$ kubectl apply -f mysql-configmap.yaml
configmap/mysql-configmap created
[TEST-k8s-configuration]$ kubectl get configmap
NAME          DATA   AGE
mysql-configmap 1      10s
[TEST-k8s-configuration]$ kubectl get configmap -n default
NAME          DATA   AGE
mysql-configmap 1      30s
[TEST-k8s-configuration]$
```



```
Terminal Shell Edit View Window Help
~/TEST-k8s-configuration — bash — 102x28

NAME          DATA    AGE
mysql-configmap  1      30s
[[TEST-k8s-configuration]$ kubectl get configmap -o wide
NAME          DATA    AGE
mysql-configmap  1      37s
[[TEST-k8s-configuration]$ kubectl get configmap -o yaml
apiVersion: v1
items:
- apiVersion: v1
  data:
    db_url: mysql-service.database
  kind: ConfigMap
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration: |
        {"apiVersion":"v1","data":{"db_url":"mysql-service.database"}, "name": "mysql-configmap", "namespace": "default"}
        creationTimestamp: "2020-02-22T17:50:30Z"
        name: mysql-configmap
        namespace: default
        resourceVersion: "112739"
        selfLink: /api/v1/namespaces/default/configmaps/mysql-configmap
        uid: 7d2ac1ea-3e77-4443-8f8e-24d69e176b1d
  kind: List
  metadata:
    resourceVersion: ""
    selfLink: ""
[TEST-k8s-configuration]$
```



## Create component in a Namespace

```
Terminal Shell Edit View Window Help ~/TEST-k8s-configuration — bash — 102x28
[TEST-k8s-configuration]$ kubectl get all -n my-namespace
No resources found in my-namespace namespace.
[TEST-k8s-configuration]$ kubectl apply -f mysql-configmap.yaml --namespace=my-namespace
configmap/mysql created
[TEST-k8s-configuration]$
```



## Create component in a Namespace

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



## Create component in a Namespace

```
apiVersion: v1
kind: ConfigMap
```

Terminal Shell Edit View Window Help

~/TEST-k8s-configuration — bash — 102x28

```
[TEST-k8s-configuration]$ kubectl get all -n my-namespace
No resources found in my-namespace namespace.
[TEST-k8s-configuration]$ kubectl apply -f mysql-configmap.yaml --namespace=my-namespace
configmap/mysql-configmap created
[TEST-k8s-configuration]$ kubectl get all -n my-namespace
No resources found in my-namespace namespace.
[TEST-k8s-configuration]$ kubectl get configmap -n my-namespace
NAME          DATA   AGE
mysql-configmap  1     18s
[TEST-k8s-configuration]$
```



## Create component in a Namespace

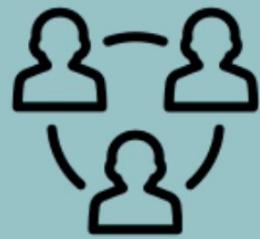
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-configmap
  namespace: my-namespace
data:
  db_url: mysql-service.database
```

**Configuration file over kubectl cmd**

**Better documented**

**More convenient**

## Change active namespace



**Only namespace  
"my-namespace"  
allowed**



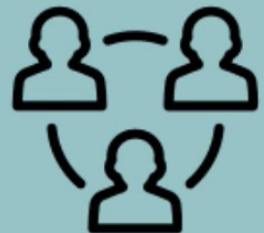
`-n my-namespace`

`-n my-namespace`

`-n my-namespace`

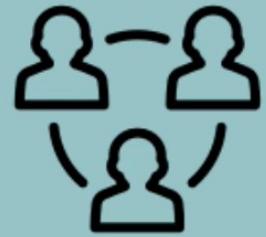
## Change active namespace

Change the active namespace with kubens!



Only namespace  
"my-namespace"  
allowed

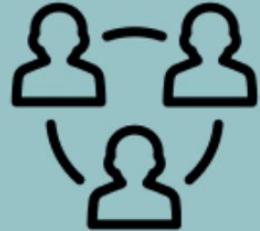
# Change active namespace



**Only namespace  
"my-namespace"  
allowed**

A screenshot of a macOS Terminal window. The window title bar says "Terminal". The menu bar includes "Terminal", "Shell", "Edit", "View", "Window", and "Help". The status bar at the bottom right shows the path "git-remote-https ~ brew.sh install kubectx" and the dimensions "89x24". The main terminal area contains the command "[~]\$ brew install kubectx" in white text on a dark background.

# Change active namespace

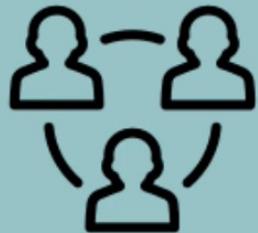


**Only namespace  
"my-namespace"  
allowed**

A screenshot of a macOS Terminal window titled "Terminal". The window shows the command `kubens` followed by a list of Kubernetes namespaces: default, kube-node-lease, kube-public, kube-system, and kubernetes-dashboard. A yellow box highlights the first namespace, "default". The terminal window has a dark background and a light gray border. The status bar at the bottom right shows the path `~ — -bash — 89x24`.

```
[~]$ kubens
default
kube-node-lease
kube-public
kube-system
kubernetes-dashboard
[~]$
```

# Change active namespace



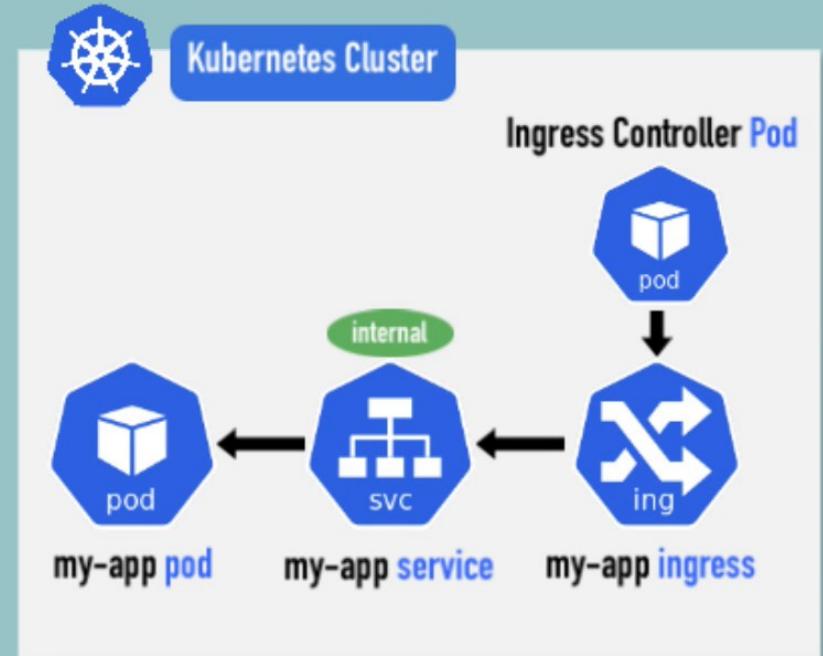
**Only namespace  
"my-namespace"  
allowed**

```
Terminal Shell Edit View Window Help
[~]$ 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    []
[~]$
```

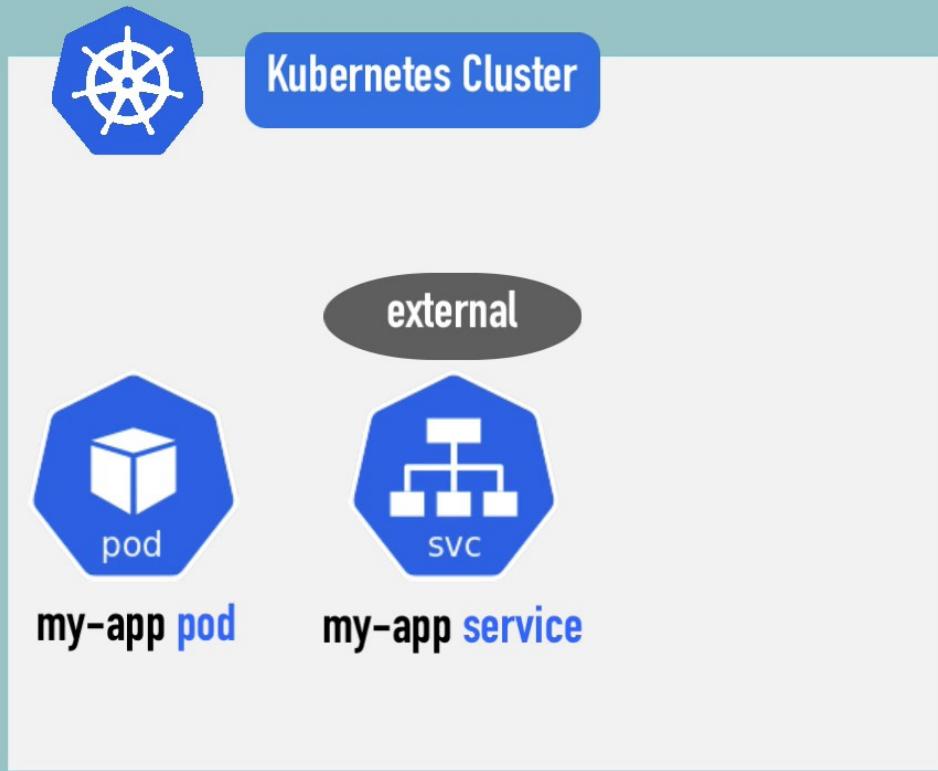


# Kubernetes Ingress explained

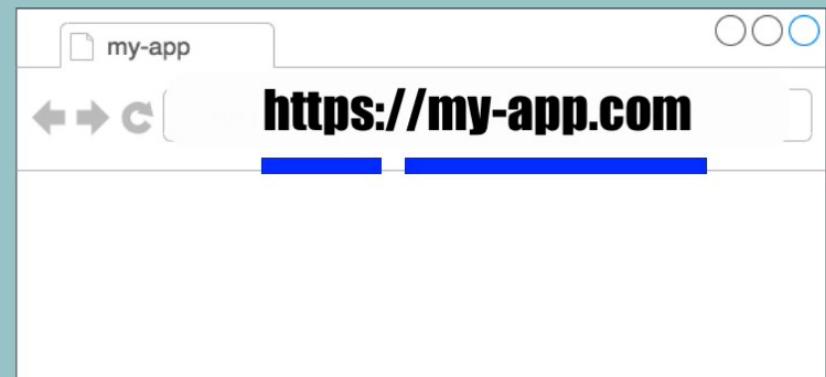
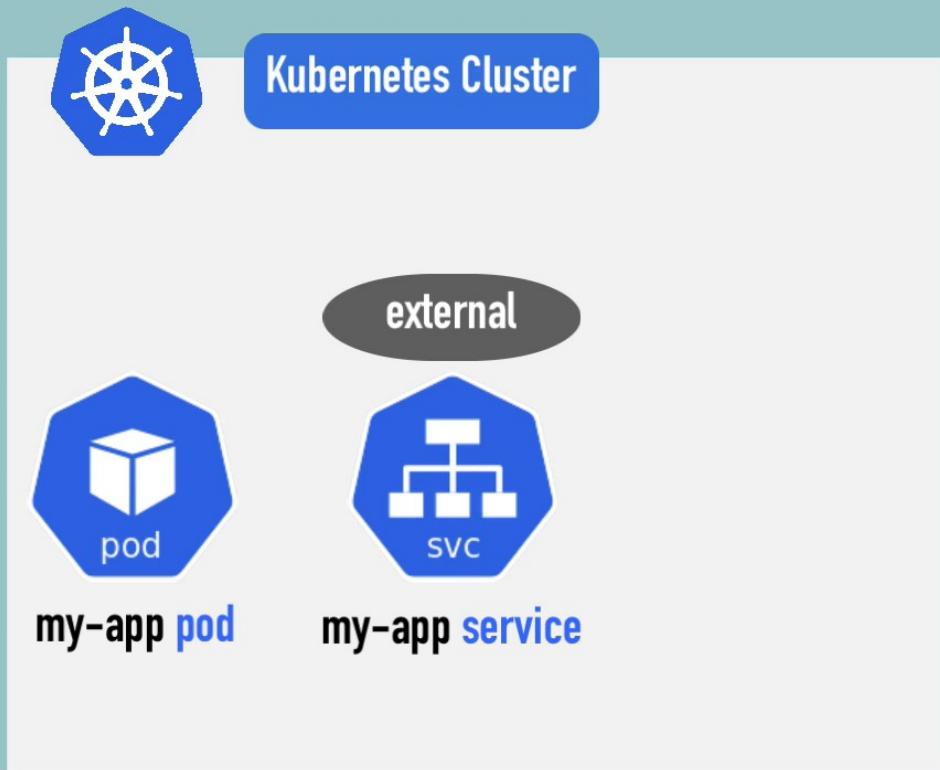
- What is Ingress? 🤔
- Ingress YAML Configuration
- When do you need Ingress?
- Ingress Controller



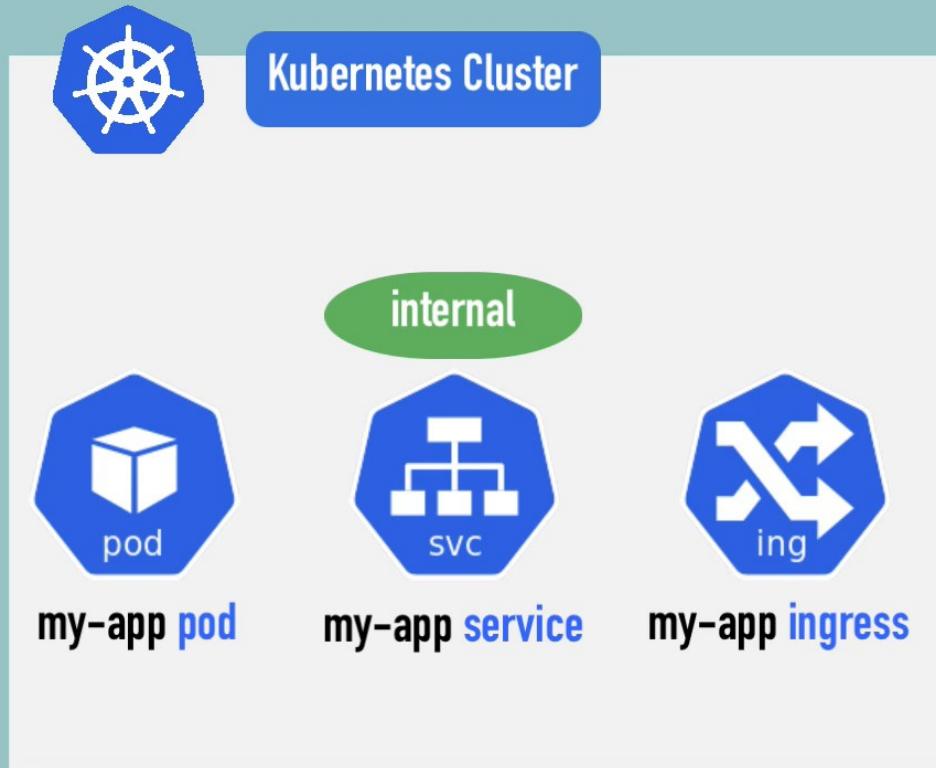
# External Service vs. Ingress



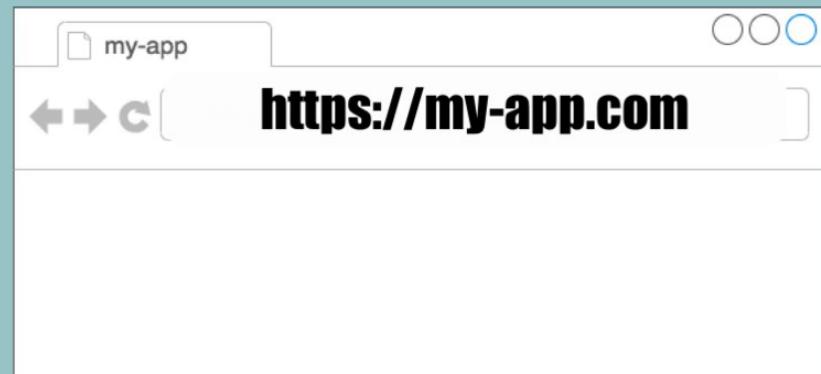
# External Service vs. Ingress



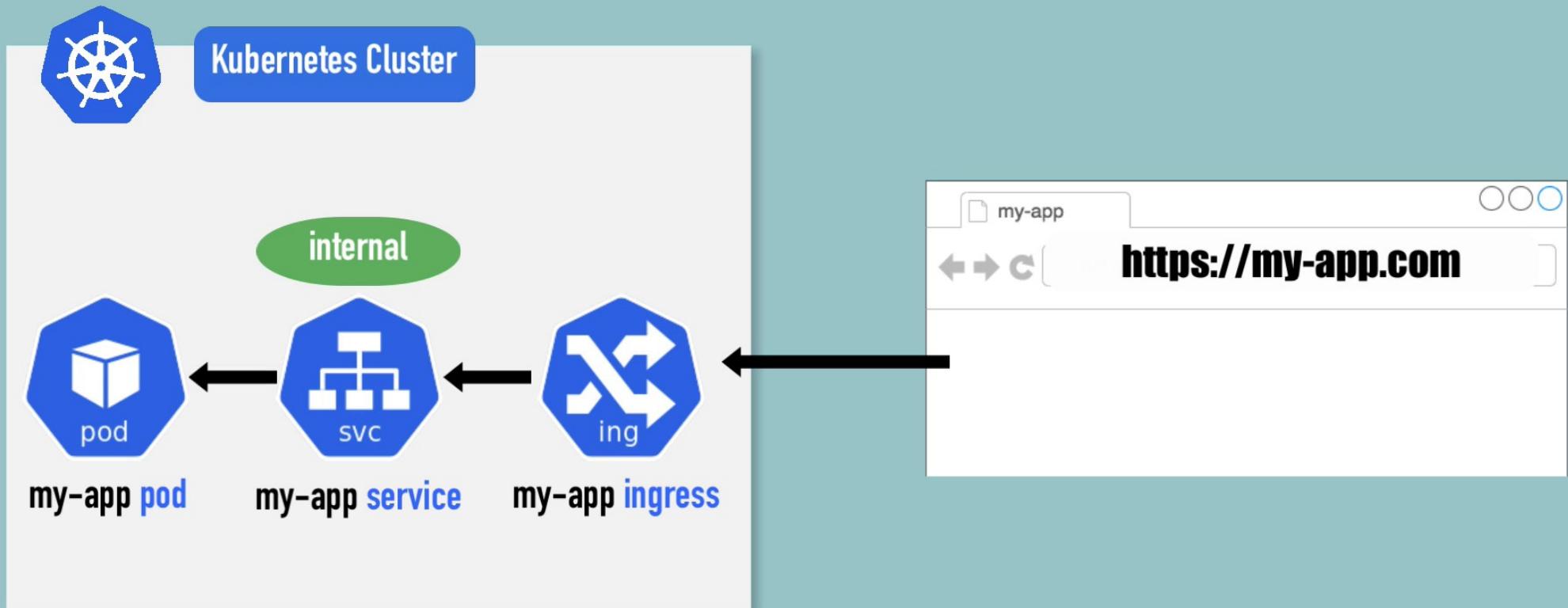
# External Service vs. Ingress



**IP address and port is not opened!**



# External Service vs. Ingress



# 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: 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
```



# 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
```

## Routing rules:

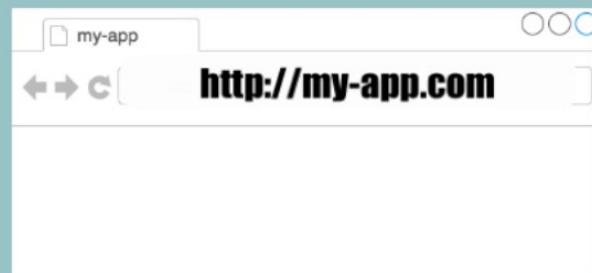
**Forward request to the internal service.**

# 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
```

## Routing rules:

**Forward request to the internal service.**

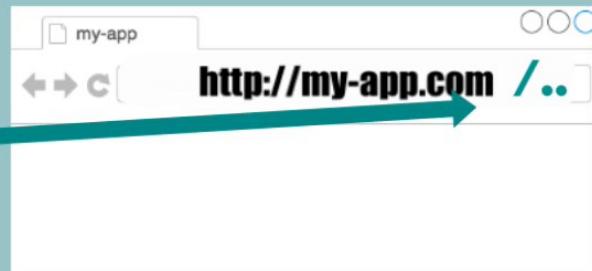


# Example YAML File: Ingress

```
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
```

## Routing rules:

**Forward request to the internal 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
```

## Routing rules:

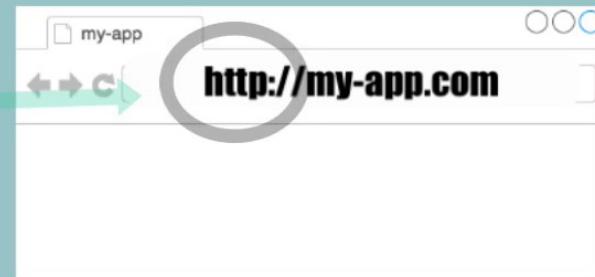
**Forward request to the internal service.**



**Nothing configured for https yet!**

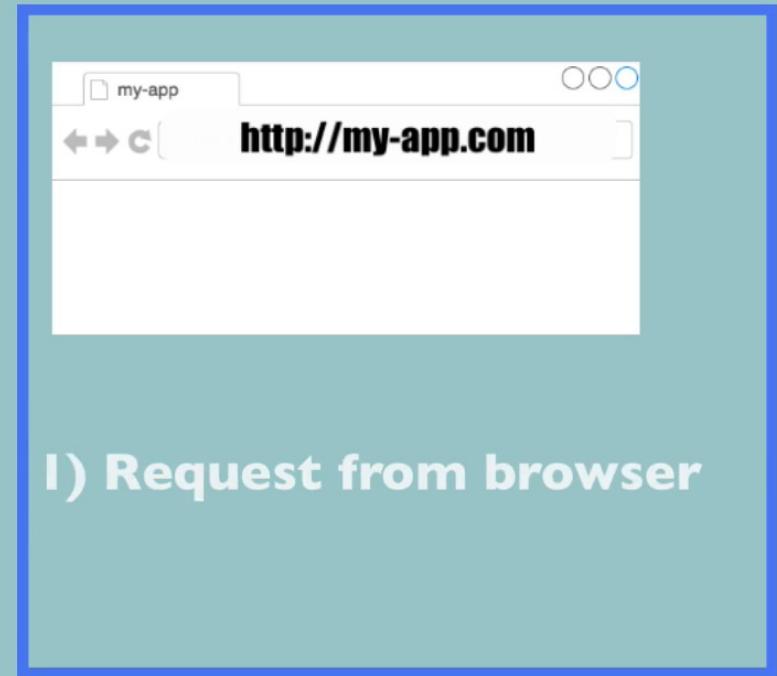
# Example YAML File: Ingress

```
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
```

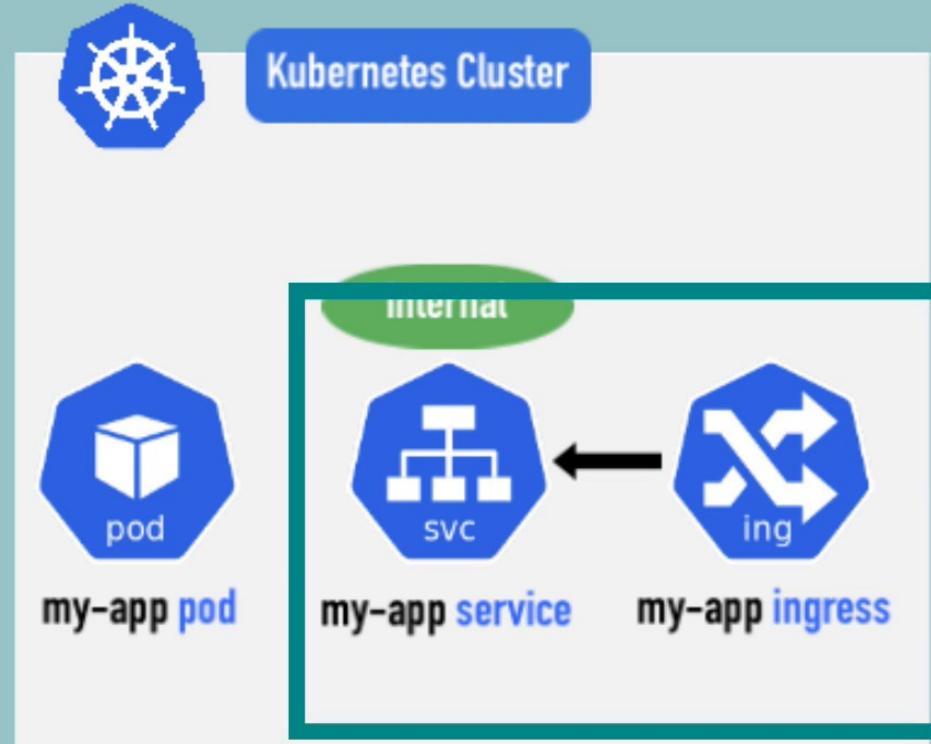


# Example YAML File: Ingress

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: myapp-ingress
spec:
  rules:
  - host: myapp.com
    http: = 2. Step: Incoming Request
      paths: gets forwarded to internal
        - backend:
            serviceName: myapp-internal-service
            servicePort: 8080
```



# Ingress and Internal Service configuration



# Ingress and Internal Service configuration

## 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
```

## Internal Service:

```
apiVersion: v1
kind: Service
metadata:
  name: myapp-internal-service
spec:
  selector:
    app: myapp
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

# Ingress and Internal Service configuration

## 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
```

## Internal Service:

```
apiVersion: v1
kind: Service
metadata:
  name: myapp-internal-service
spec:
  selector:
    app: myapp
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 8080
```

# Ingress and Internal Service configuration

## Ingress:

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

## Internal Service:

```
apiVersion: v1
kind: Service
metadata:
  name: myapp-internal-service
spec:
  selector:
    app: myapp
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 8080
```

**No nodePort in internal service!**

**Instead of Loadbalancer default type: ClusterIP**

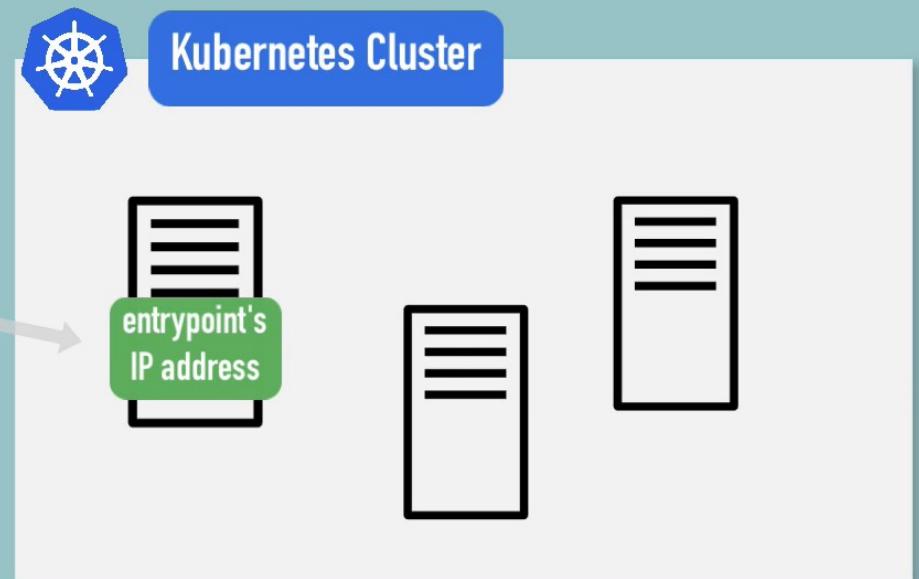
# Ingress and Internal Service configuration

## 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
```

## Host:

- valid domain address
- map domain name to Node's IP address, which is the entrypoint



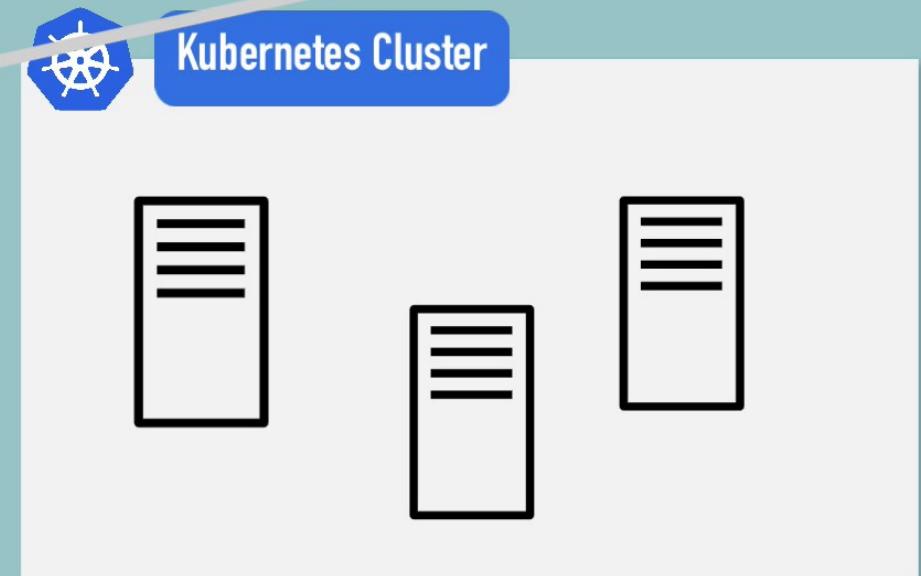
# Ingress and Internal Service configuration

## Ingress:

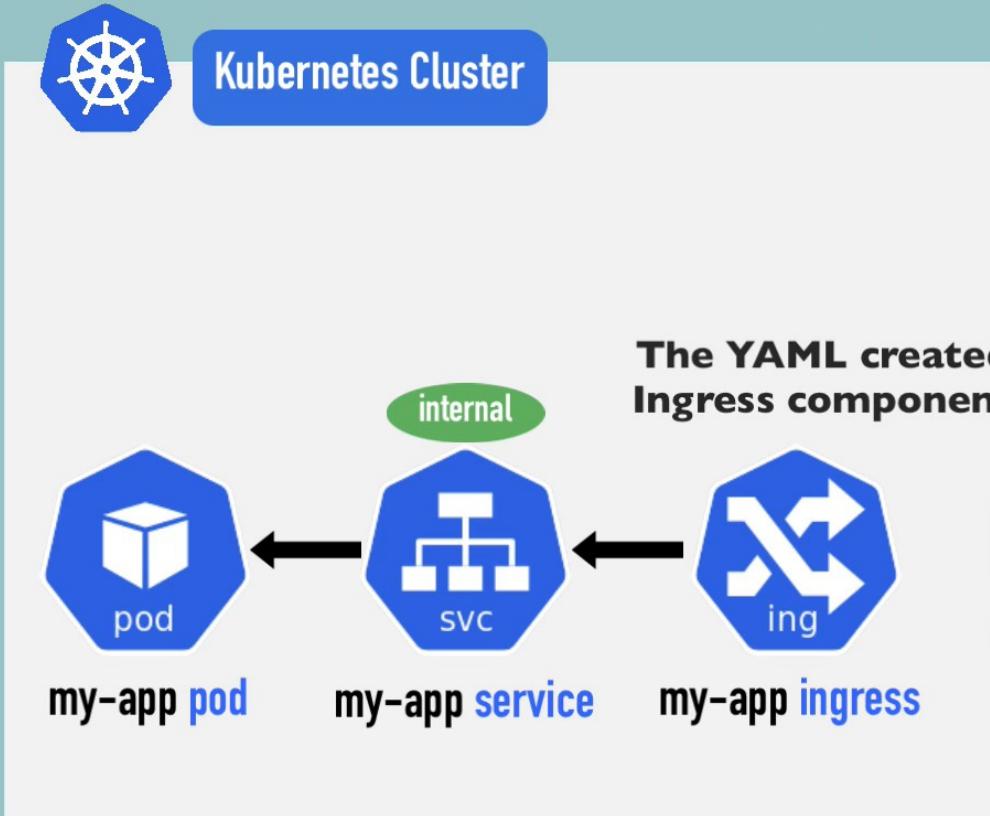
```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: myapp-ingress
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - backend:
          serviceName: myapp-internal-service
          servicePort: 8080
```

## Host:

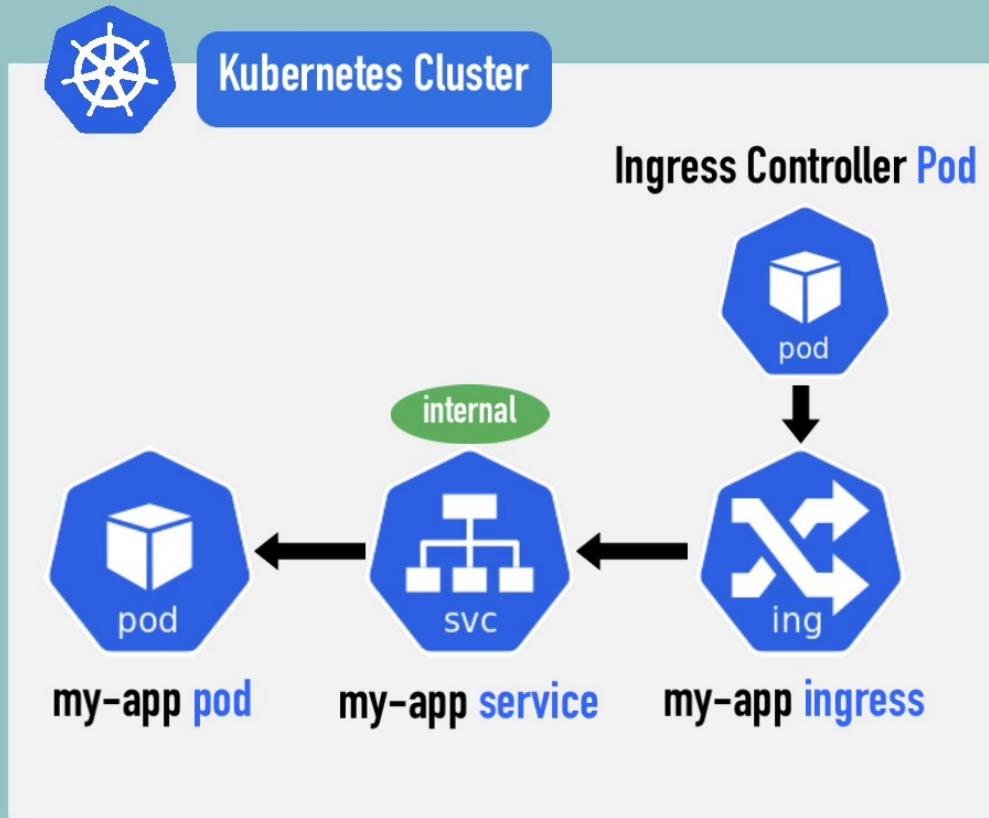
- valid domain address
- map domain name to Node's IP address, which is the endpoint's IP address



# How to configure Ingress in your Cluster?



# How to configure Ingress in your Cluster?

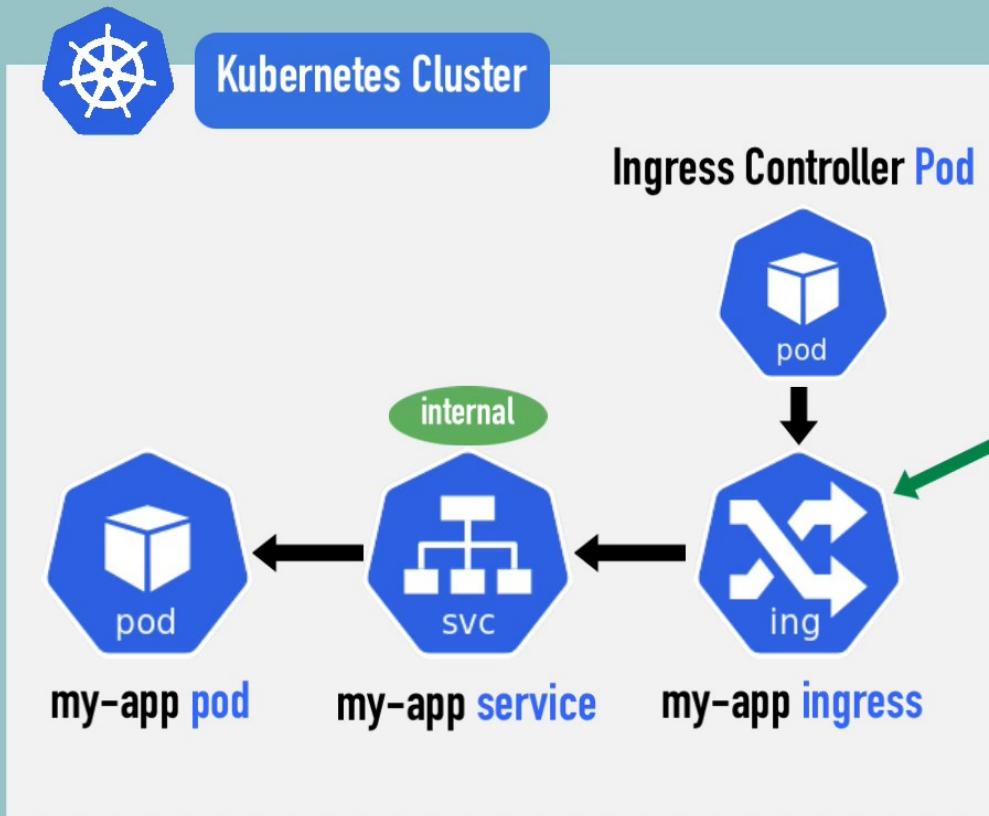


You need an **implementation** for Ingress!

Which is **Ingress Controller**

- evaluates and processes Ingress rules

# What is Ingress Controller? 🤔

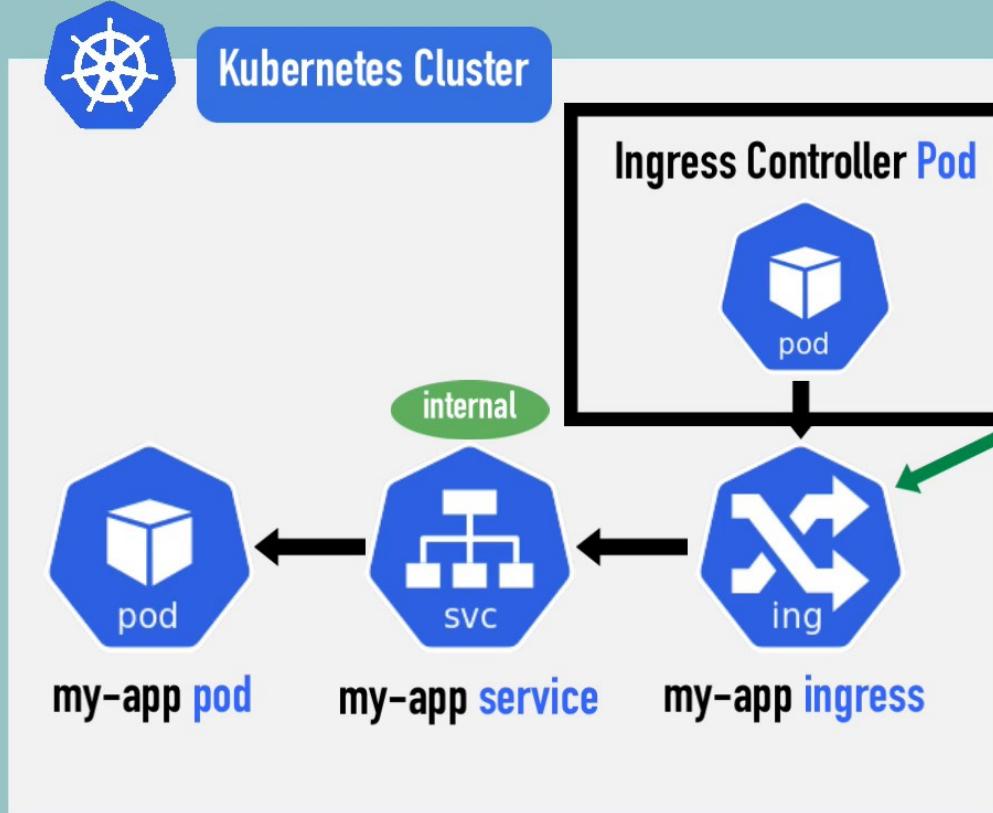


You need an **implementation** for Ingress!

Which is Ingress Controller

```
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
```

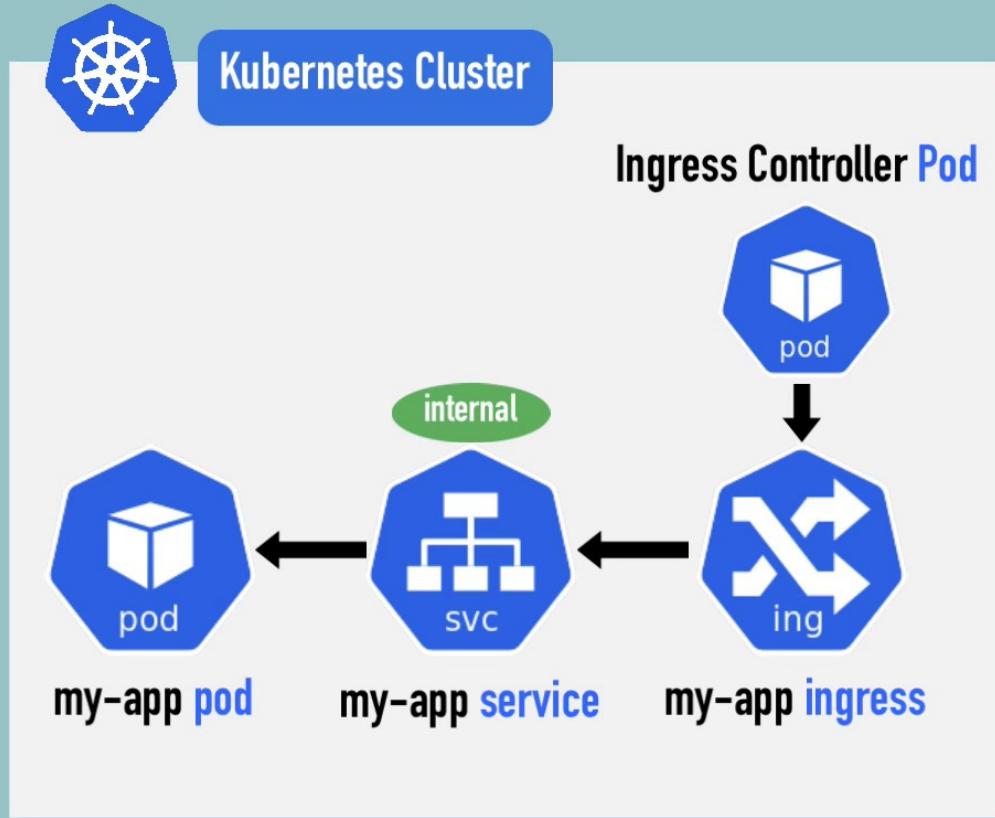
# What is Ingress Controller?



- evaluates all the rules
- manages redirections
- entrypoint to cluster

```
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
```

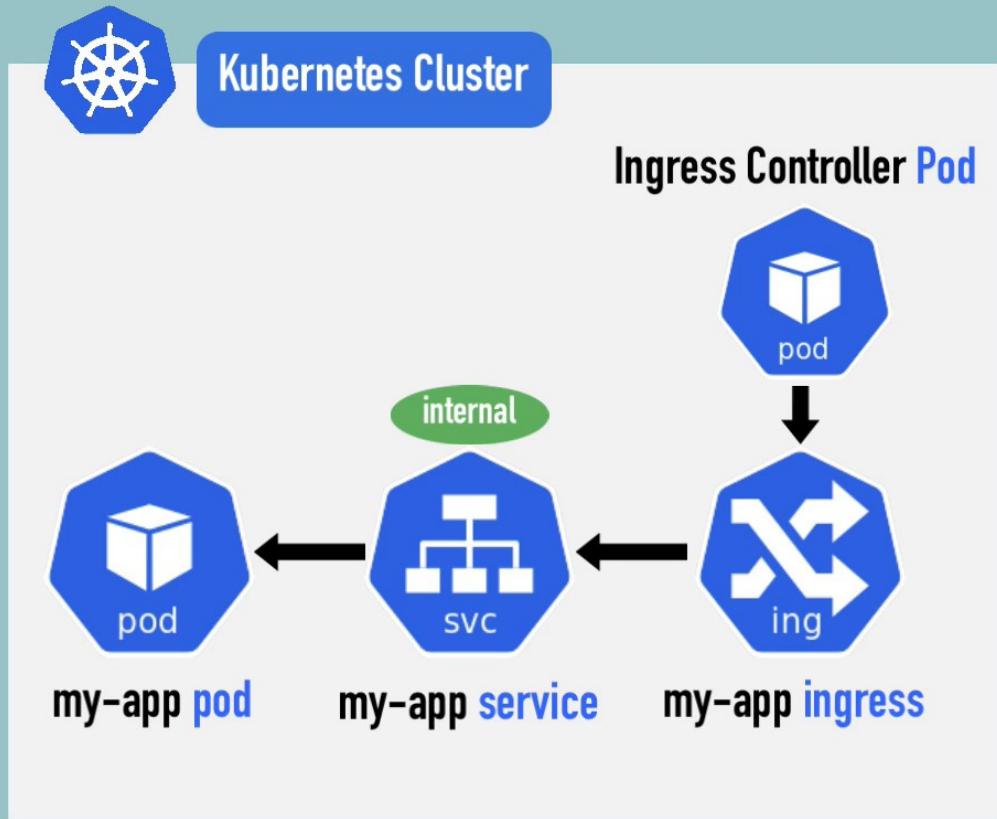
# What is Ingress Controller?



- evaluates all the rules
- manages redirections
- entrypoint to cluster
- many third-party implementations
- K8s Nginx Ingress Controller



# Environment on which your cluster runs



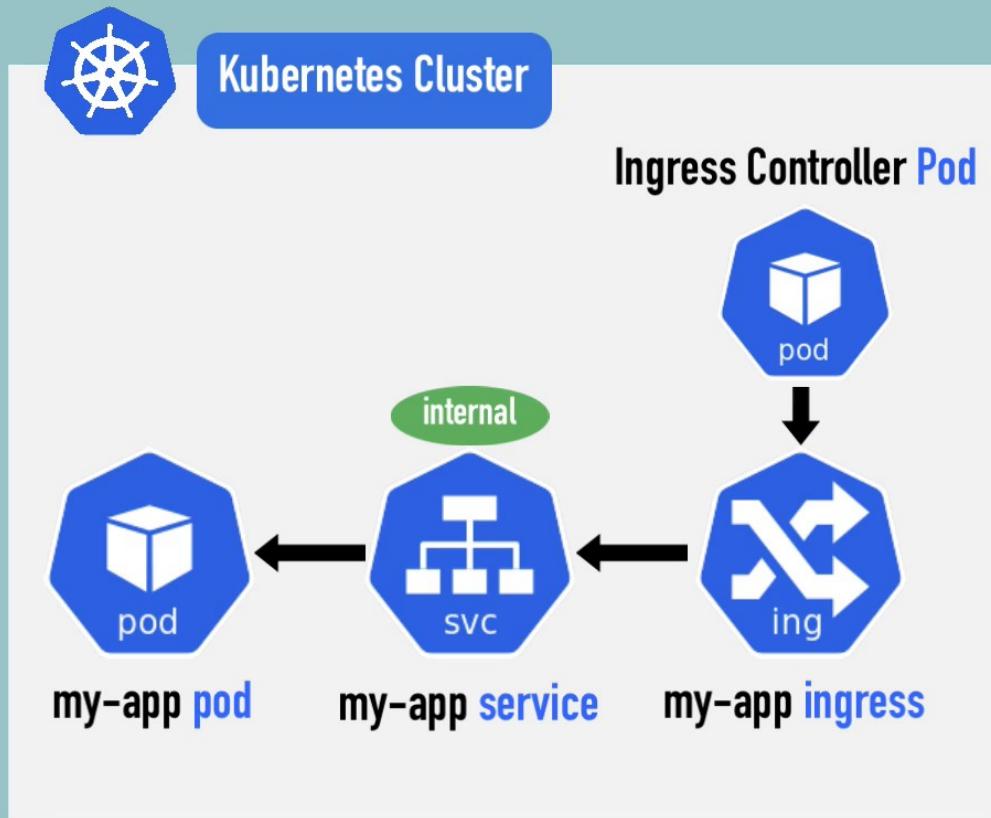
**Cloud service provider:**

**AWS**

**Google Cloud**

**Linode**

# Environment on which your cluster runs

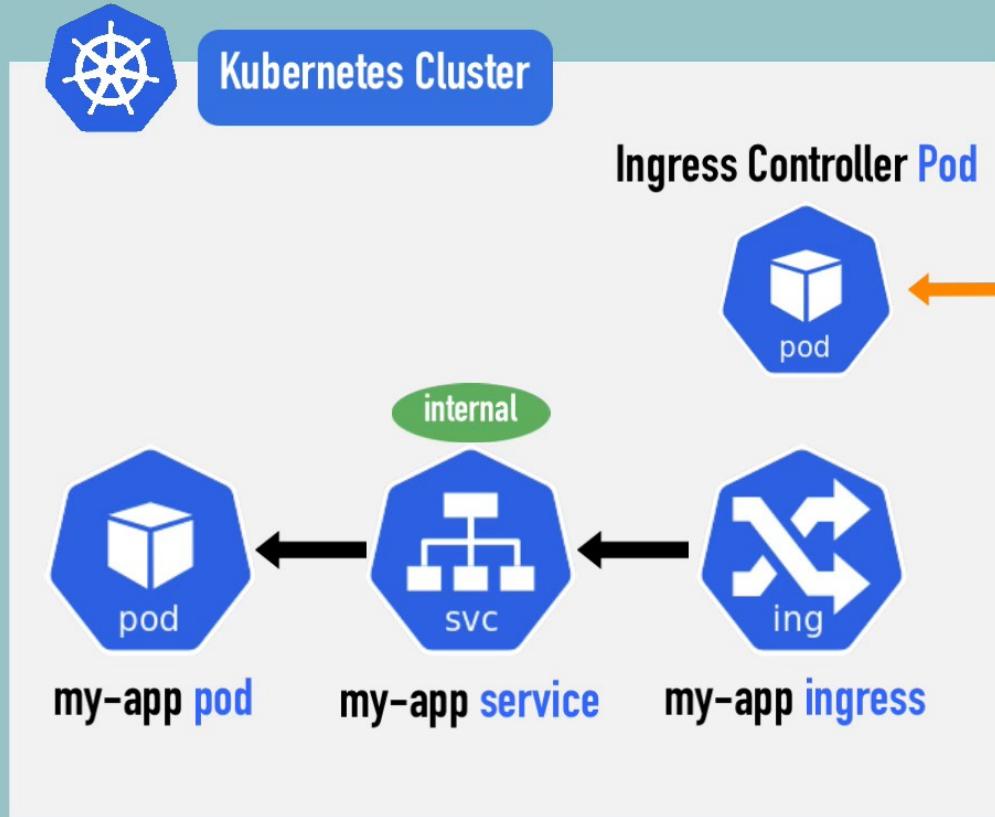


**Cloud service provider:**

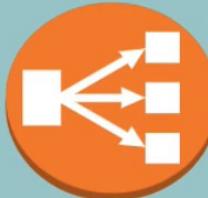
**Out-of-the-box K8s  
solutions**

**Own virtualized Load  
Balancer**

# Environment on which your cluster runs

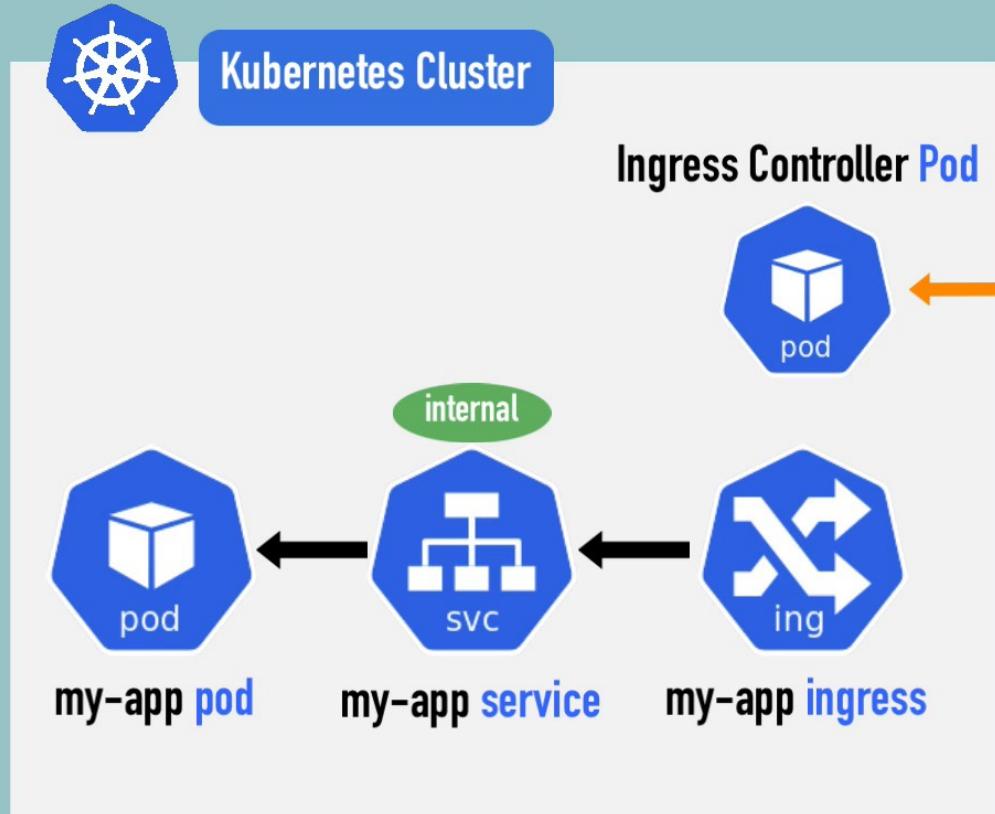


Cloud Load Balancer



You can configure it in  
different ways!

# Environment on which your cluster runs



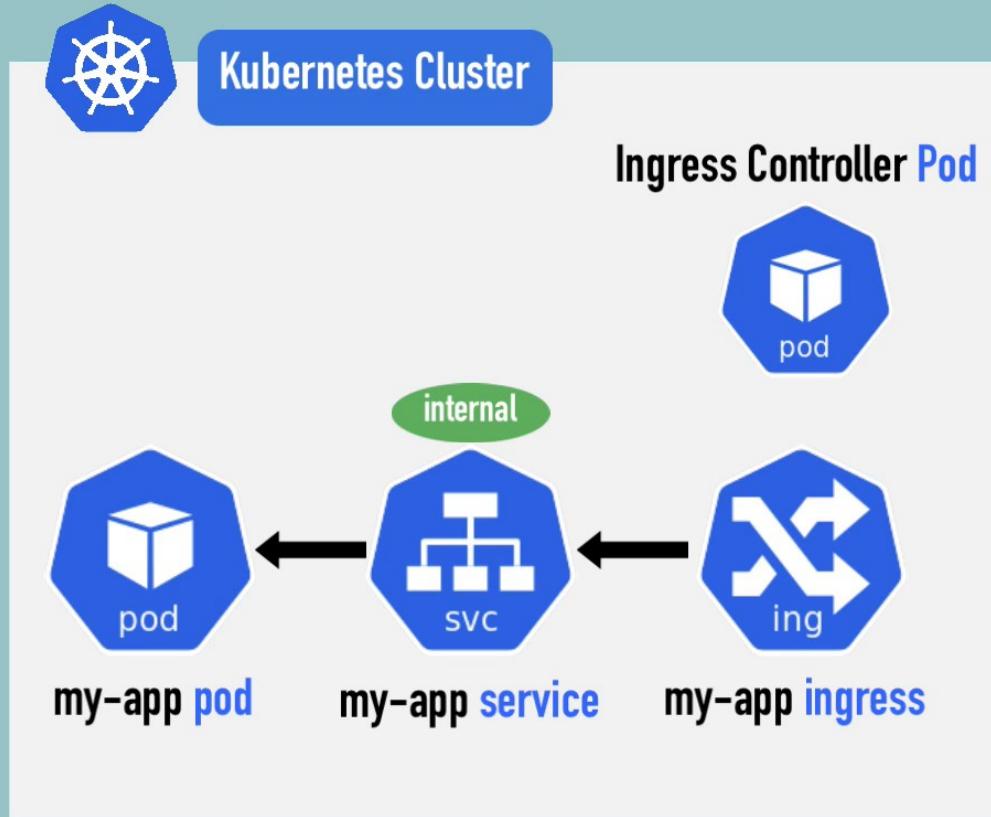
Cloud Load Balancer



**Advantage:**

**You don't have to implement load balancer by yourself!**

# Environment on which your cluster runs

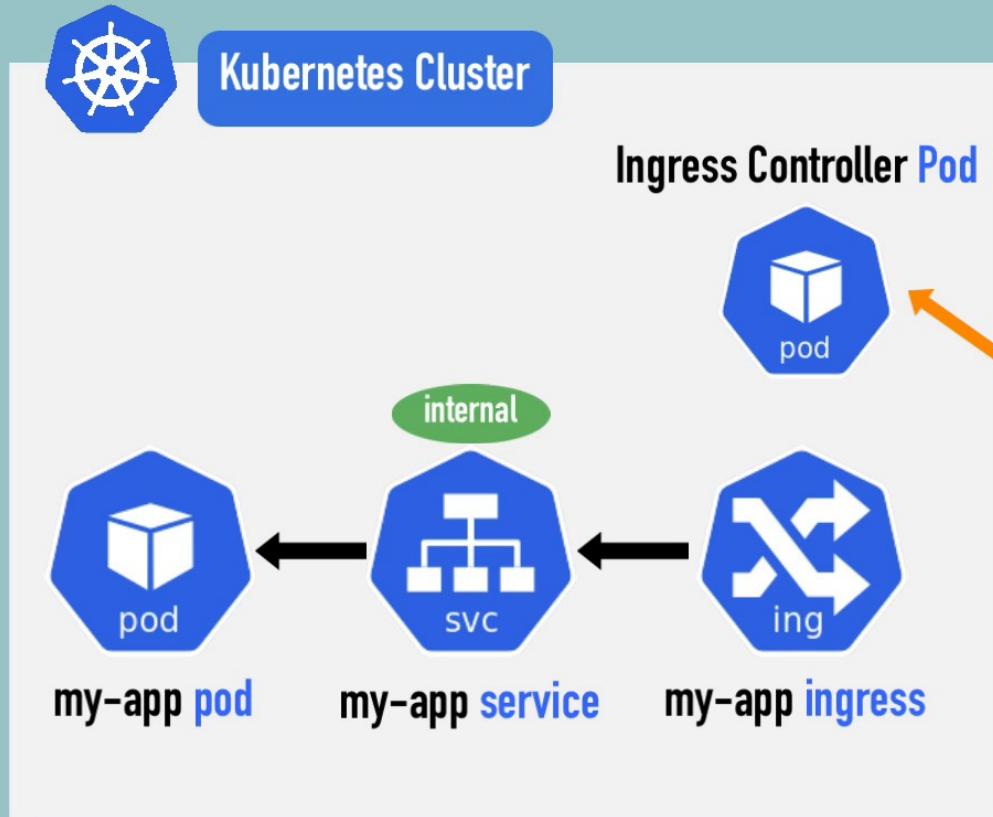


## Bare Metal:

You need to configure some kind of entrypoint

Either inside of cluster or outside as separate server

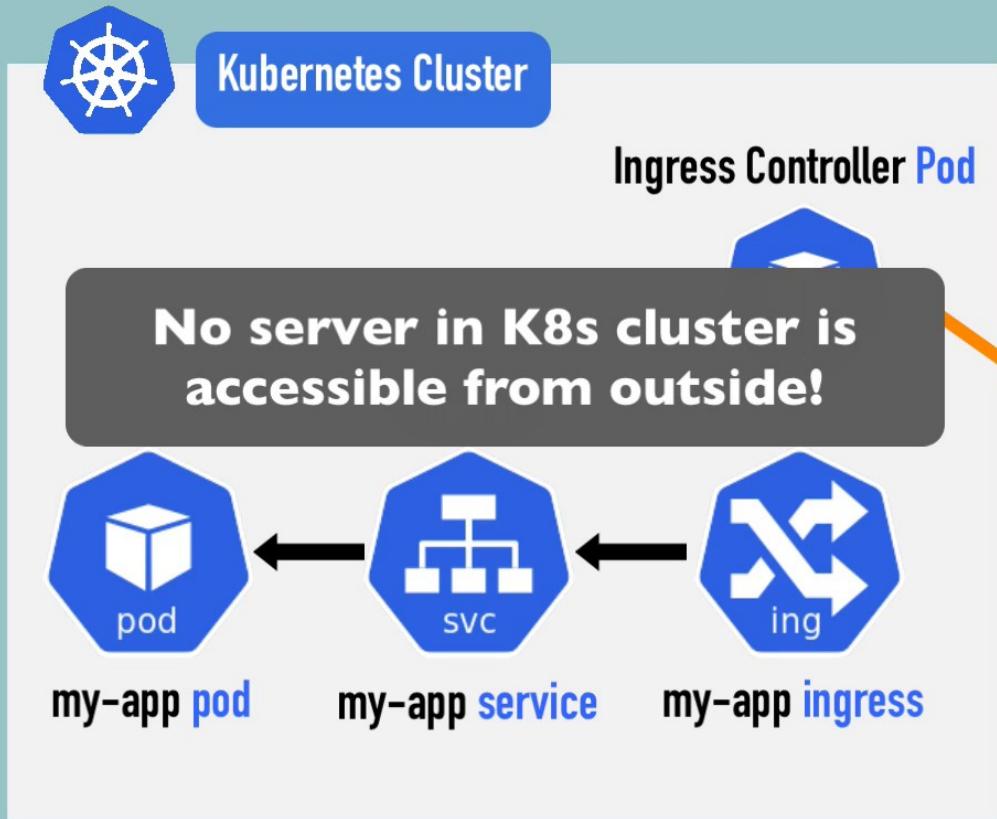
# Environment on which your cluster runs



**Software or hardware solution**



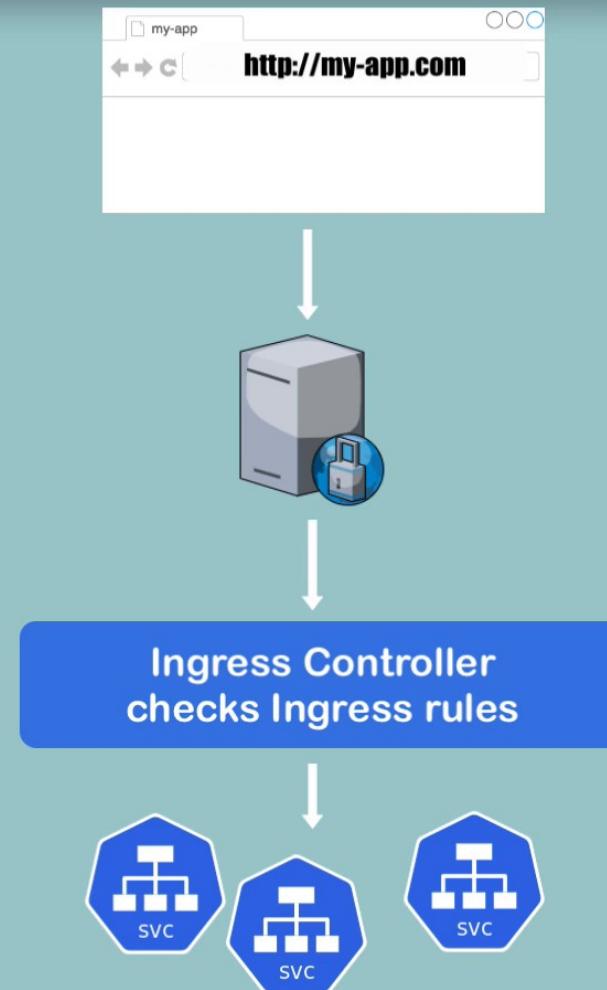
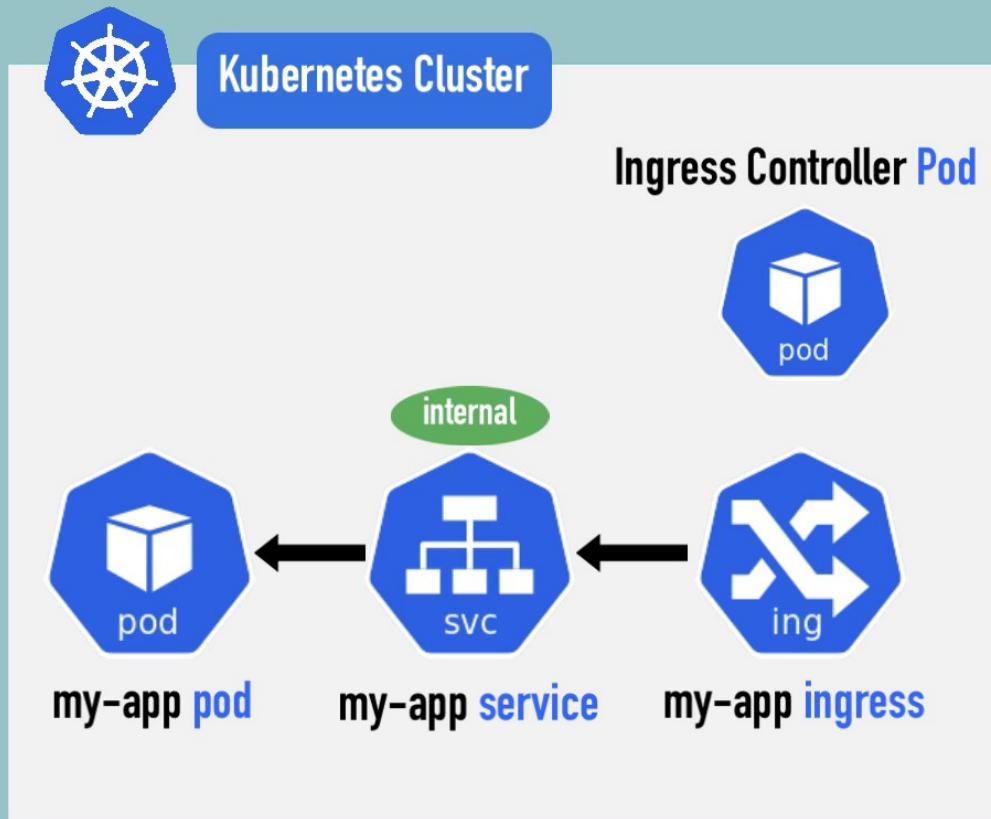
# Environment on which your cluster runs



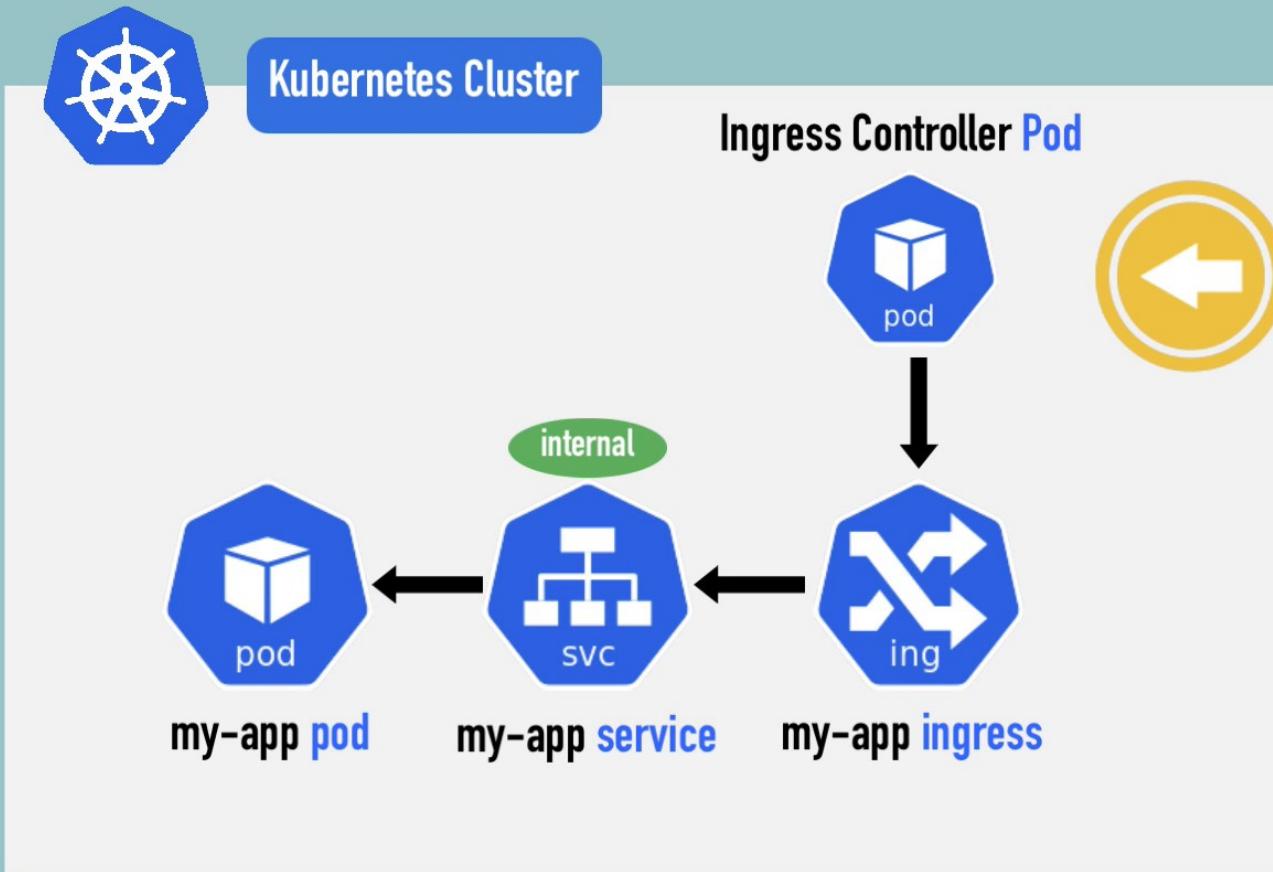
- separate server
- public IP address and open ports
- entrypoint to cluster



# Environment on which your cluster runs



# Ingress Controller in Minikube



**Install Ingress  
Controller in  
Minikube**

# Install Ingress Controller in Minikube



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

**Automatically starts the K8s Nginx implementation of Ingress Controller**

**Ingress Controller configured with just 1 command :)**

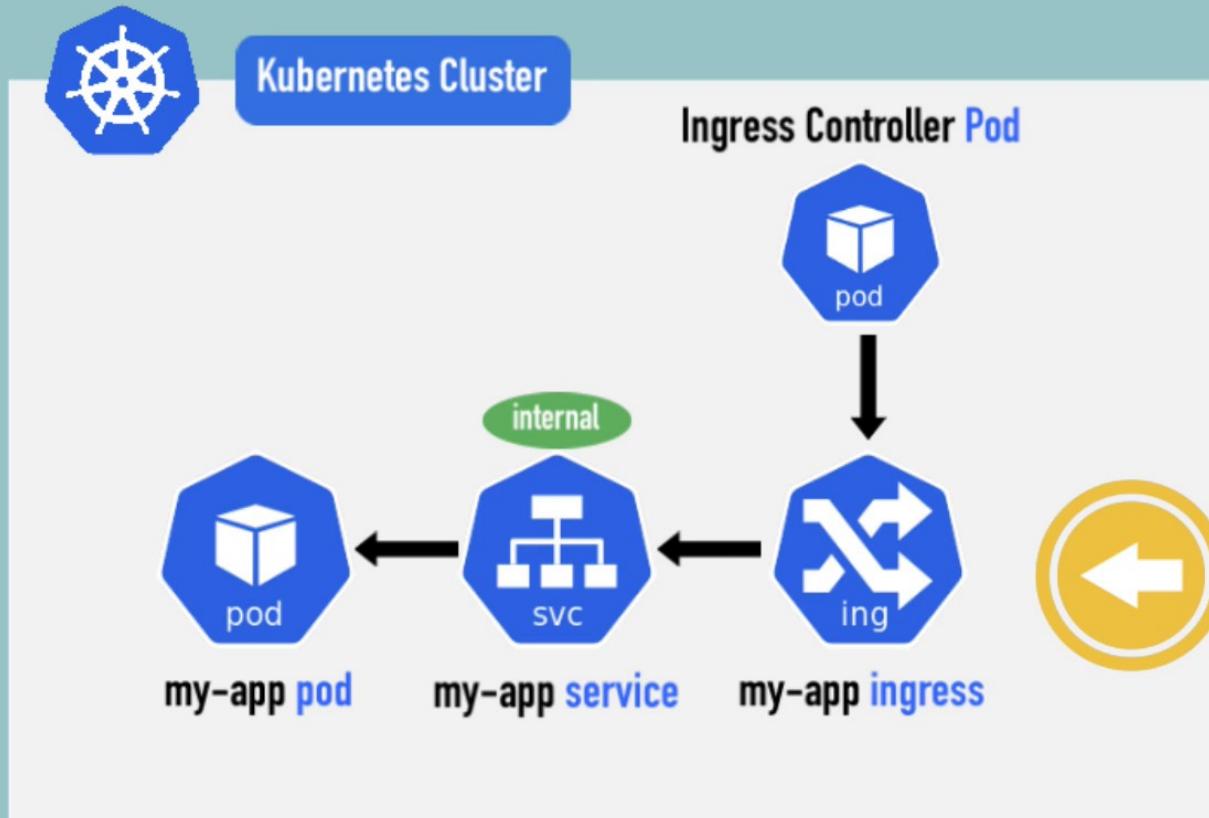


# Install Ingress Controller in Minikube

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

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	40	22d
nginx-ingress-controller-6fc5bcc8c9-wd4g9	1/1	Running	0	30h
storage-provisioner	1/1	Running	0	32h

# Install Ingress Controller in Minikube



**Create Ingress rule**

```
[~]$ kubectl get ns
NAME          STATUS  AGE
default        Active  23d
kube-node-lease  Active  23d
kube-public    Active  23d
kube-system    Active  23d
kubernetes-dashboard  Active  17d
[~]$
```

In Minikube cluster kubernetes-dashboard already exists out-of-the-box

```
[~]$ kubectl get ns
NAME          STATUS  AGE
default       Active  23d
kube-node-lease  Active  23d
kube-public    Active  23d
kube-system    Active  23d
kubernetes-dashboard  Active  17d
[~]$
```

**BUT not accisible externally!**



```
[~]$ kubectl get ns
```

NAME	STATUS	AGE
default	Active	23d
kube-node-lease	Active	23d
kube-public	Active	23d
kube-system	Active	23d
<b>kubernetes-dashboard</b>	Active	17d

```
[~]$
```

Internal service and Pod already exists.

kube-node-lease	Active	23d				
kube-public	Active	23d				
kube-system	Active	23d				
kubernetes-dashboard	Active	17d				
[~]\$ 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
[~]\$						

! 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   [ ]
```

**Namespace = same as service and pod!**



## ! 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:
```

```
[~]$ 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
[~]$ clear

[~]$
```

## ! 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|
```

```
[~]$ 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
[~]$ clear

[~]$
```

## ! 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
```

Resolve IP address to host name?

```
[~]$ 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[~]$ sudo vim /etc/hosts
Password: 
```





Search



## Overview

### Cluster

Cluster Roles

Namespaces

Nodes

Persistent Volumes

Storage Classes

### Namespace

default

## Overview

### Workloads

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

### Discovery and Load Balancing

#### Services

Name	Namespace	Labels	Cluster IP	Internal Endpoints	External Endpoints	Age	⋮
kubernetes	default	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	23 days	⋮

1 - 1 of 1 ⌂ ⌂ ⌂ ⌂ ⌂

### Config and Storage

#### Config Maps

Name	Namespace	Labels	Age	⋮
mysql-configmap	default	-	17 days	⋮

1 - 1 of 1 ⌂ ⌂ ⌂ ⌂ ⌂

#### Secrets

Name	Namespace	Labels	Type	Age	⋮
					⋮

```
[~]$ kubectl get ingress -n kubernetes-dashboard --watch
NAME          HOSTS           ADDRESS        PORTS   AGE
dashboard-ingress  dashboard.com  192.168.64.5  80      42s
[c^C[~]$ sudo vim /etc/hosts
Password:
[~]$ 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
---  -----  ---   ----

```

# 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
```

# Recap

- **What is Ingress?** 
- **How you can use it?**
- **Demo how you create an Ingress rule** 

**Let's see more Use Cases** 🤩

# Multiple paths for same host

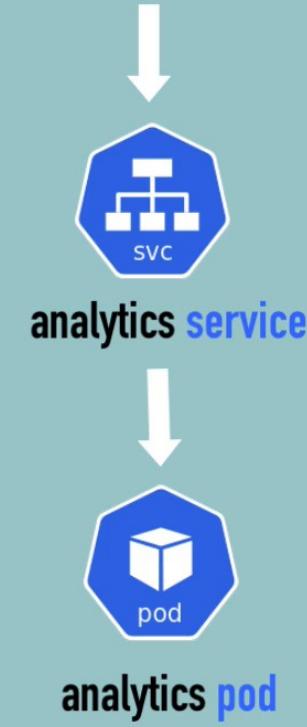
```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
annotations:
  nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - path: /analytics
        backend:
          serviceName: analytics-service
  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

http://myapp.com/analytics





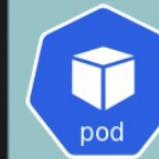
```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: myapp.com
    http:
      paths:
        - path: /analytics
          backend:
            serviceName: analytics-service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

http://myapp.com/analytics

.../shopping



analytics service



analytics pod



shopping service



shopping pod

# 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
          servicePort: 8080
```

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
            servicePort: 8080
```

```
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.**

# 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
          servicePort: 8080
```

http://**analytics.myapp.com**



**analytics service**



**analytics pod**

# Configuring TLS Certificate



# Configuring TLS Certificate - https//



```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tls-example-ingress
spec:
  rules:
    - host: myapp.com
      http:
        paths:
          - path: /
            backend:
              serviceName: myapp-
              servicePort: 8080
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
```

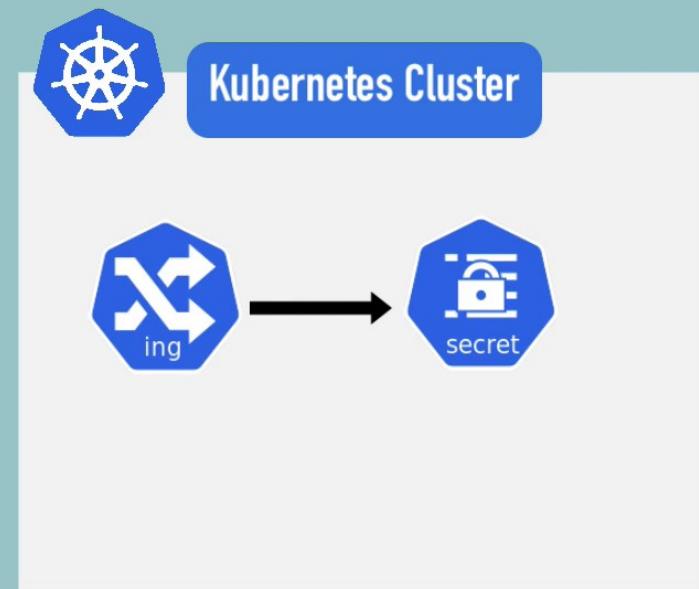
# Configuring TLS Certificate - https://



```
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
```

- **tls**

- **secretName**



# 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
```



```
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
```



```
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: 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
```



- 1) 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**

# 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
```

# Helm explained

- Main concepts of Helm
- Helm changes a lot between versions, so:  
Understand the basic common principles  
and use cases

# Helm explained

## Overview:

- What is Helm?
- What are Helm Charts?
- How to use them?
- When to use them?
- What is Tiller?

# What is Helm?

Package Manager for Kubernetes



apt

yum



# What is Helm?

Package Manager for Kubernetes



To package YAML Files

and distribute them in public and private repositories

# What is Helm?



Kubernetes Cluster



my-app pod



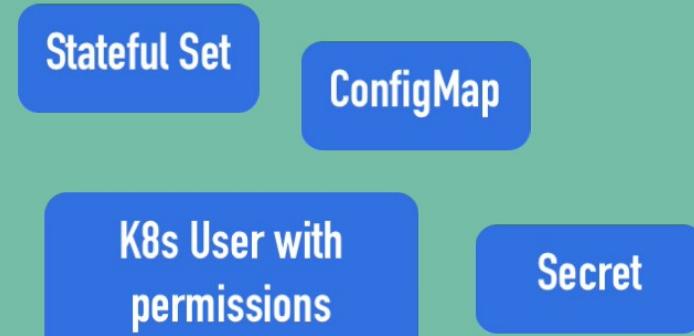
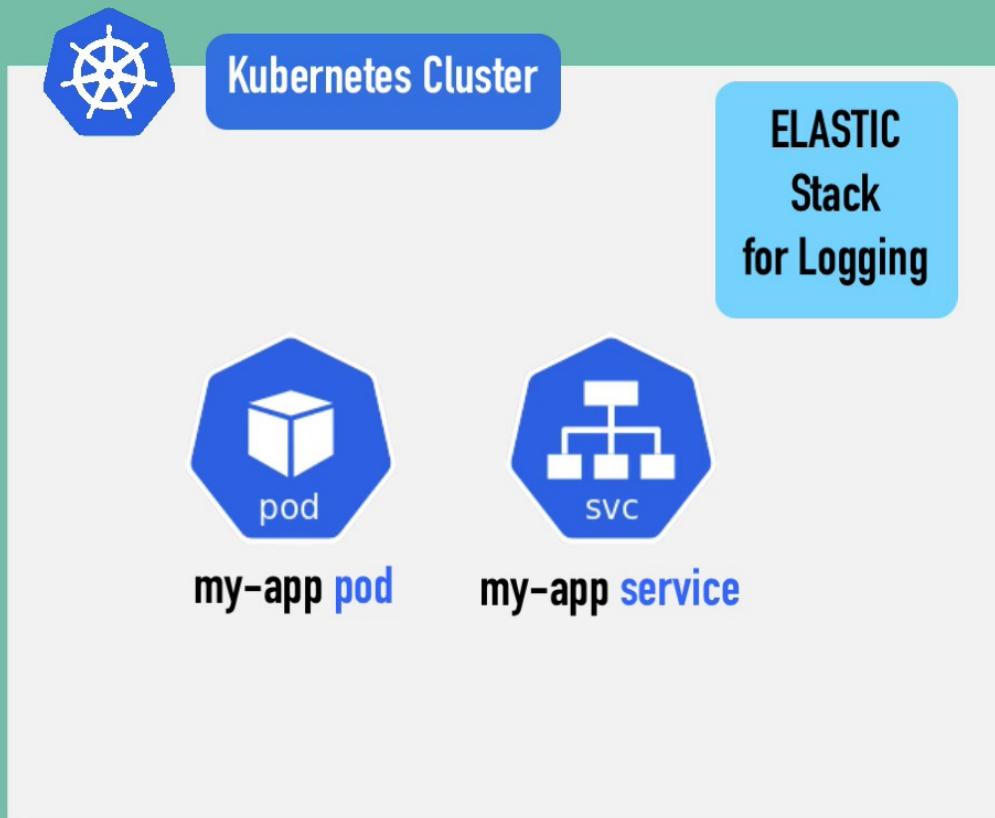
my-app service

ELASTIC  
Stack  
for Logging

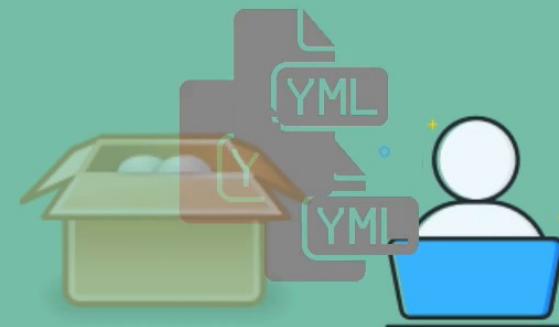
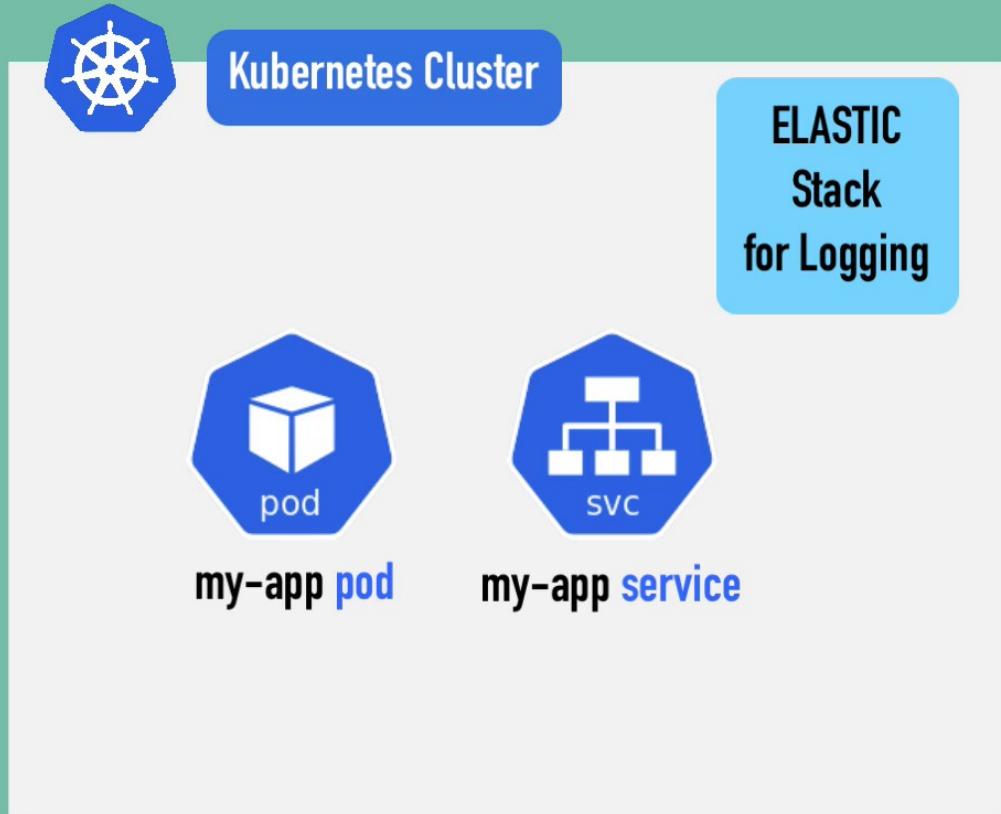
Stateful Set

for stateful apps

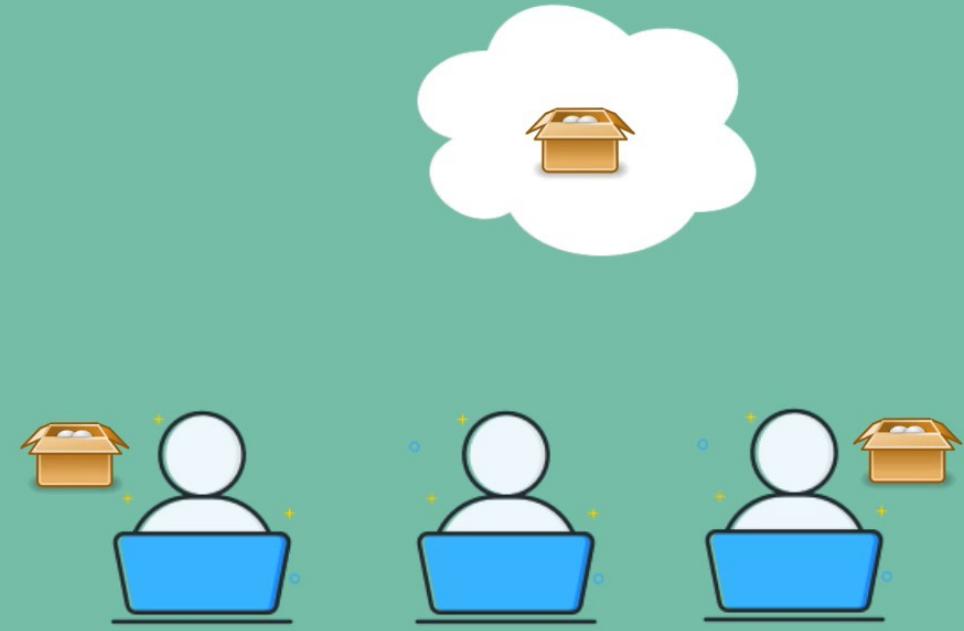
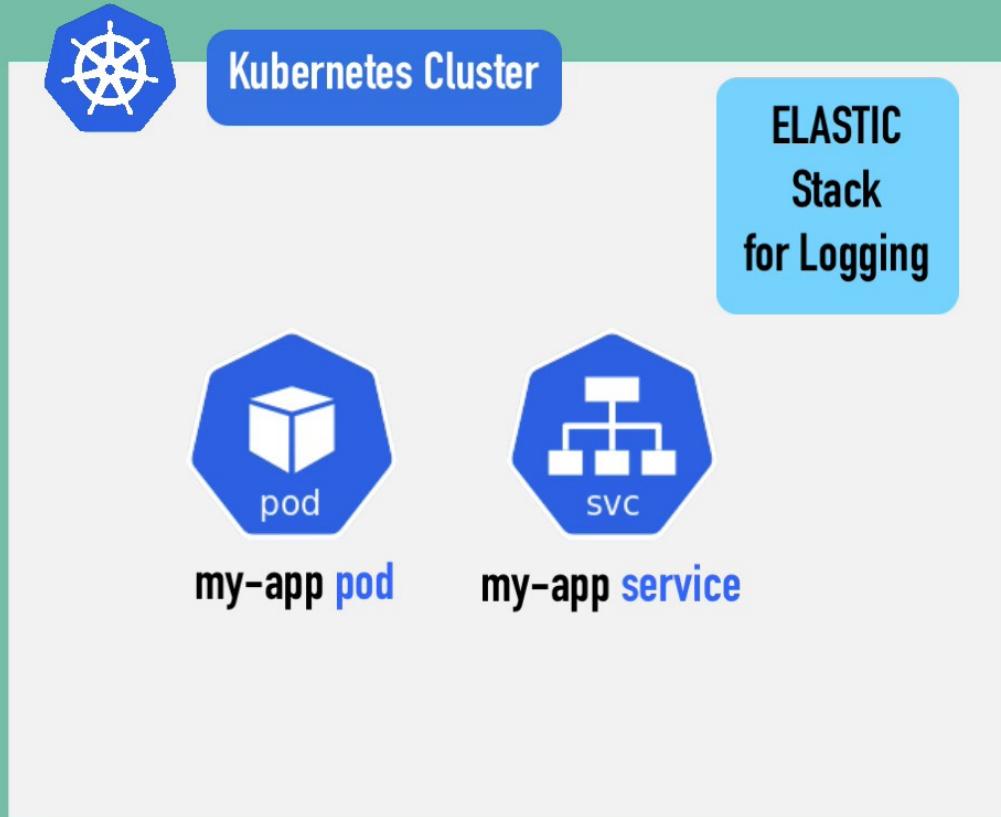
# What is Helm?



# What is Helm?



# What is Helm?



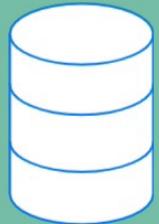
# What is Helm?

## Helm Charts

- Bundle of YAML Files
- Create your own Helm Charts with Helm
- Push them to Helm Repository
- Download and use existing ones

# What is Helm?

## Helm Charts



Database Apps

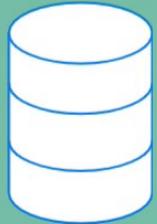
MongoDB

MySQL

Elasticsearch

# What is Helm?

Helm Charts



Database Apps

Monitoring Apps

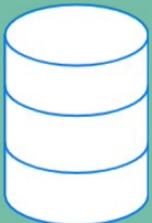


Prometheus

# What is Helm?

Those charts are already available

## Helm Charts



Database Apps

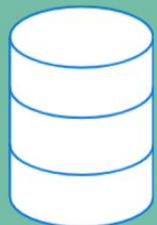
Monitoring Apps



# What is Helm?

So you can reuse that configuration!

## Helm Charts



Database Apps

Monitoring Apps



# Sharing Helm Charts 😎



Kubernetes Cluster

Need some Deployment?



my-app pod



my-app service

`helm search <keyword>`

or Helm Hub:

Discover & launch great  
Kubernetes-ready apps

Search charts...

1140 charts ready to deploy

# Sharing Helm Charts 😎

**Public Registries:**

Discover & launch great  
Kubernetes-ready apps

Search charts...

1140 charts ready to deploy

**Private Registries:** share in organisations

**Second Feature:**

**Templating Engine**

# Templating Engine



Kubernetes Cluster



microservice



microservice



microservice



microservice



microservice

YAML config

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

Deployment and Service configurations almost the same!

# Templating Engine



Kubernetes Cluster



microservice



microservice



microservice



microservice



microservice

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

## YAML config

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

# Templating Engine



Kubernetes Cluster



microservice



microservice



microservice



microservice



microservice

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

Most values are  
the same!

# Templating Engine



Kubernetes Cluster



microservice



microservice

1) Define a common blueprint



microservice



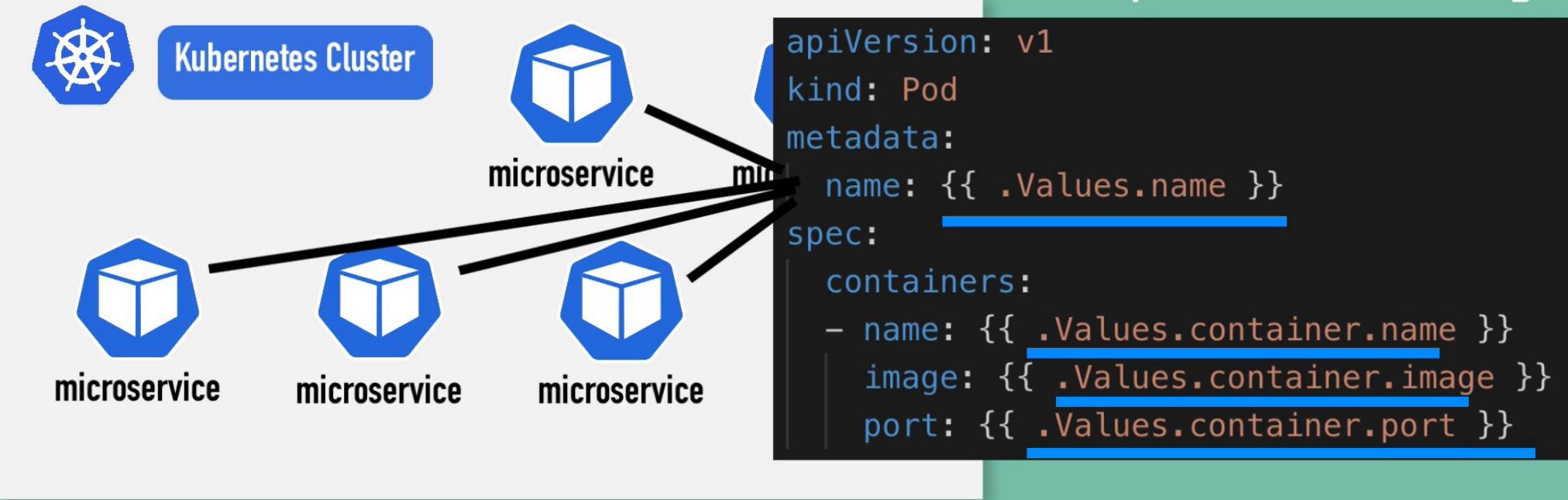
microservice



microservice

2) Dynamic values are replaced by placeholders

# Templating Engine



# Templating Engine

values.yaml

`{{ .Values... }}`

```
name: my-app
container:
  name: my-app-container
  image: my-app-image
  port: 9001
```

Template YAML config

```
apiVersion: v1
kind: Pod
metadata:
  name: {{ .Values.name }}
spec:
  containers:
    - name: {{ .Values.container.name }}
      image: {{ .Values.container.image }}
      port: {{ .Values.container.port }}
```

Object, which is created based on the values defined

# Templating Engine

values.yaml

`{{ .Values... }}`

```
name: my-app
container:
  name: my-app-container
  image: my-app-image
  port: 9001
```

Template YAML config

```
apiVersion: v1
kind: Pod
metadata:
  name: {{ .Values.name }}
spec:
  containers:
    - name: {{ .Values.container.name }}
      image: {{ .Values.container.image }}
      port: {{ .Values.container.port }}
```

Values defined either via yaml file or with --set flag

# Templating Engine

many Yaml Files

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

just 1 Yaml File

```
apiVersion: v1
kind: Pod
metadata:
  name: {{ .Values.name }}
spec:
  containers:
    - name: {{ .Values.container.name }}
      image: {{ .Values.container.image }}
      port: {{ .Values.container.port }}
```

# Templating Engine

## Practical for CI / CD

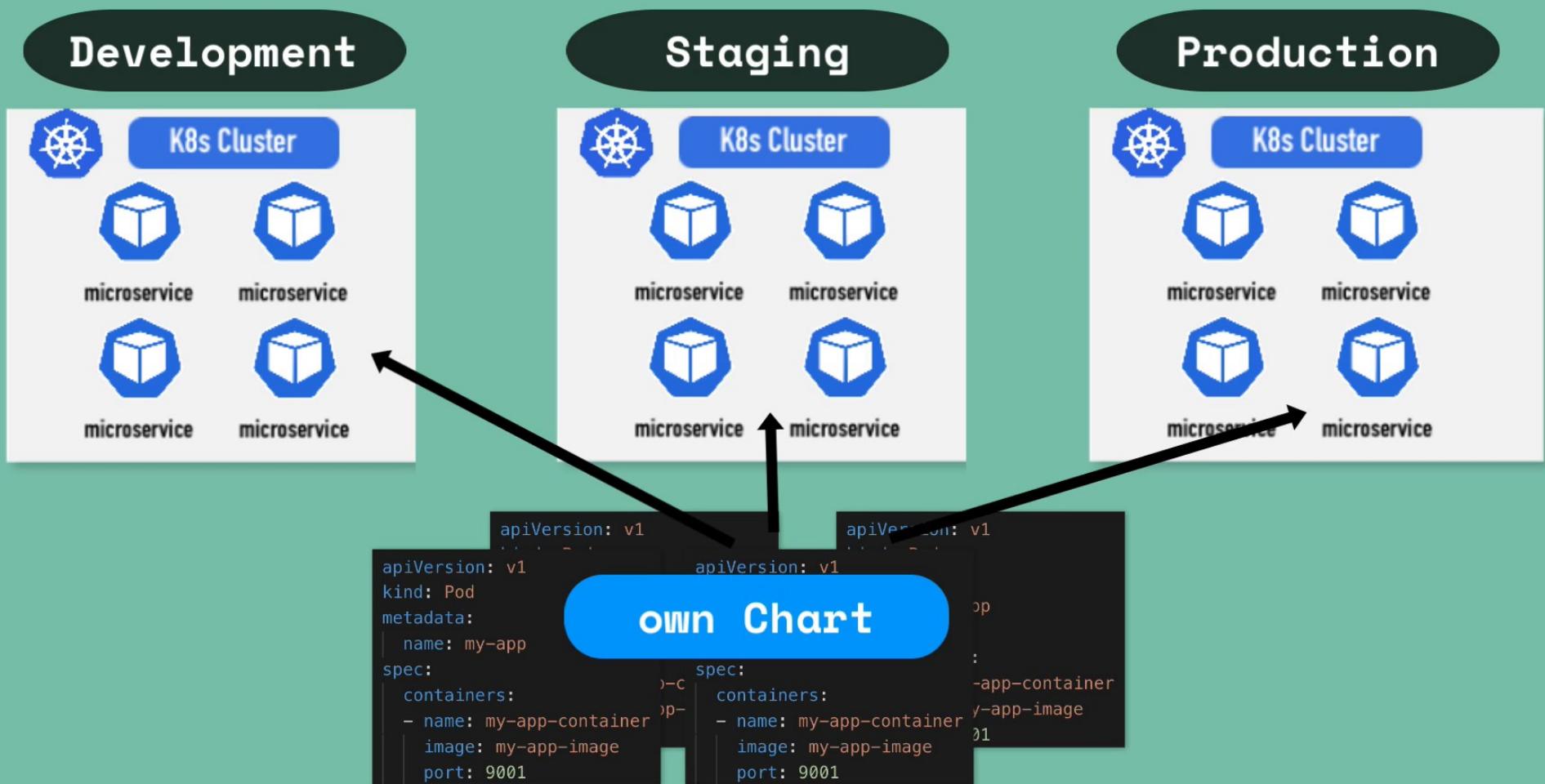
In your Build you can  
replace the values on the  
fly

```
apiVersion: v1
kind: Pod
metadata:
  name: {{ .Values.name }}
spec:
  containers:
    - name: {{ .Values.container.name }}
      image: {{ .Values.container.image }}
      port: {{ .Values.container.port }}
```

## **Another Use Case for Helm features**

**Same Applications across different environments**

# Same Applications across different environments



# Helm Chart Structure

Directory structure:

```
mychart/  
  Chart.yaml  
  values.yaml  
charts/  
templates/  
...
```

Top level **mychart** folder → name of chart

**Chart.yaml** → meta info about chart

**values.yaml** → values for the template files

**charts** folder → chart dependencies

**templates** folder → the actual template files

# Helm Chart Structure

```
mychart/  
  Chart.yaml  
  values.yaml  
  charts/  
  templates/  
  ...
```

Template files will be filled with  
the values from values.yaml

`helm install <chartname>`

# Helm Chart Structure

```
mychart/  
  Chart.yaml  
  values.yaml  
  charts/  
  templates/  
  ...
```

Optionally other files like  
Readme or licence file

# Values injection into template files

**values.yaml**

```
imageName: myapp  
port: 8080  
version: 1.0.0
```

**default**

**my-values.yaml**

```
version: 2.0.0
```

**override values**

```
helm install --values=my-values.yaml <chartname>
```

# Values injection into template files

**values.yaml**

```
imageName: myapp
port: 8080
version: 1.0.0
```

**default**

**my-values.yaml**

```
version: 2.0.0
```

**override values**

**result**

```
imageName: myapp
port: 8080
version: 2.0.0
```

**.Values object**

OR on Command Line:

```
helm install --set version=2.0.0
```

**Third Feature:**

**Release Management**

# Release Management

## Helm Version 2 vs. 3

Helm Version 2 comes in two parts:



CLIENT (helm CLI)

```
helm install <chartname>
```

Sends requests to Tiller

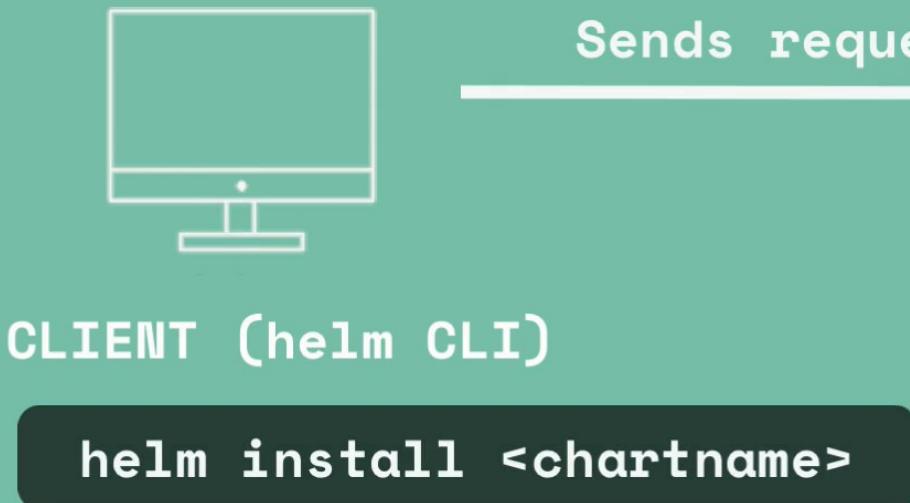


SERVER (Tiller)

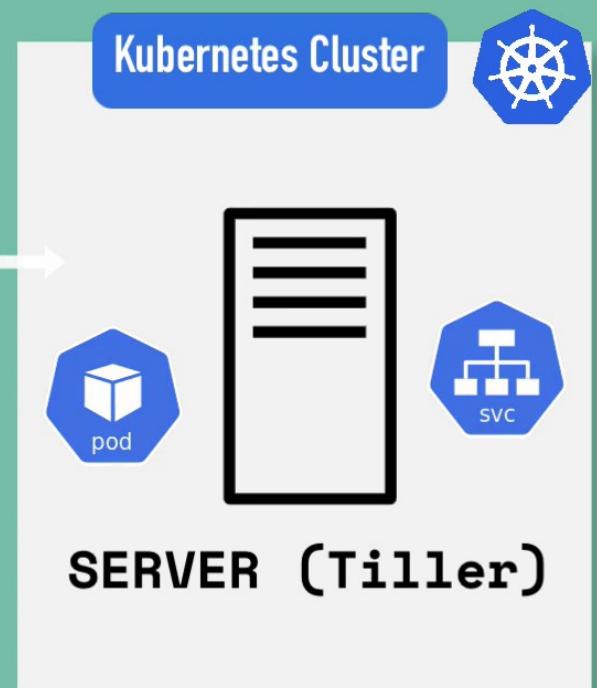
# Release Management

## Helm Version 2 vs. 3

Helm Version 2 comes in two parts:



Sends requests to Tiller



# Release Management

## Create or change deployment

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

*stores copy of configuration*

Kubernetes Cluster



SERVER (Tiller)

# Release Management

Keeping track of all chart executions:

Revision

Request

1

Installed chart

2

Upgraded to v 1.0.0

`helm install <chartname>`

`helm upgrade <chartname>`

Kubernetes Cluster



SERVER (Tiller)

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

# Release Management

Keeping track of all chart executions:

Revision	Request
1	Installed chart
2	Upgraded to v 1.0.0
3	Rolled back to 1

```
helm install <chartname>
```

```
helm upgrade <chartname>
```

```
helm rollback <chartname>
```

- Changes are applied to existing deployment instead of creating a new one
- Handling rollbacks

## Downsides of Tiller

- Tiller has too much power inside of K8s cluster

CREATE



DELETE

UPDATE

## Downsides of Tiller

- Tiller has too much power inside of K8s cluster
- Security Issue
- Solves the Security Concern 👍

In Helm 3 Tiller got removed!



# Kubernetes Volumes explained



How to **persist data** in Kubernetes using volumes?

Persistent Volume



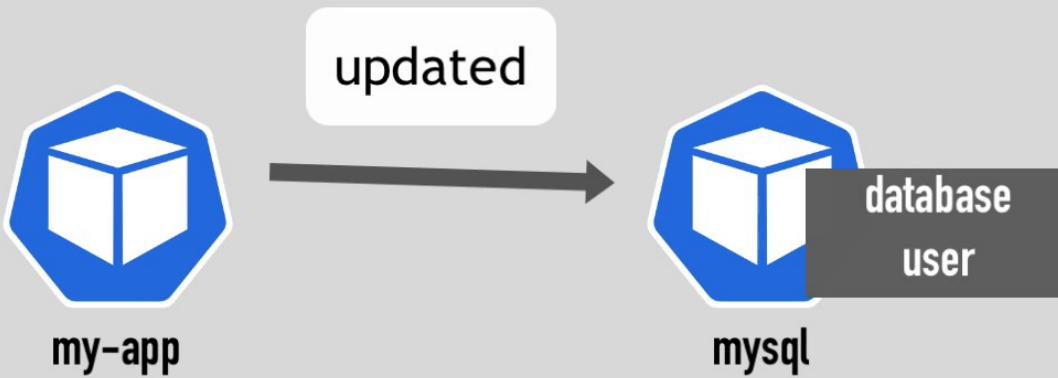
Persistent Volume Claim



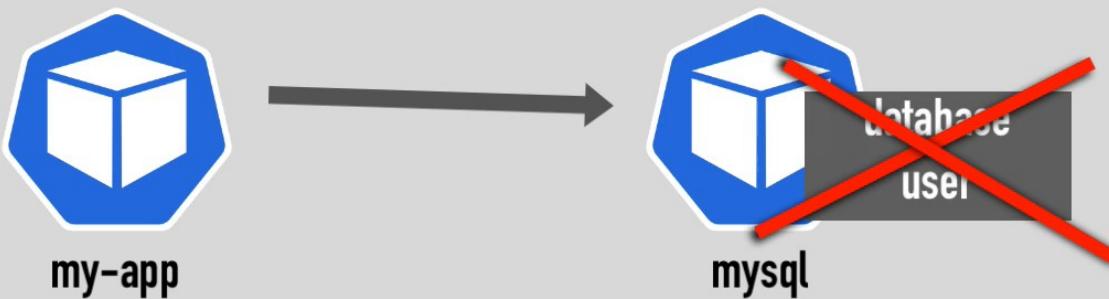
Storage Class



## The need for Volumes



## The need for Volumes



No data persistence out of the box!

## The need for Volumes



You need to configure that

## Storage Requirements



Storage that **doesn't depend on the pod lifecycle**.

## Storage Requirements



my-app



mysql

Storage that **doesn't depend on the pod lifecycle**.

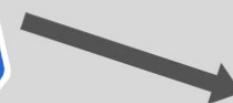
## Storage Requirements



my-app



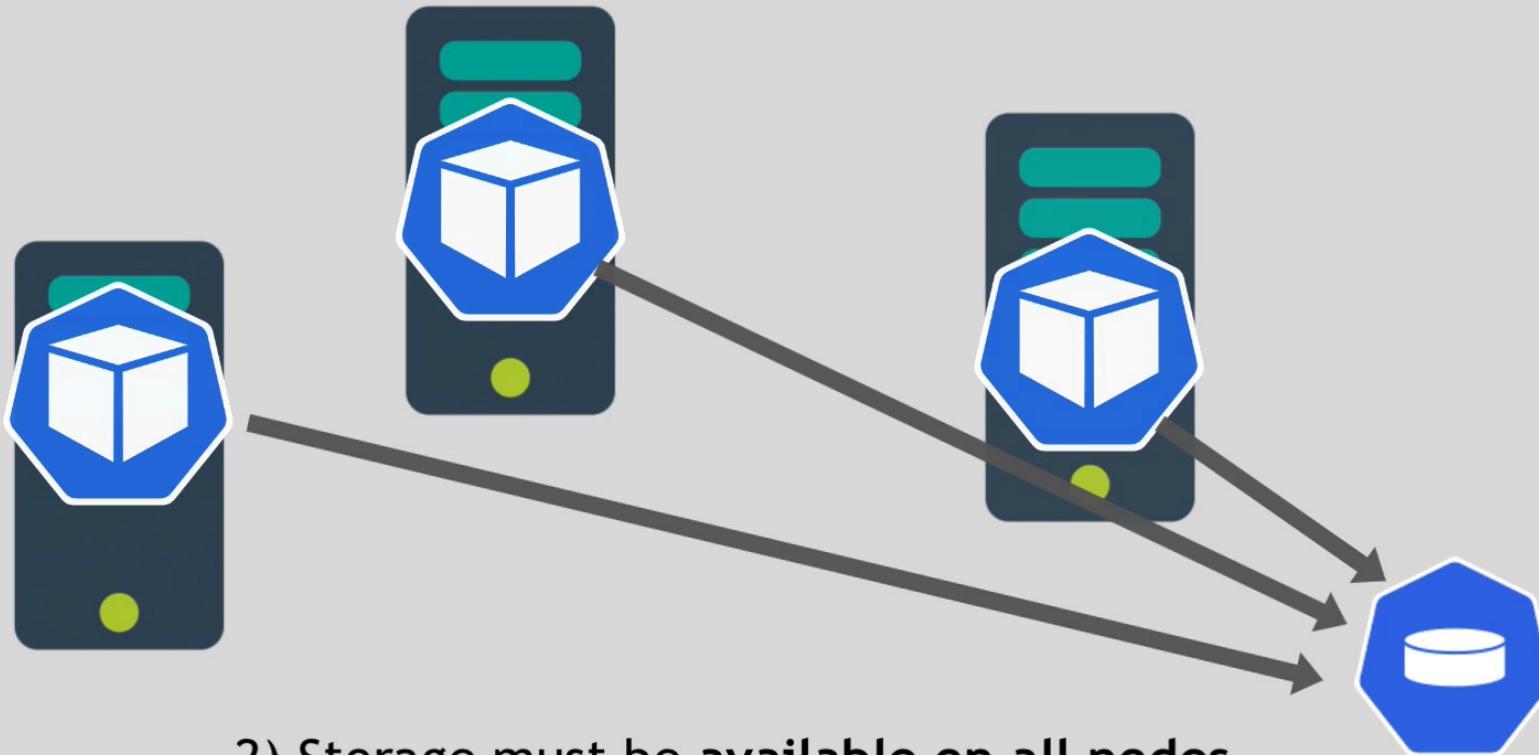
mysql



storage

Storage that **doesn't depend on the pod lifecycle.**

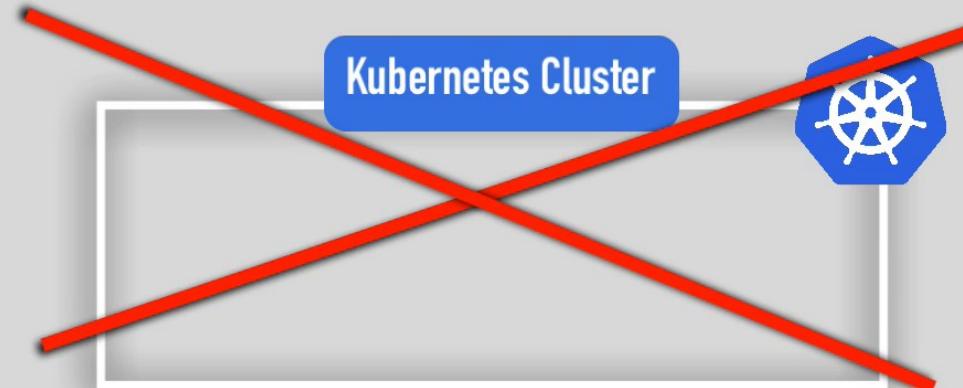
## Storage Requirements



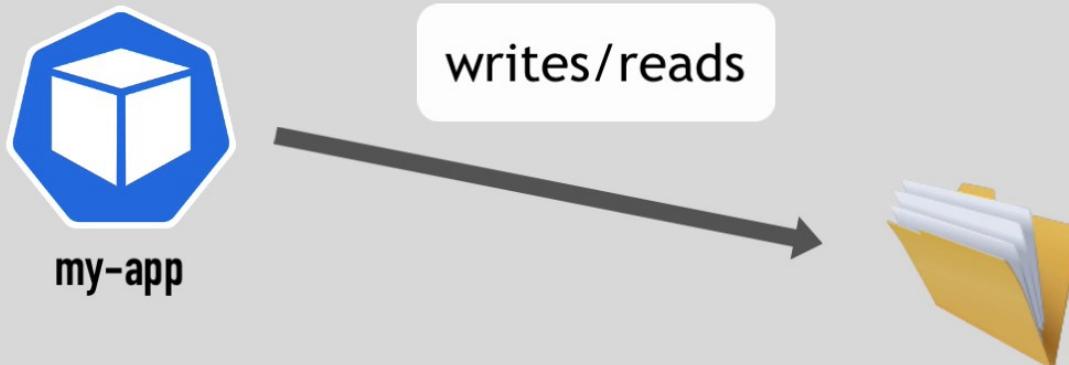
2) Storage must be available on all nodes.

## Storage Requirements

- 1) Storage that **doesn't depend on the pod lifecycle**.
- 2) Storage must be **available on all nodes**.
- 3) Storage needs to **survive even if cluster crashes**.



## Another Use Case for persistent storage



# Persistent Volume



Kubernetes Cluster

- a cluster resource

CPU

RAM



# Persistent Volume



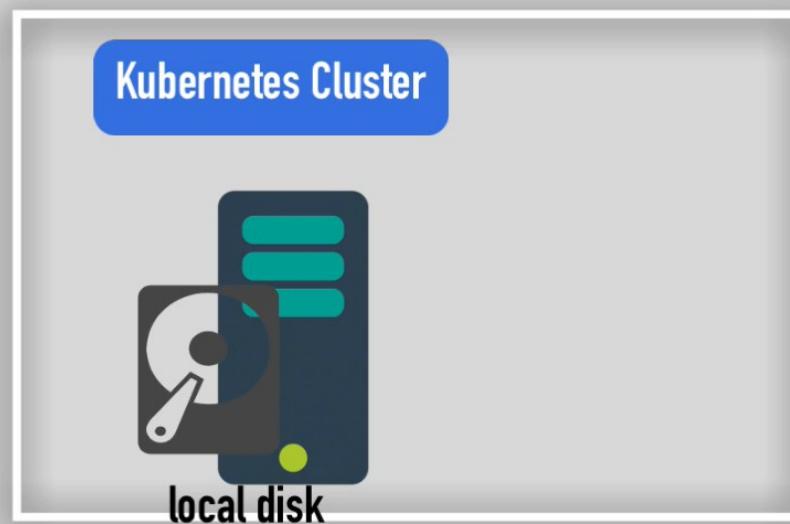
- a cluster resource
- created via YAML file
  - kind: PersistentVolume
  - spec: e.g. how much storage?

```
● ● ●  
apiVersion: v1  
kind: PersistentVolume  
metadata:  
  name: pv-name  
spec:  
  capacity:  
    storage: 5Gi  
  volumeMode: Filesystem  
  accessModes:  
    - ReadWriteOnce  
  persistentVolumeReclaimPolicy: Recy  
  storageClassName: slow  
  mountOptions:
```

# Persistent Volume



Needs actual physical storage, like

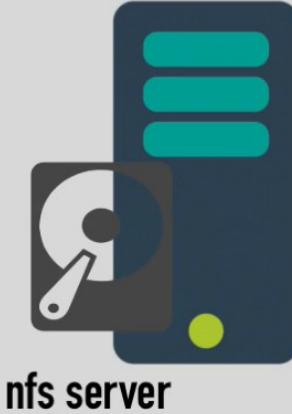
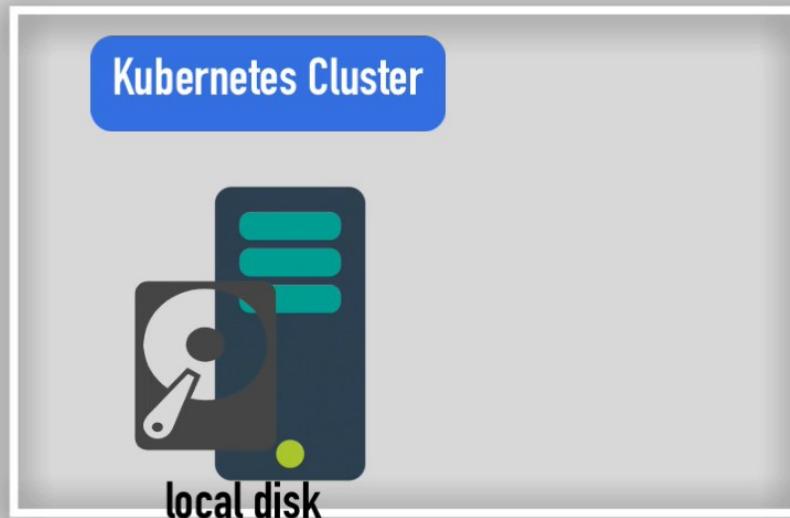


# Persistent Volume



Where does this storage come from  
and who makes it available to the cluster?

cloud-storage



## Persistent Volume



What type of storage do you need?

You need to **create and manage** them by yourself

external plugin  
to your cluster



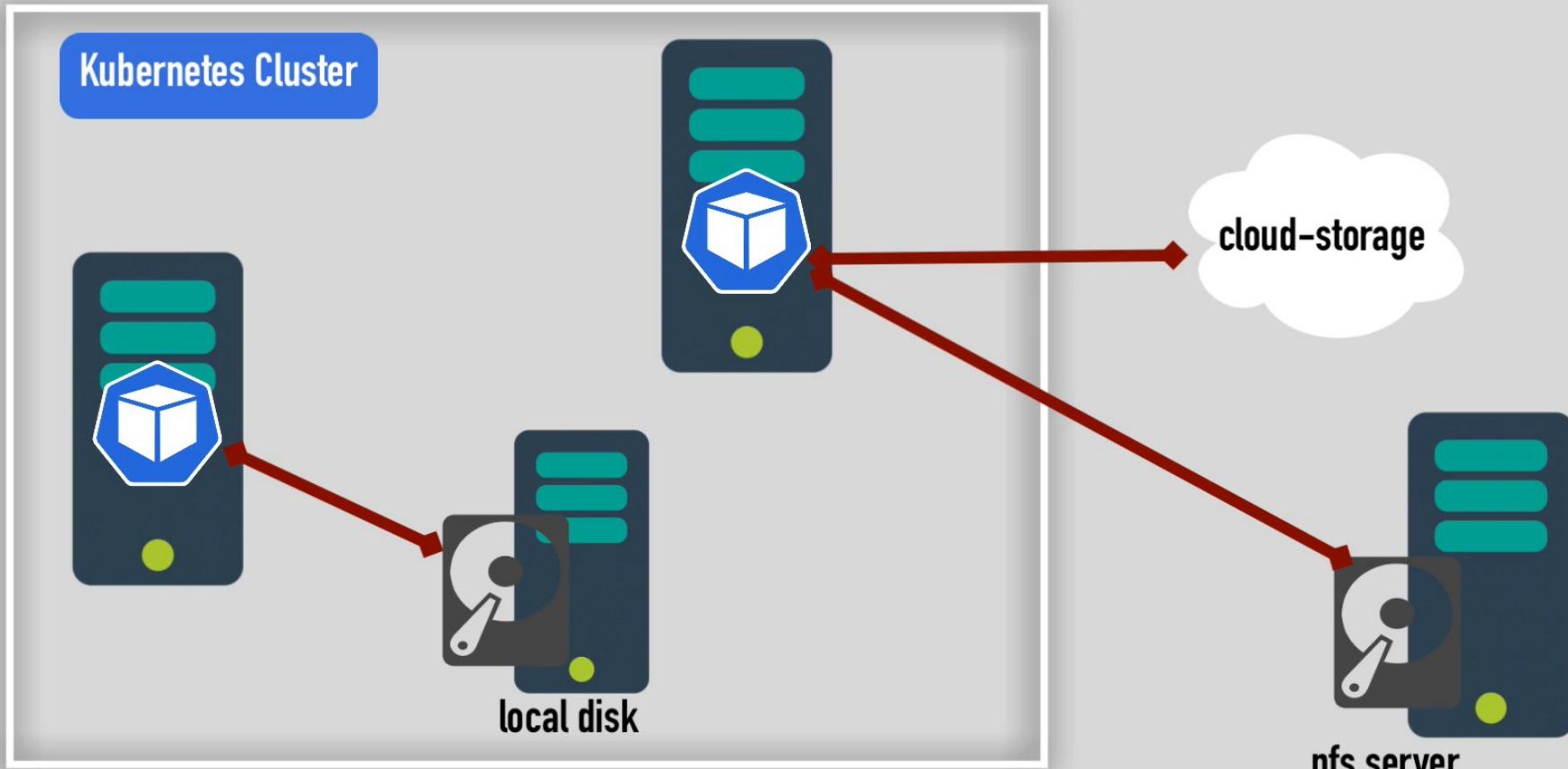
Kubernetes Cluster

cloud-storage



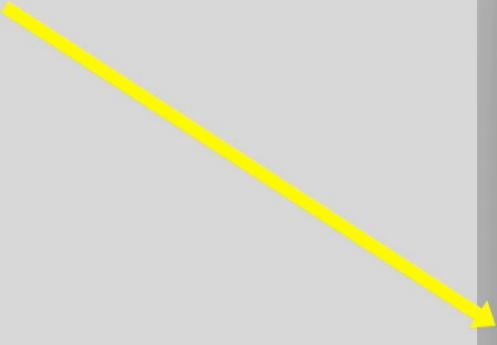
nfs server

# Persistent Volume



## Persistent Volume YAML Example

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



```
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

NFS Storage

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

How much:

Additional params,  
like access:

Nfs parameters:

```
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

Depending on storage type,  
spec attributes differ

Node Affinity:

Local storage

```
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
```

# Persistent Volume YAML Example

Depending on storage type,  
spec attributes differ

Complete list of storage backends  
supported by K8s:

[Documentation](#) [Blog](#) [Training](#) [Partners](#) [Community](#) [Ca](#)

the Pod must independently specify where to mo

## Types of Volumes

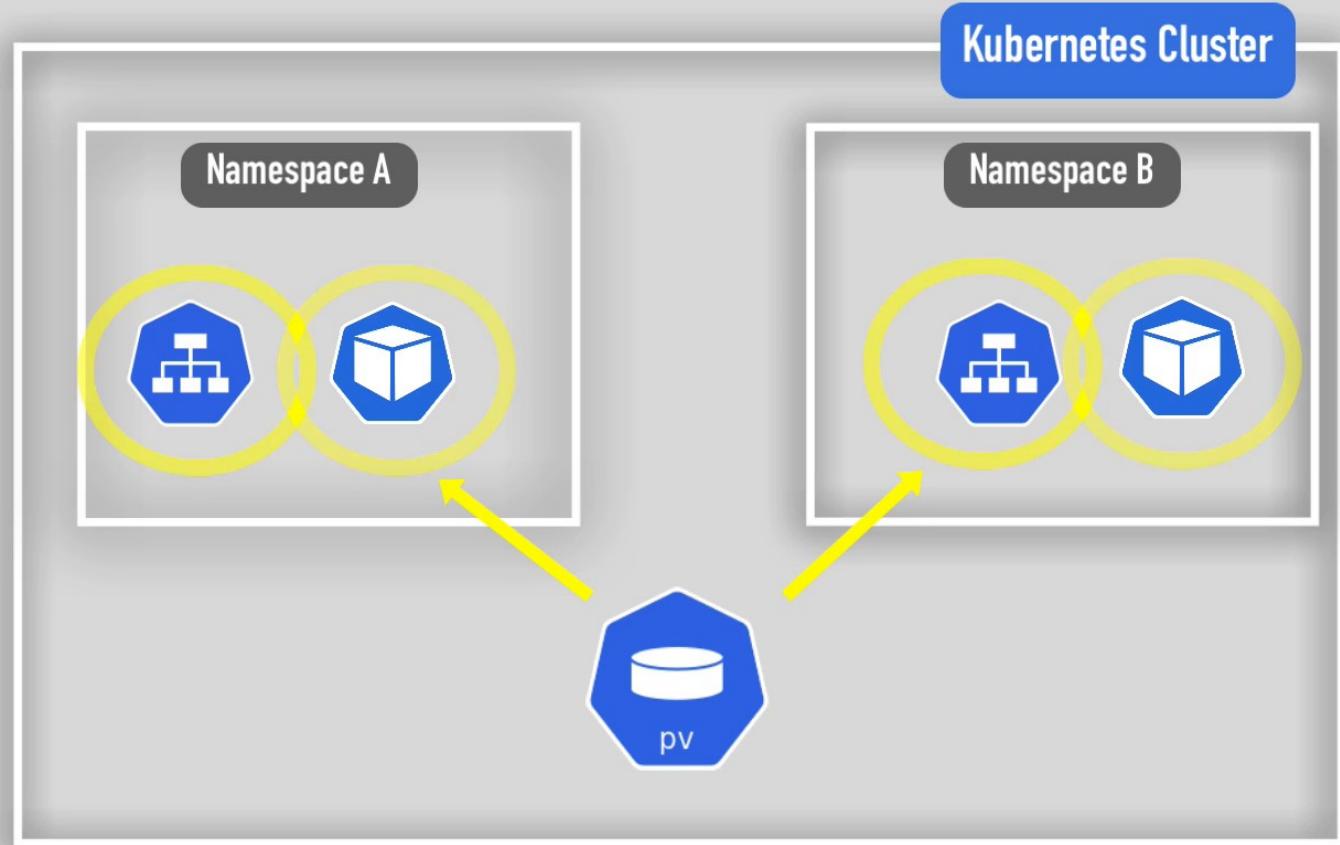
Kubernetes supports several types of Volumes:

- [awsElasticBlockStore](#)
- [azureDisk](#)
- [azureFile](#)
- [cephfs](#)
- [cinder](#)
- [configMap](#)
- [csi](#)
- [downwardAPI](#)
- [emptyDir](#)
- [fc \(fibre channel\)](#)
- [flexVolume](#)
- [flocker](#)

## Persistent Volumes are NOT namespaced

PV outside of the namespaces

Accessible to the whole cluster



## Local vs. Remote Volume Types

Each volume type has it's 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!

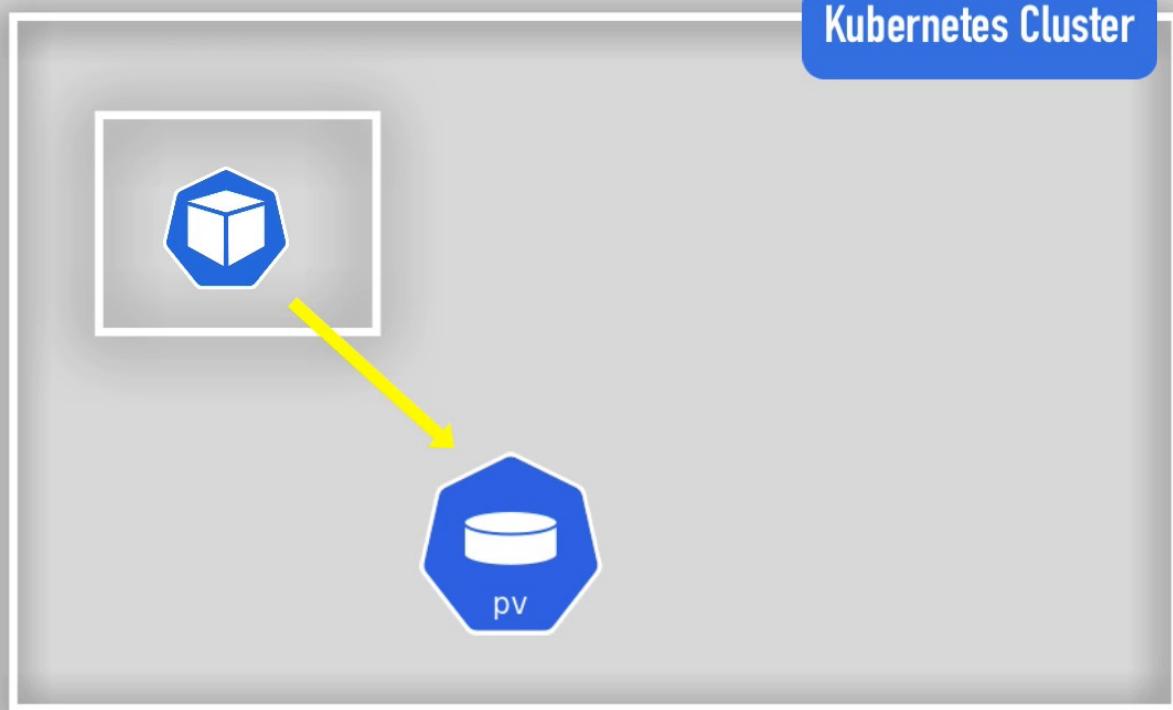
## K8s Administrator and K8s User

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



## K8s Administrator and K8s User



K8s Admin sets up and maintains the cluster

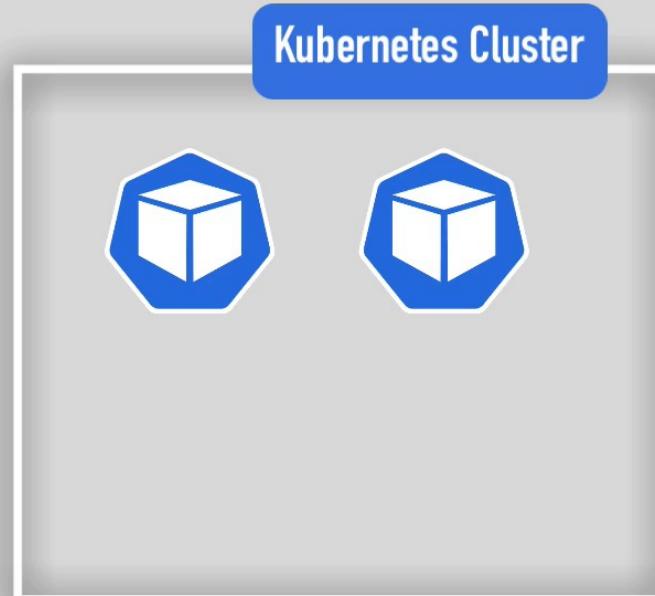
Kubernetes Cluster

enough resources?

## K8s Administrator and K8s User



K8s Admin sets up and maintains the cluster

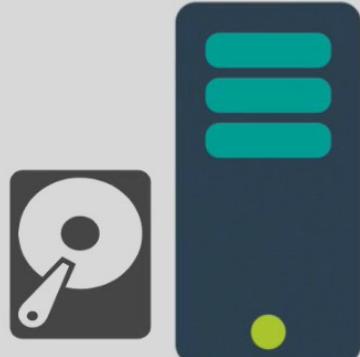


K8s User deploys applications in cluster

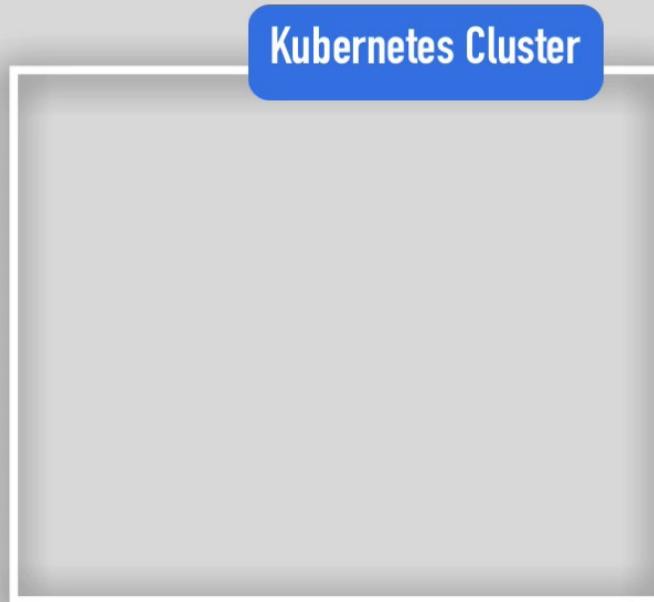
## K8s Administrator and K8s User



Storage resource is provisioned by Admin.



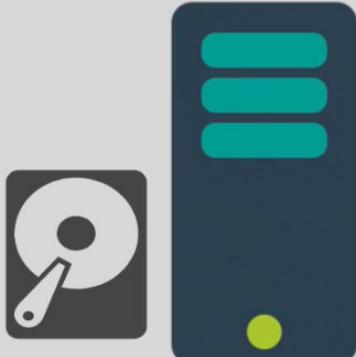
nfs-storage



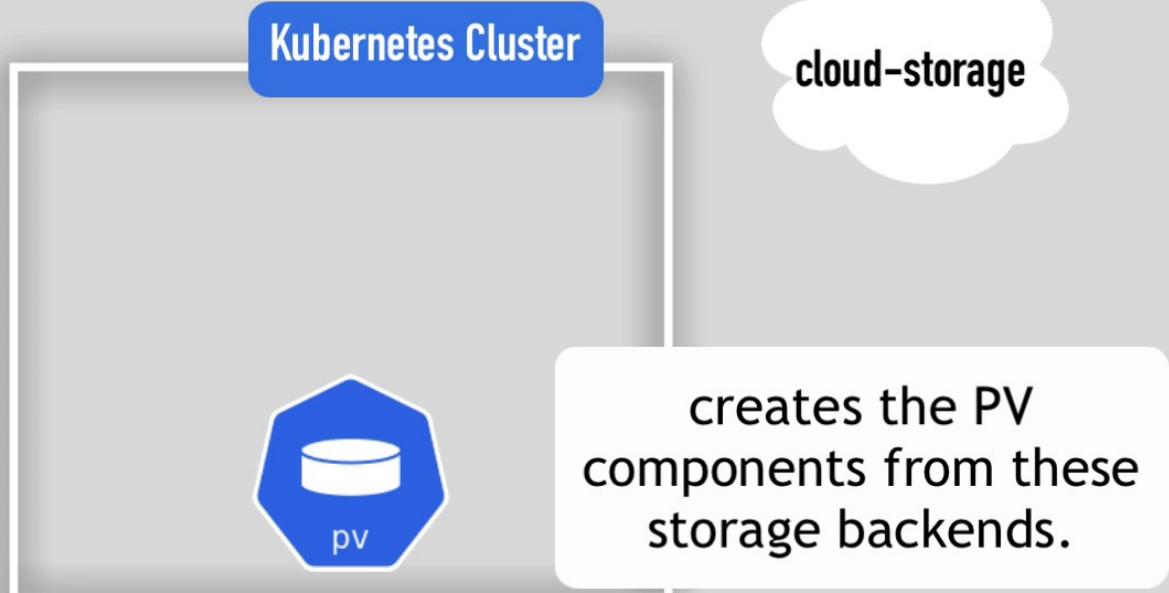
## K8s Administrator and K8s User



Storage resource is provisioned by Admin.



**nfs-storage**



creates the PV  
components from these  
storage backends.

## K8s Administrator and K8s User



*We need a Google  
Cloud storage*

## Persistent Volume Claim component

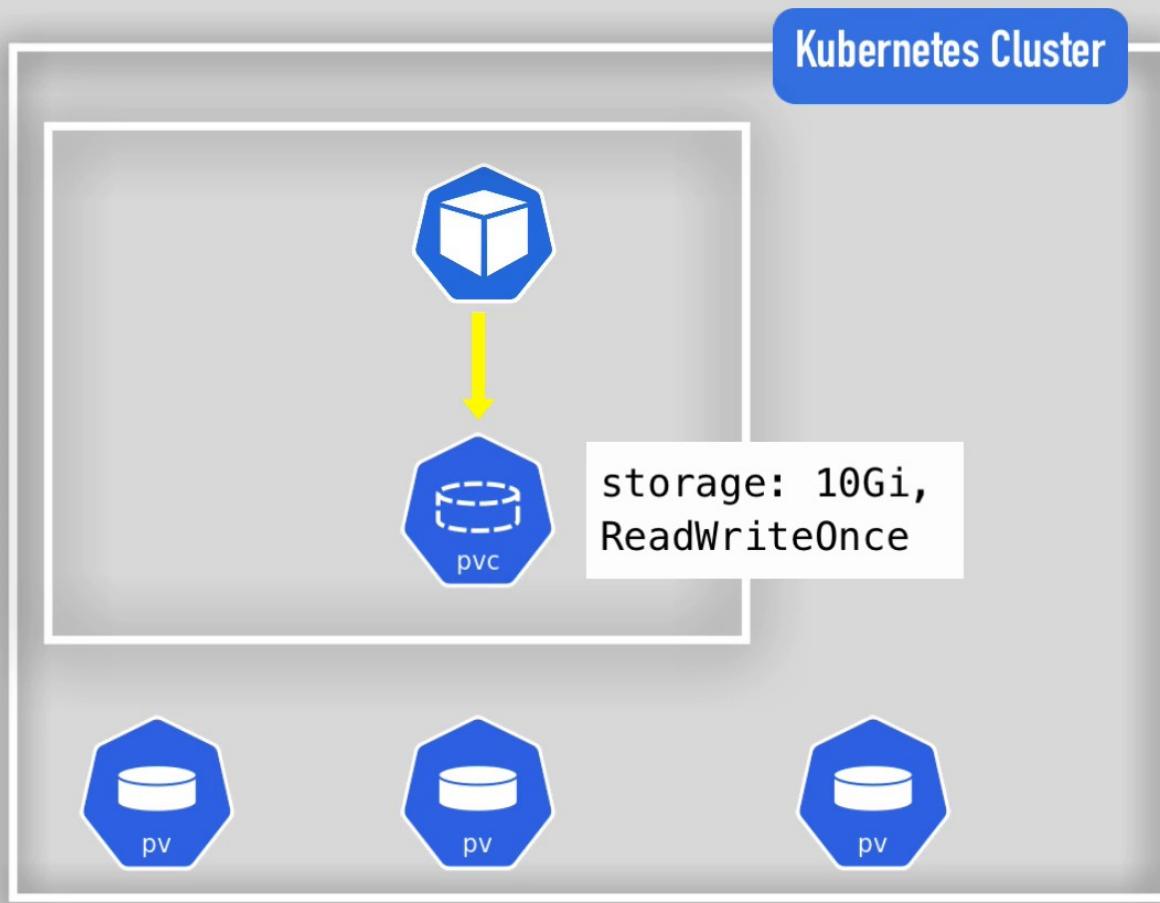
Kubernetes Cluster



Application has to **claim** the Persistent Volume



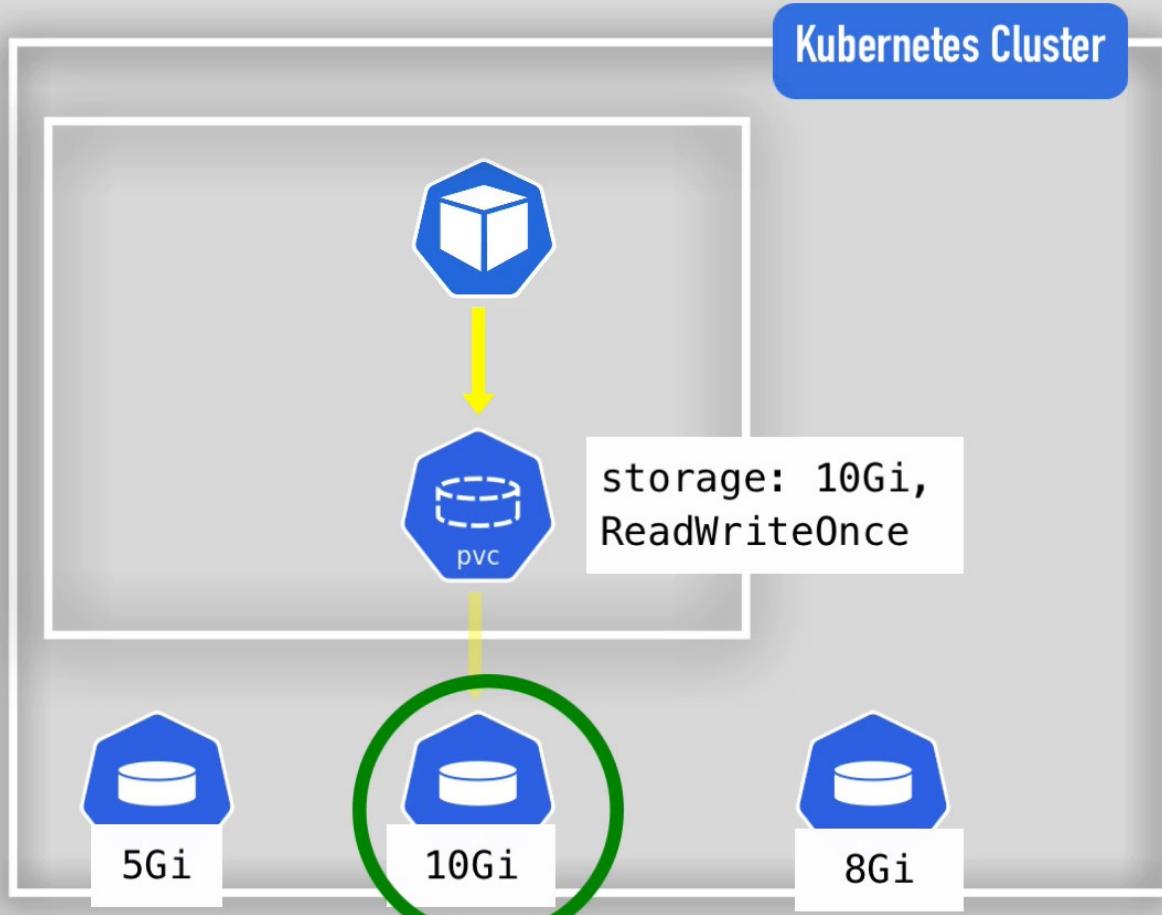
## Persistent Volume Claim component



```
● ○ ●

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-name
spec:
  storageClassName: manual
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
resources:
  requests:
    storage: 10Gi
```

## Persistent Volume Claim component



```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-name
spec:
  storageClassName: manual
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
resources:
  requests:
    storage: 10Gi
```

# PersistentVolumeClaim component

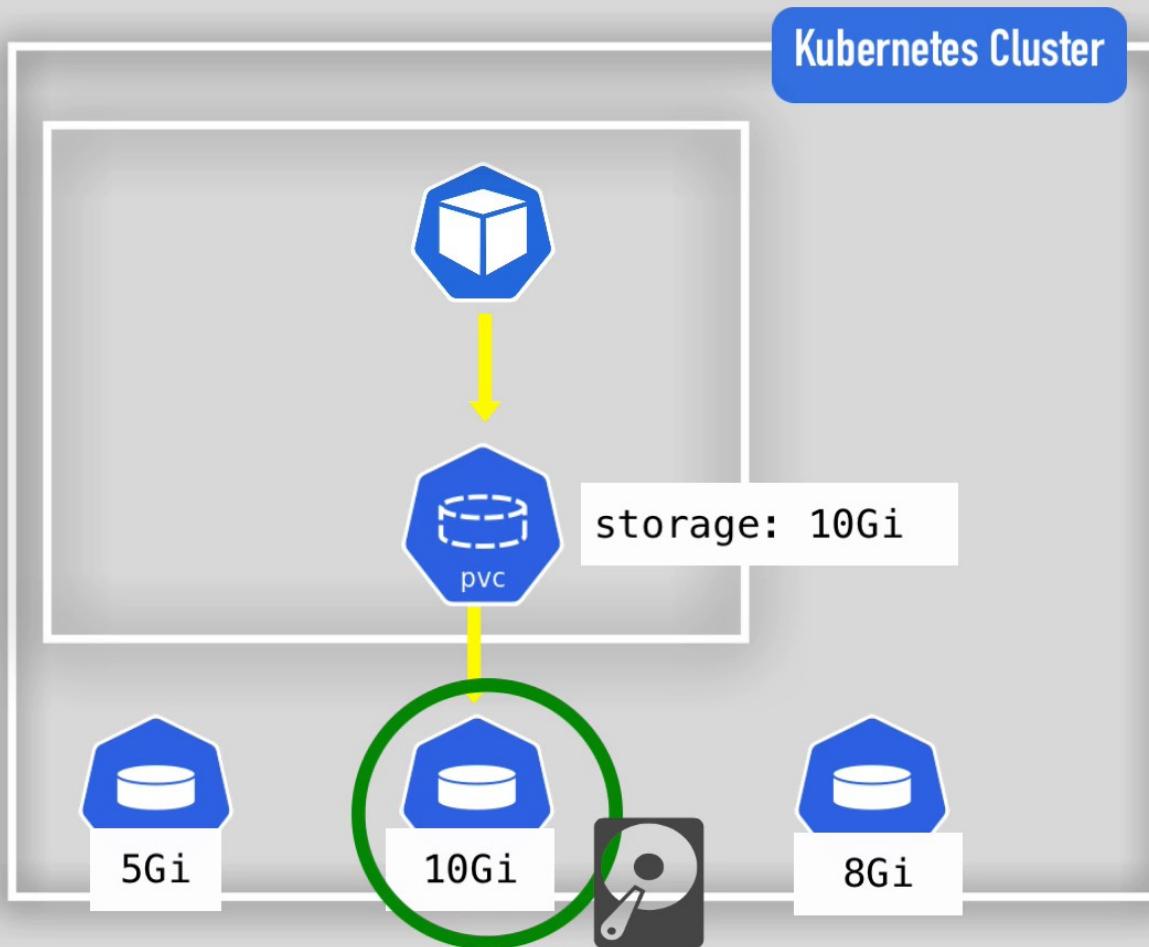
Use that PVC in Pods configuration

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
        - mountPath: "/var/www/html"
          name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: pvc-name
```



```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-name
spec:
  storageClassName: manual
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

## Levels of Volume abstractions

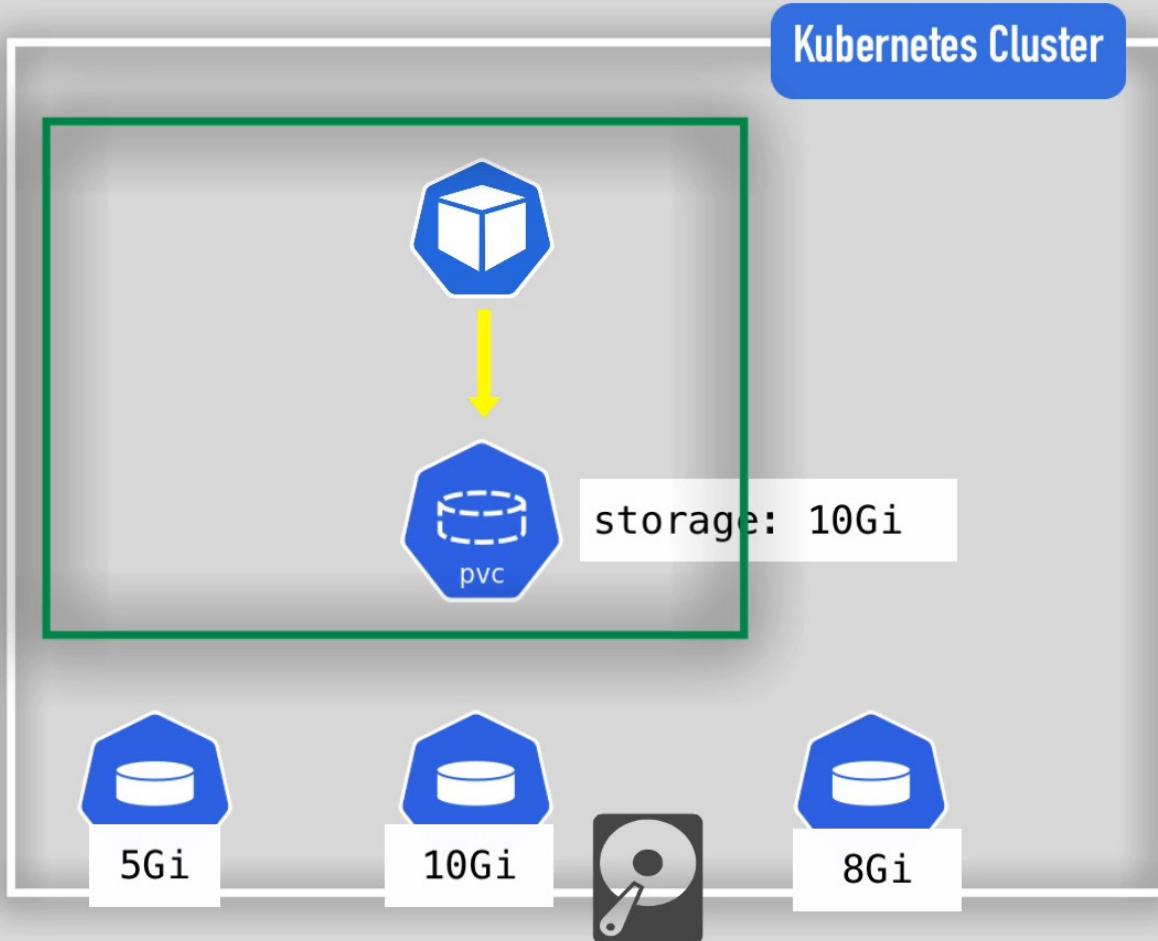


Pod requests the volume through the PV claim

Claim tries to find a volume in cluster

Volume has the actual storage backend

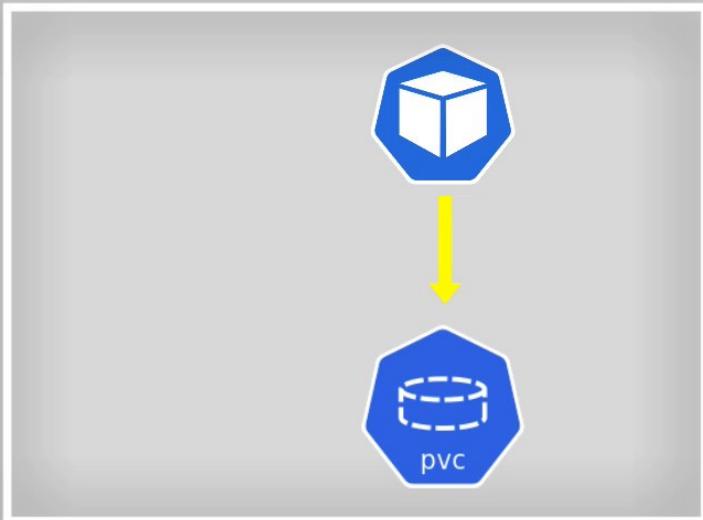
## Levels of Volume abstractions



Claims must be in the **same namespace!**

# Levels of Volume abstractions

## Kubernetes Cluster

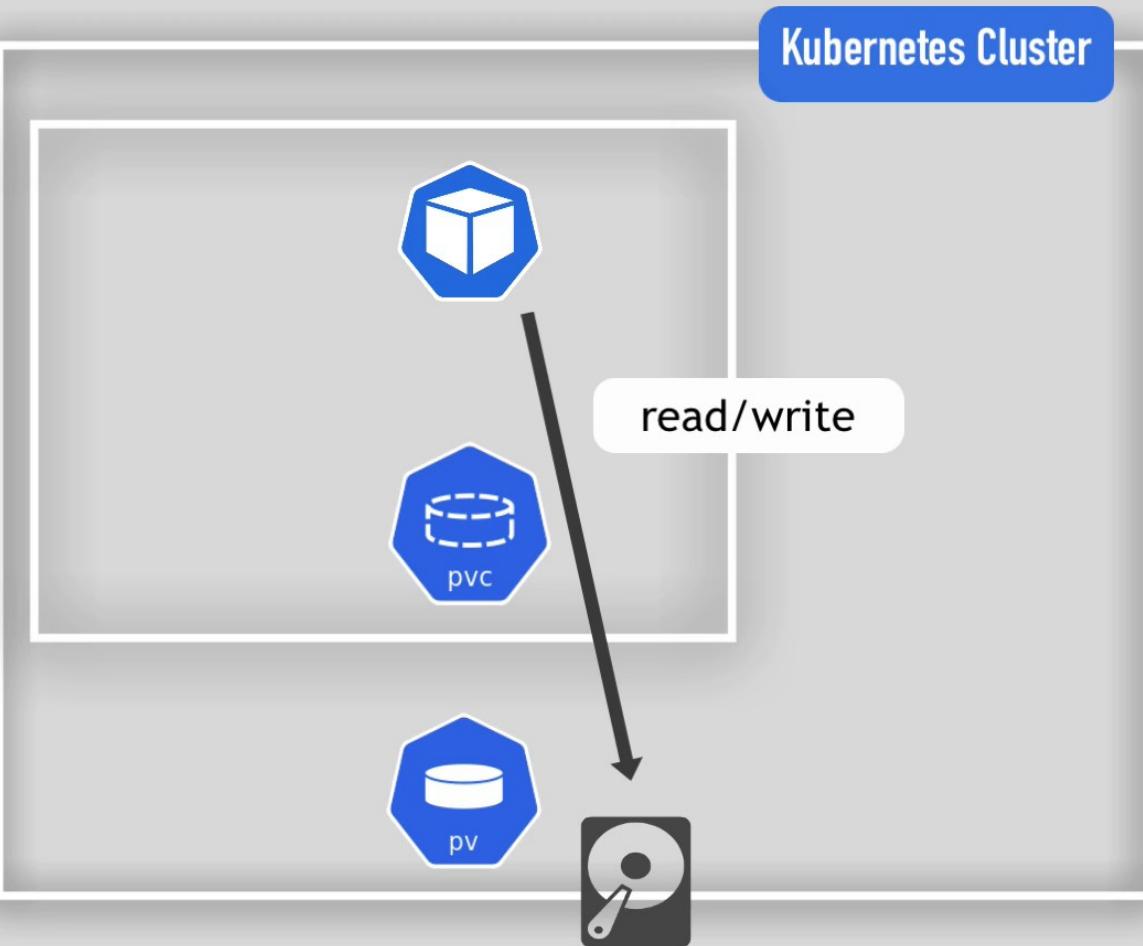


```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
        - mountPath: "/var/www/html"
          name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: pvc-name
```

Volume is mounted into Container

Volume is mounted into the Pod

# Levels of Volume abstractions



```
● ○ ●  
apiVersion: v1  
kind: Pod  
metadata:  
  name: mypod  
spec:  
  containers:  
    - name: myfrontend  
      image: nginx  
      volumeMounts:  
        - mountPath: "/var/www/html"  
          name: mypd  
  volumes:  
    - name: mypd  
  persistentVolumeClaim:  
    claimName: pvc-name
```

# Why so many abstractions? 🤔



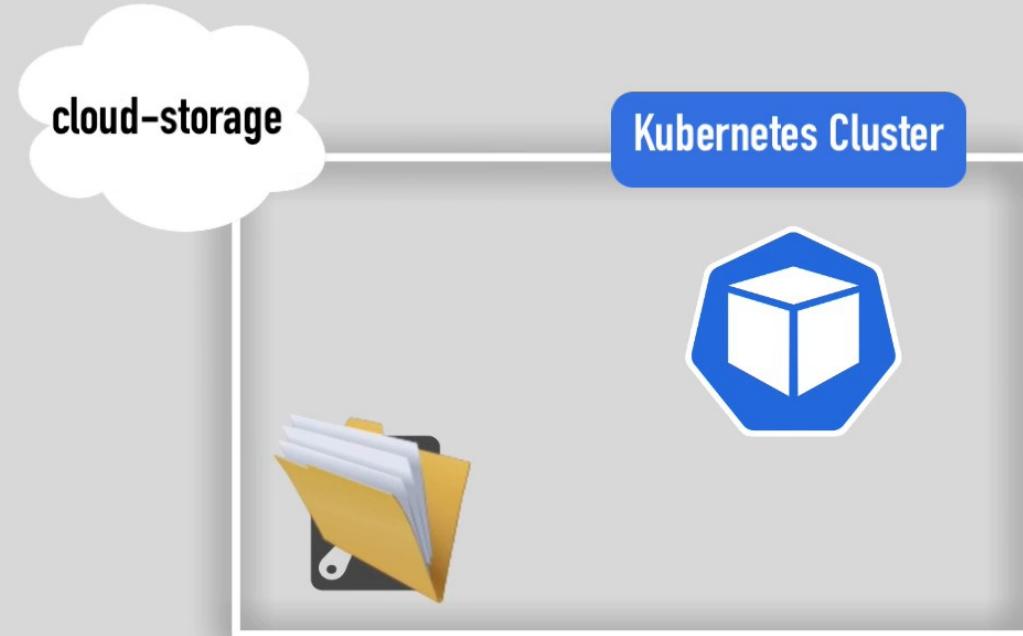
Admin provisions storage resource



User creates claim to PV



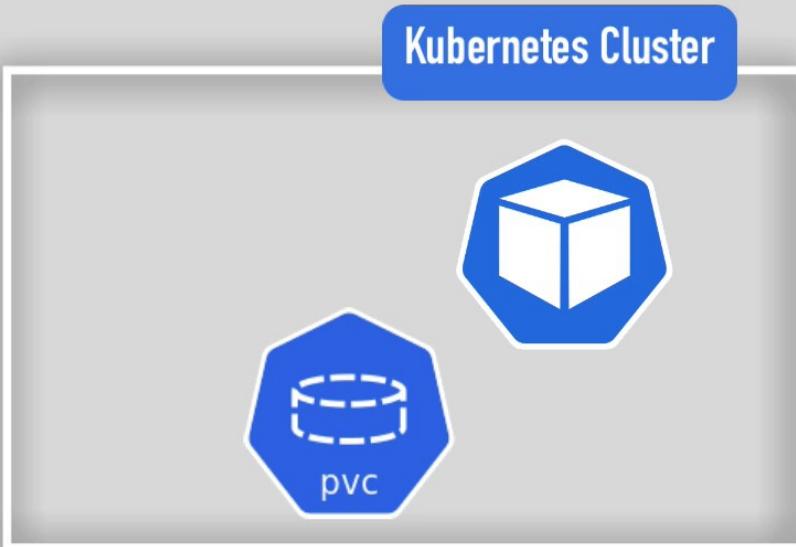
## Why so many abstractions? 🤯



Data should be safely stored



## Why so many abstractions? 🤔



Data should be safely stored

Don't want to set up the actual  
storages



Why so many abstractions? 🤔

Easier for developers

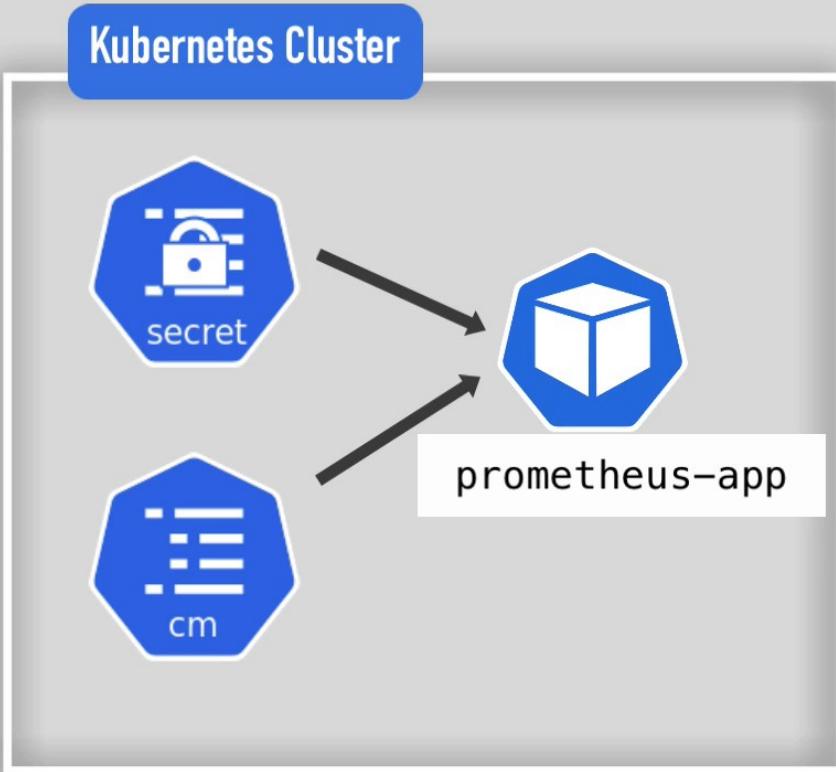


## ConfigMap and Secret



- local volumes
- not created via PV and PVC
- managed by Kubernetes

# ConfigMap and Secret



Configuration file for your pod.

Certificate file for your pod.



# ConfigMap and Secret

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: busybox-container
      image: busybox
      volumeMounts:
        - name: config-dir
          mountPath: /etc/config
  volumes:
    - name: config-dir
      configMap:
        name: bb-configmap
```

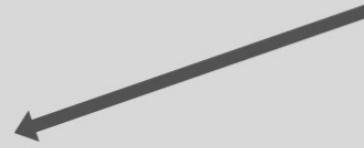
- 1) Create ConfigMap and/or Secret component
- 2) Mount that into your pod/container

## What we've covered so far ⚡

Volume is directory with some data

These volumes are accessible in containers in a pod

How made available, backed by which storage medium  
- defined by specific volume types



/var/lib

## What we've covered so far



```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
        - mountPath: "/var/www/html"
          name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: pvc-name
```

Where to mount those  
in the containers?

Pod specifies what  
Volumes to provide

## What we've covered so far



```
● ● ●  
apiVersion: v1  
kind: Pod  
metadata:  
  name: mypod  
spec:  
  containers:  
    - name: myfrontend  
      image: nginx  
      volumeMounts:  
        - mountPath: "/var/www/html"  
          name: mypd  
  volumes:  
    - name: mypd  
      persistentVolumeClaim:  
        claimName: pvc-name
```

➤ Apps can access the mounted data here:  
**"/var/www/html"**

## What we've covered so far !



```
● ● ●

apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
        - mountPath: "/var/www/html"
          name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: pvc-name
```



elastic-app



## Different volume type

```
kind: Deployment
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
```



elastic-app



secret



cm

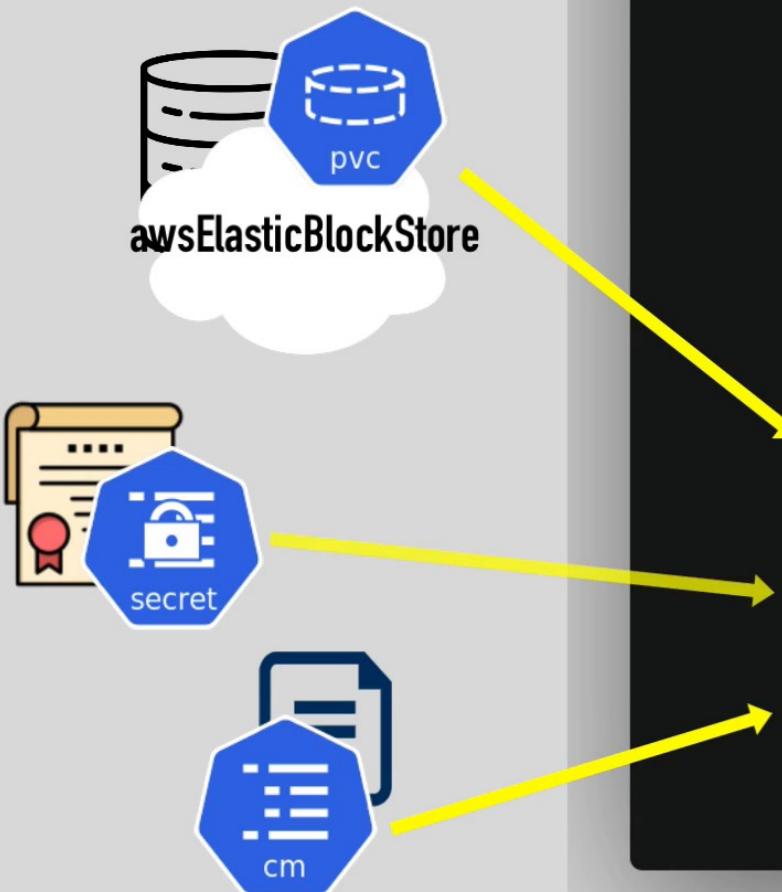


awsElasticBlockStore

```
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:  
        secretName: es-secret  
    - name: es-config-dir  
      configMap:  
        name: es-config-map
```



elastic-app



```
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:  
        secretName: es-secret  
    - name: es-config-dir  
      configMap:  
        name: es-config-map
```



elastic-app



## Different volume type

```
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:  
            secretName: es-secret  
        - name: es-config-dir  
          configMap:  
            name: es-config-map
```

# Storage Class



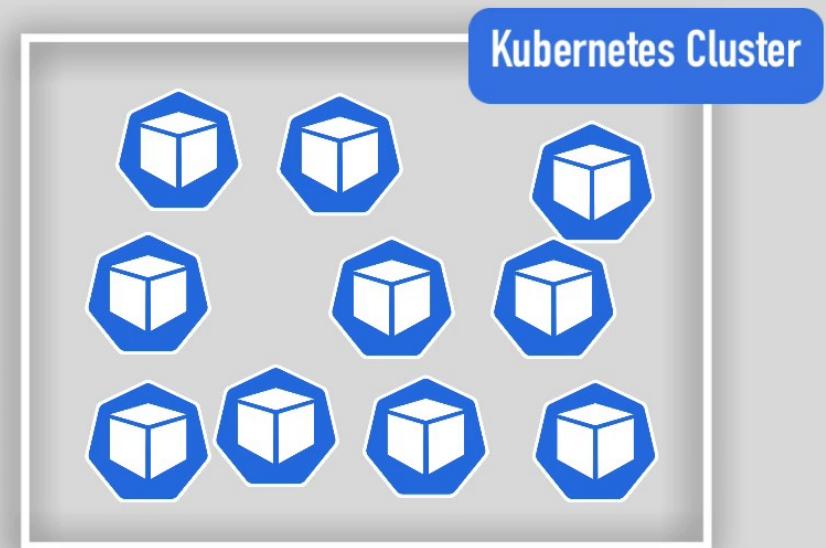
1) Admins configure storage



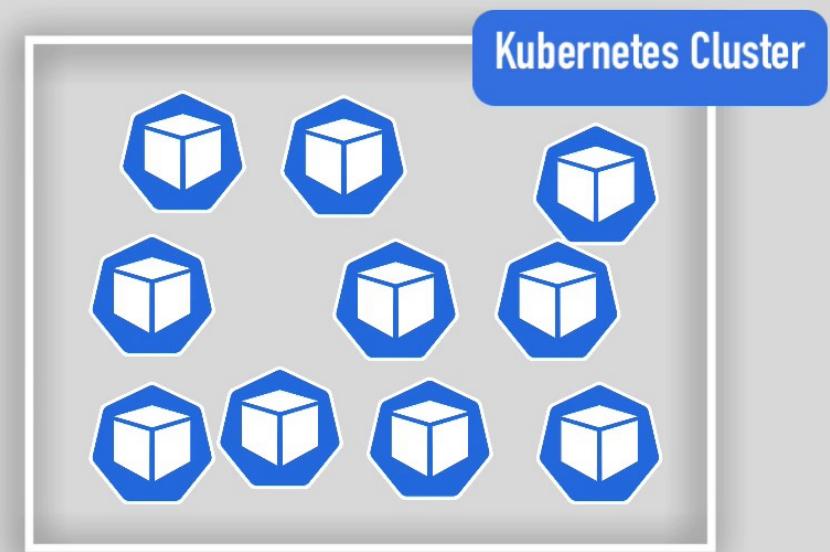
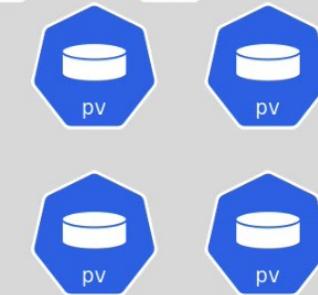
2) create Persistent Volumes



3) K8s Users claim PV using PVC



# Storage Class



3rd K8s Component, which makes  
process more efficient💡



## Storage Class



SC provisions Persistent Volumes **dynamically..**

..when PersistentVolumeClaim claims it



# Storage Class



kind: StorageClass



```
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



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

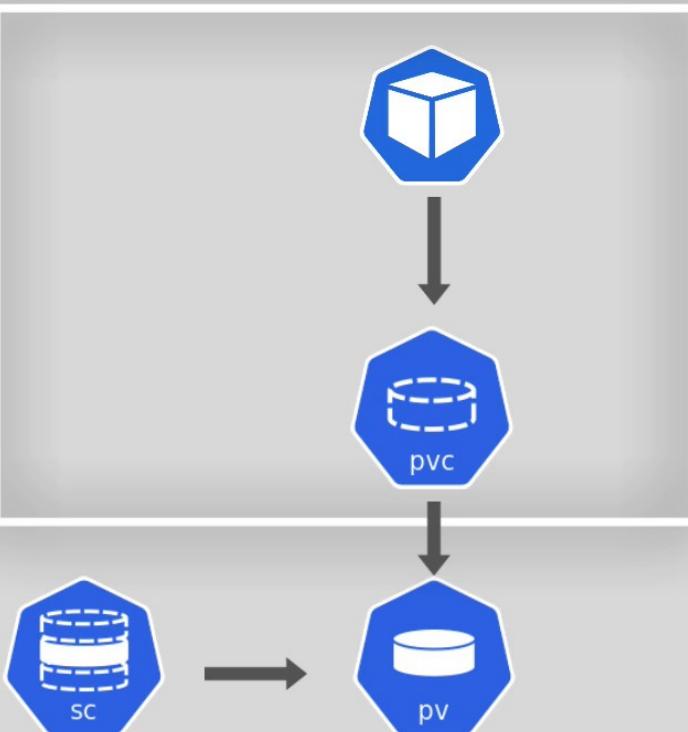
Storage Class Config

```
● ● ●  
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

Persistent Volume



Persistent Volume Claim



Storage Class













