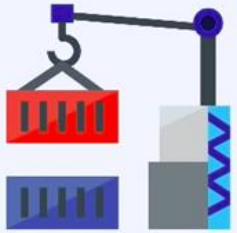
kubernetes

Official definition of Kubernetes

- Open source container orchestration tool
- Developed by Google
- Helps manage containerized applications in different deployment environments



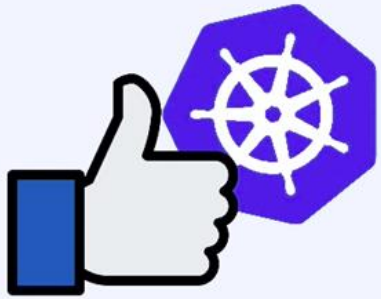
What problems does Kubernetes solve?

What are the tasks of an orchestration tool?

Need for container orchestration tool

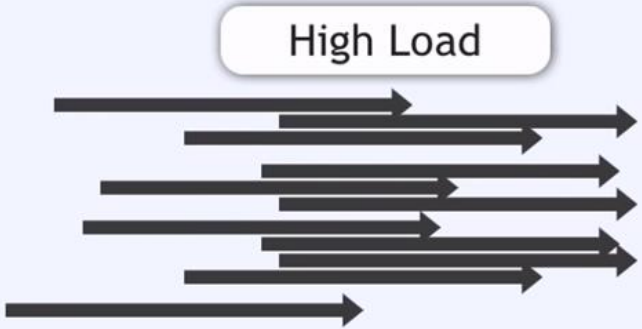
- Trend from Monolith to Microservices
- Increased usage of containers
- Demand for a proper way of managing those hundreds of containers

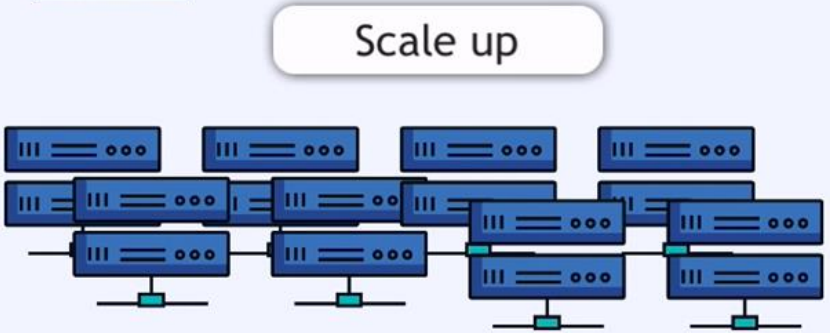
What features do orchestration tools offer?



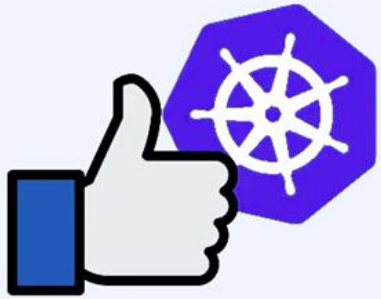
- High **Availability** or no downtime

- **Scalability** or high performance





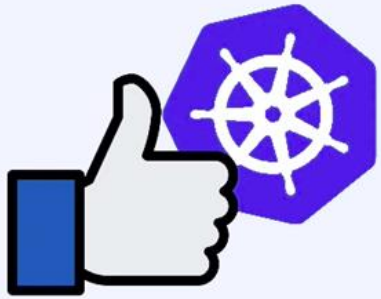
What features do orchestration tools offer?



- High **Availability** or no downtime

- **Scalability** or high performance

What features do orchestration tools offer?

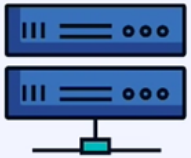


- High **Availability** or no downtime

- **Scalability** or high performance

- **Disaster recovery** - backup and restore

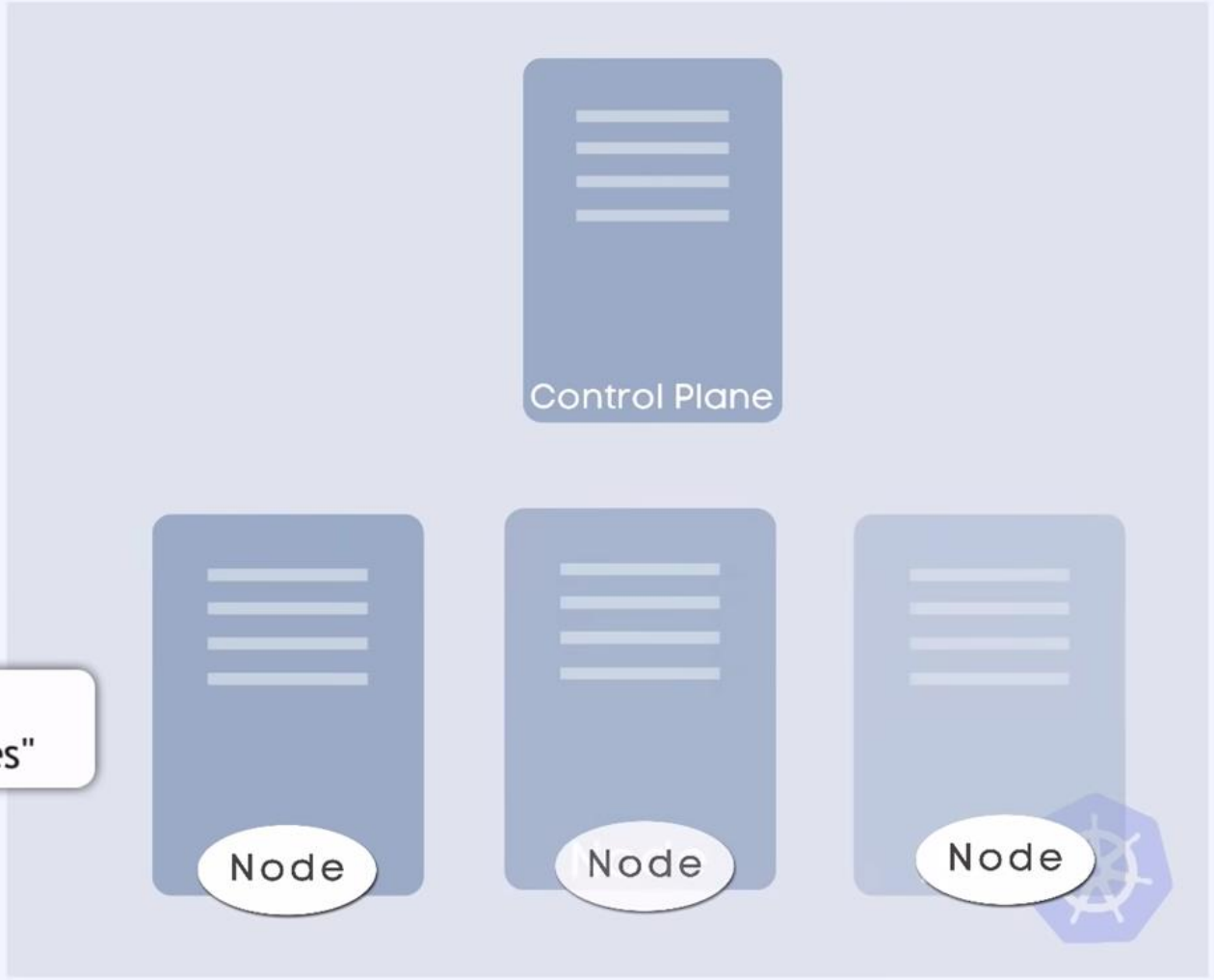
Kubernetes Architecture



Node = virtual or physical machine

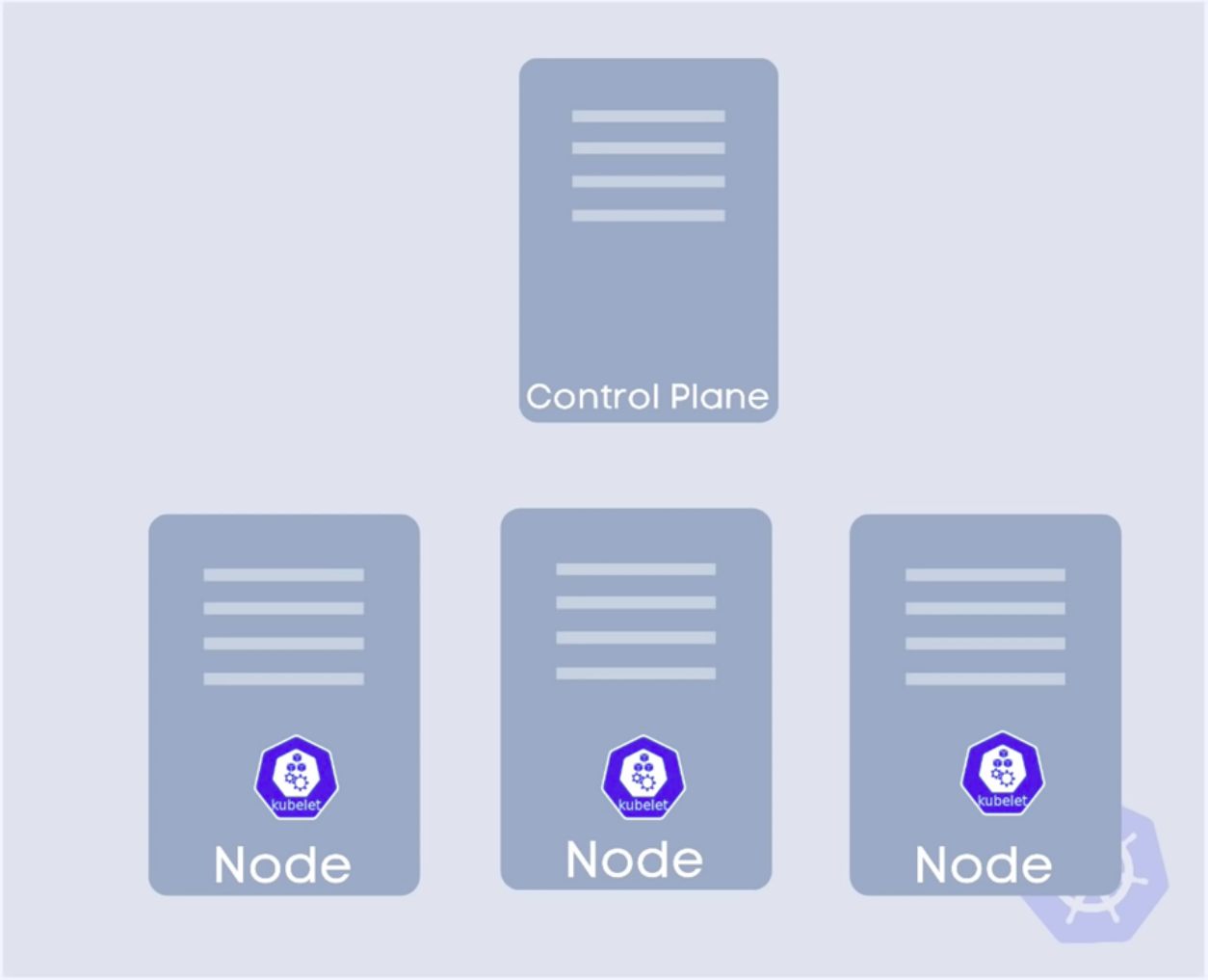
Kubernetes Architecture

Worker Nodes =
mostly just referred to as "Nodes"

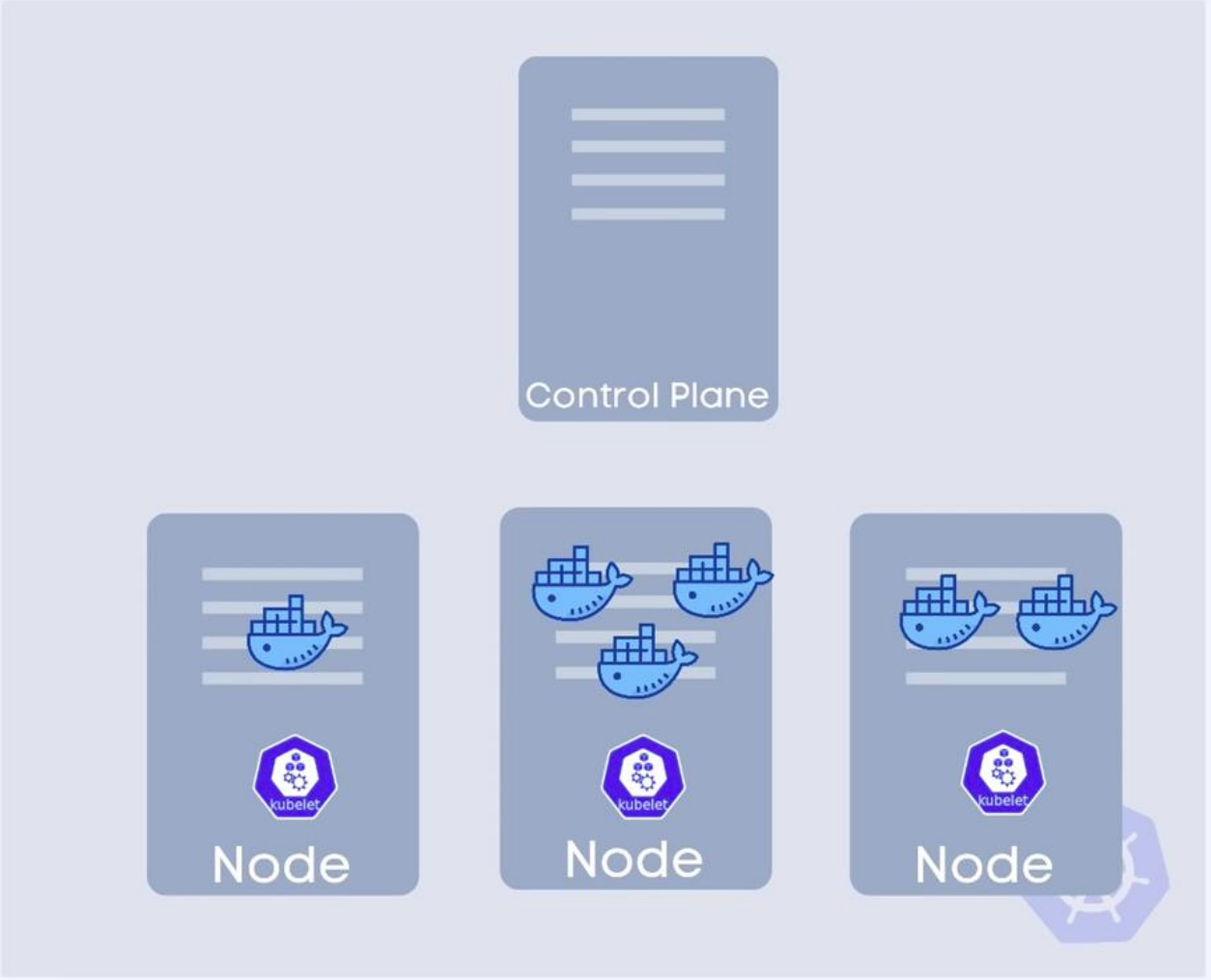


Kubernetes
Architecture

primary "node agent"

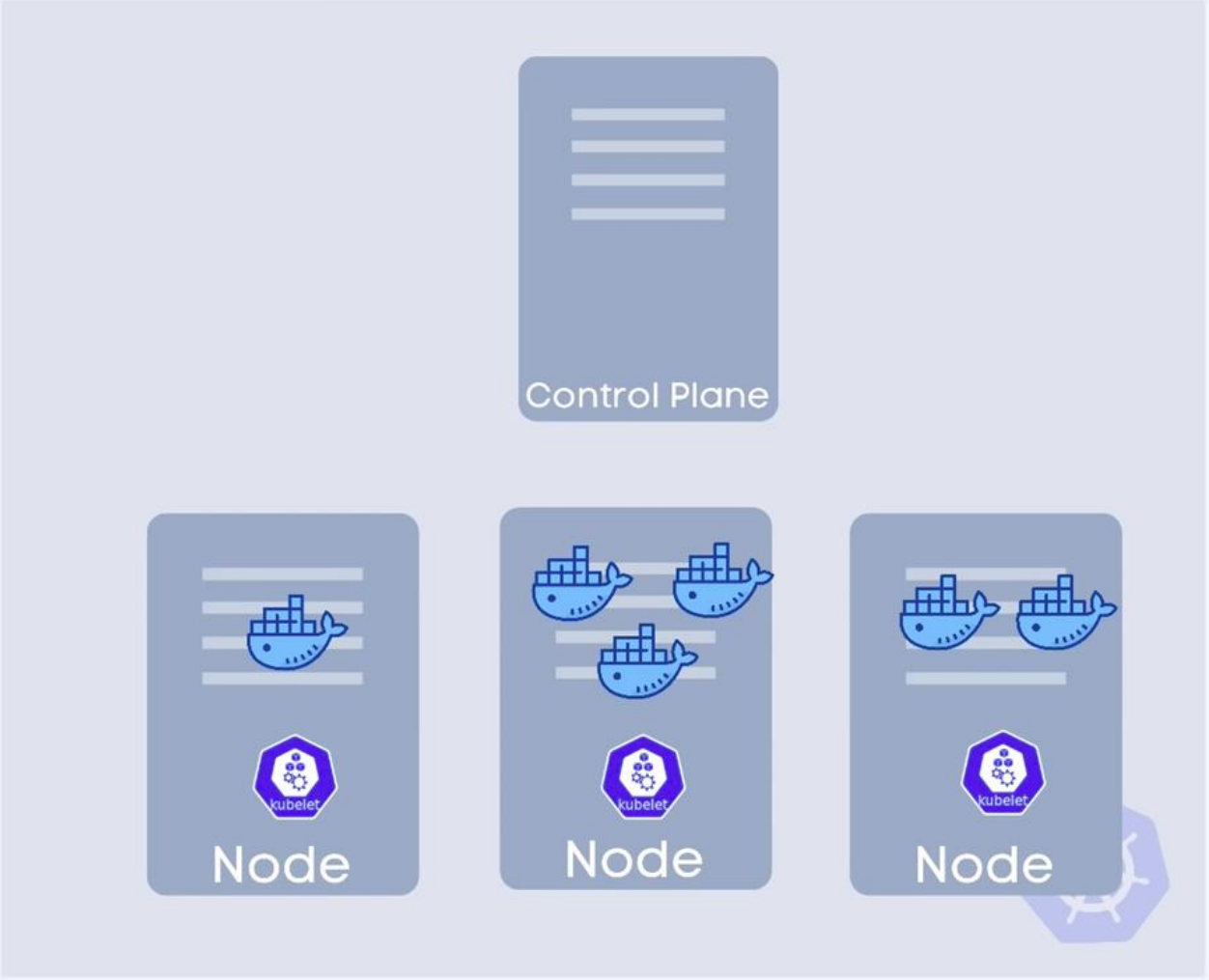


Kubernetes
Architecture

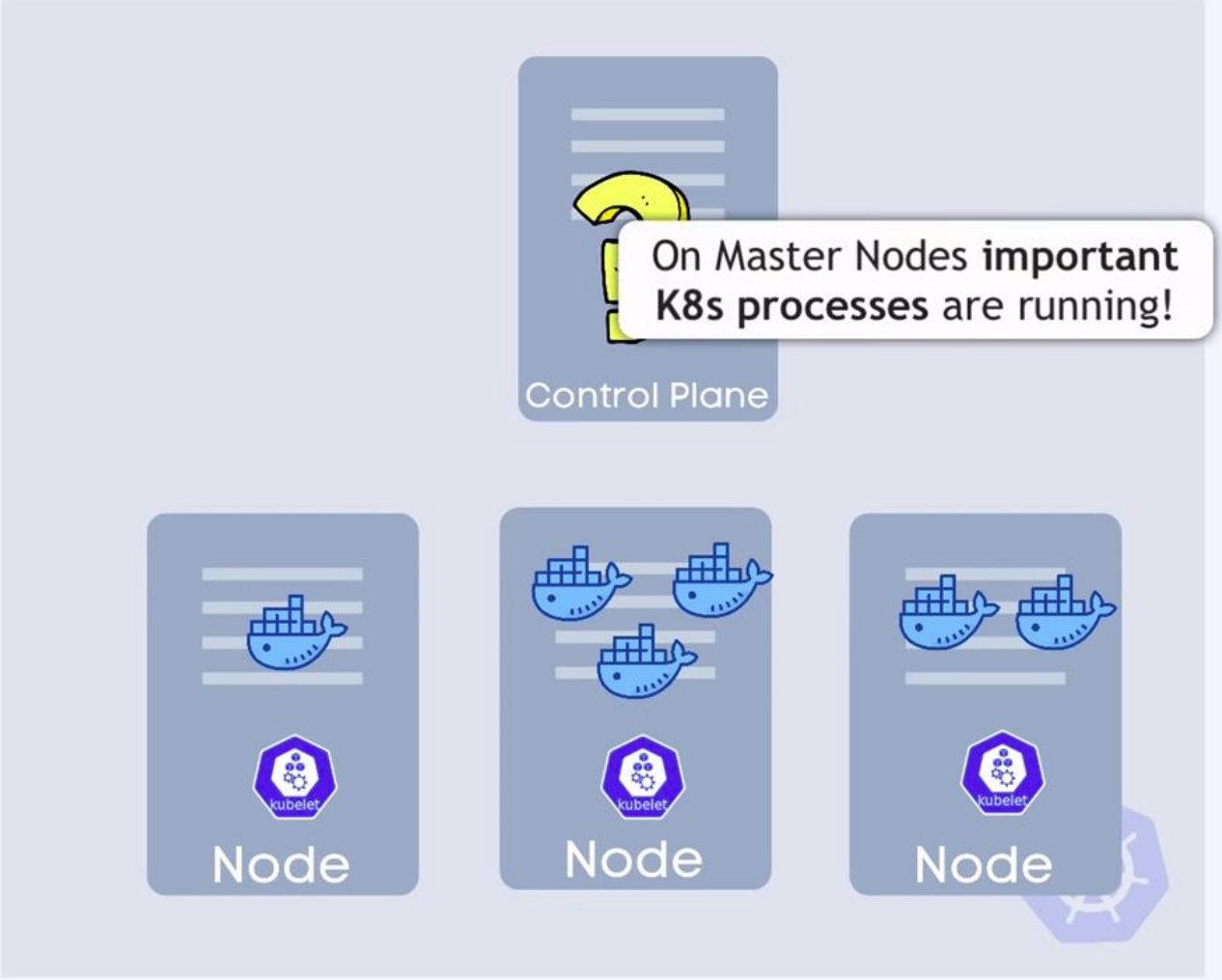


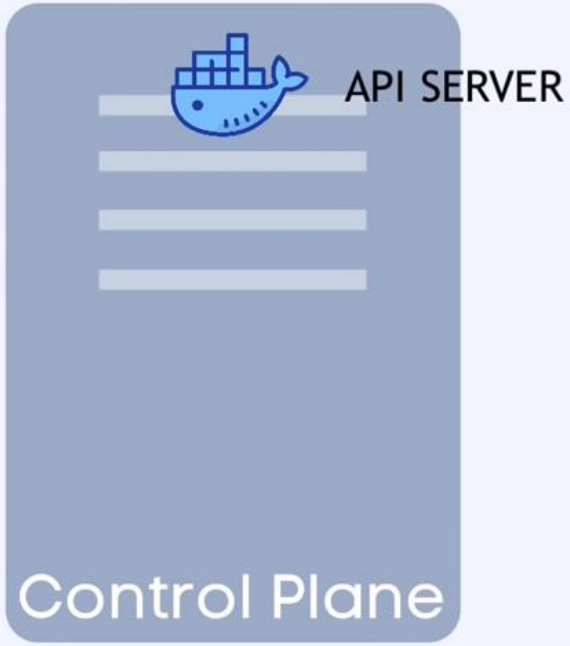
Kubernetes Architecture

On Worker Nodes your applications are running

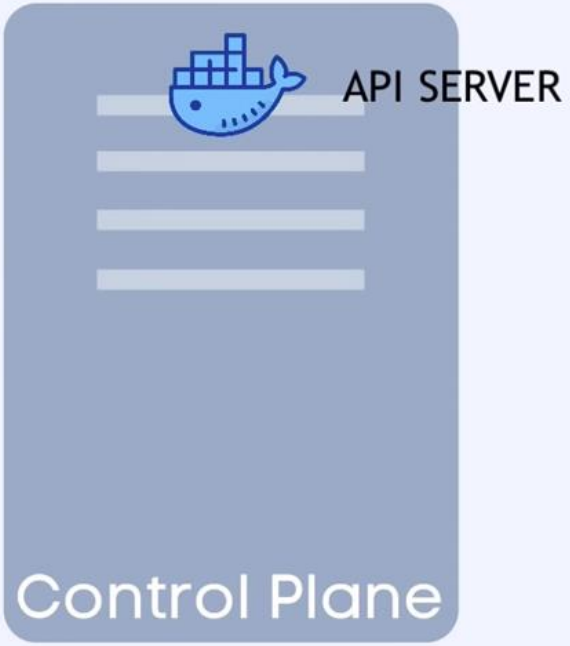


Kubernetes Architecture

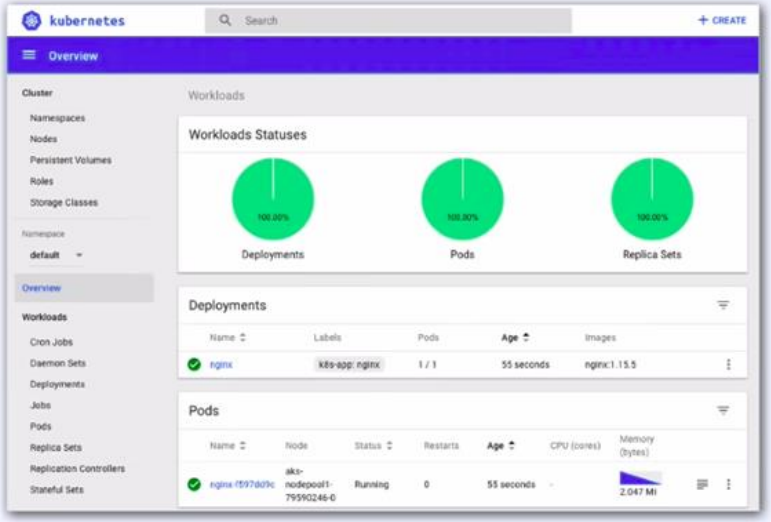


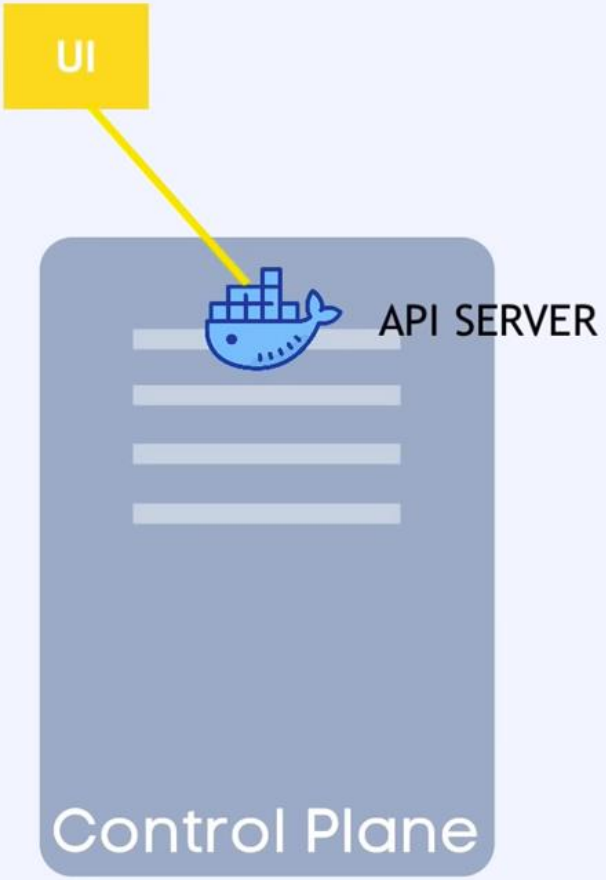

API SERVER
Control Plane

API Server =
Entrypoint to K8s cluster

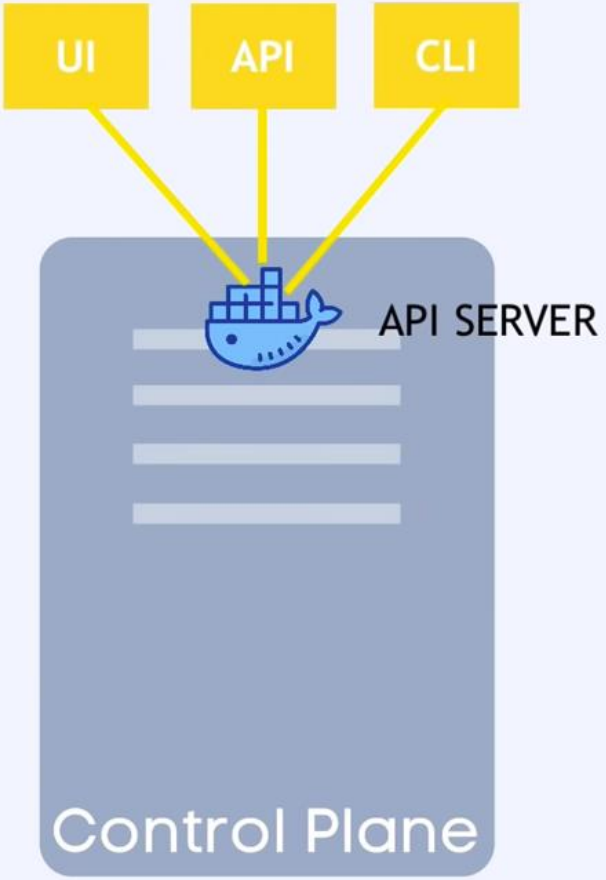


API Server =
Entrypoint to K8s cluster



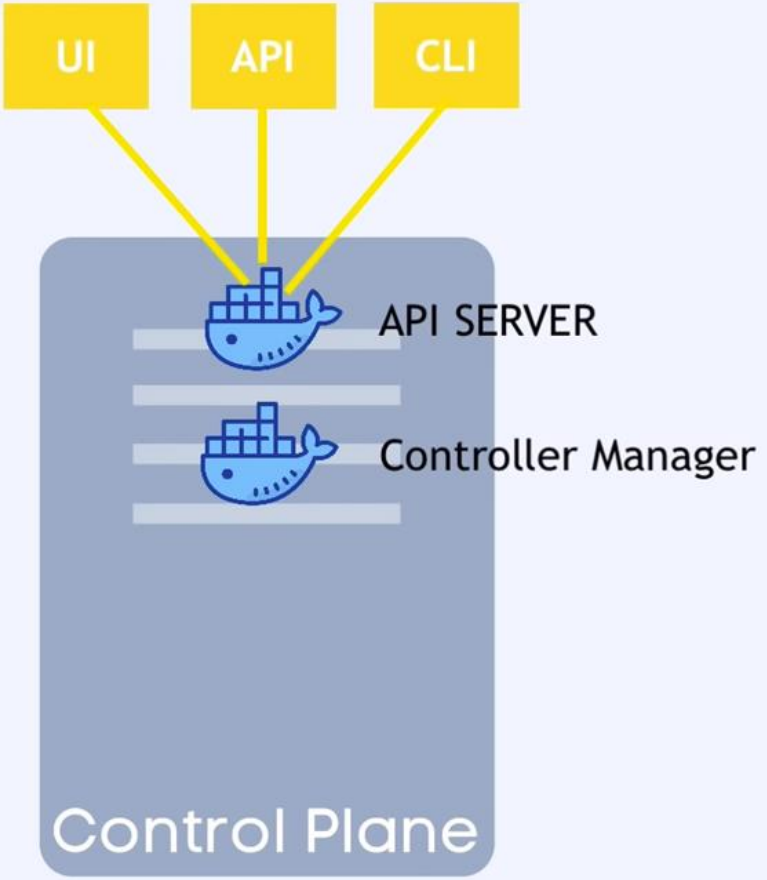


API Server =
Entrypoint to K8s cluster



API Server =
Entrypoint to K8s cluster

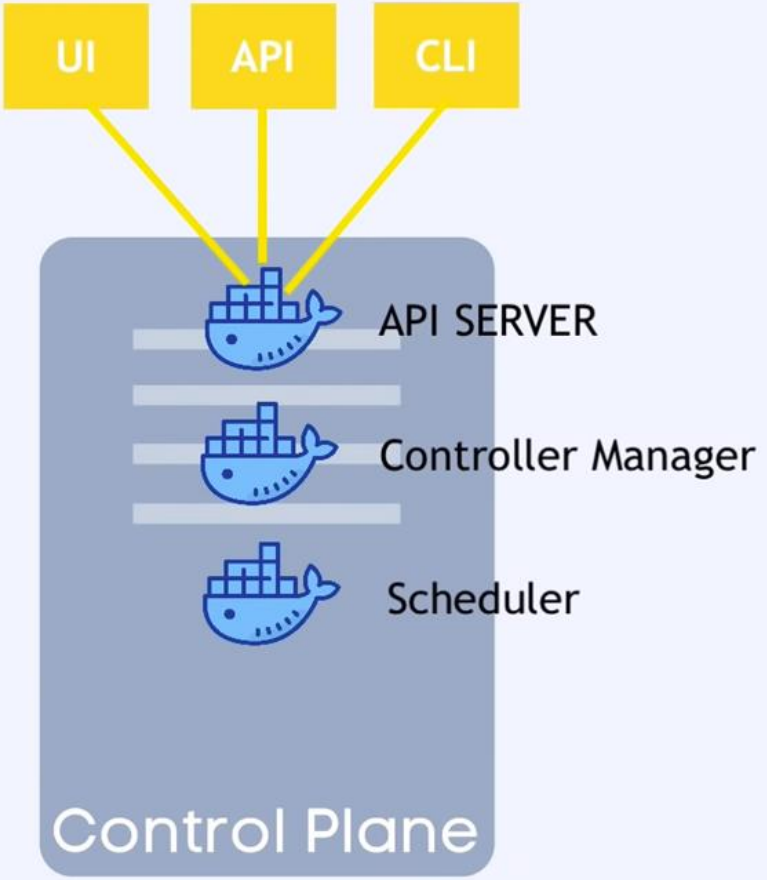
Controller Manager = keeps track of whats happening in the cluster



API Server =
Entrypoint to K8s cluster

Controller Manager = keeps track of whats happening in the cluster

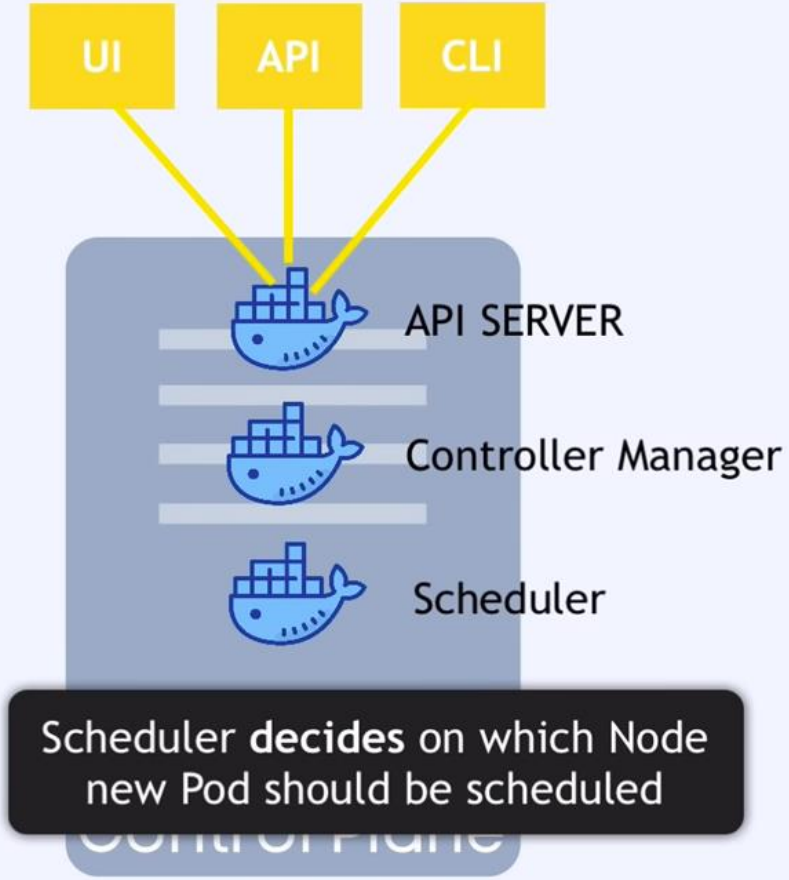
Scheduler =
ensures Pods placement



API Server =
Entrypoint to K8s cluster

Controller Manager = keeps track of whats happening in the cluster

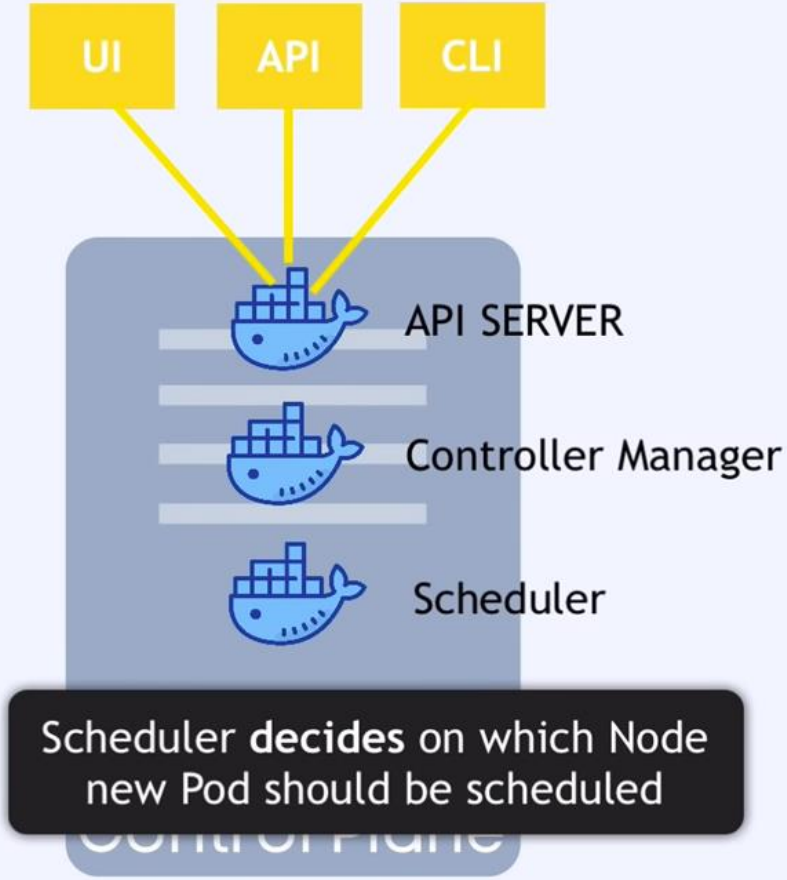
Scheduler =
ensures Pods placement



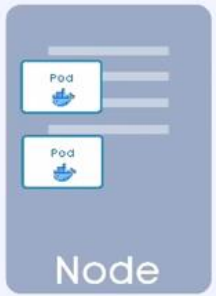
API Server =
Entrypoint to K8s cluster

Controller Manager = keeps track of whats happening in the cluster

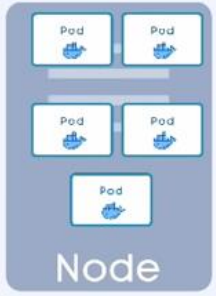
Scheduler =
ensures Pods placement



30% used



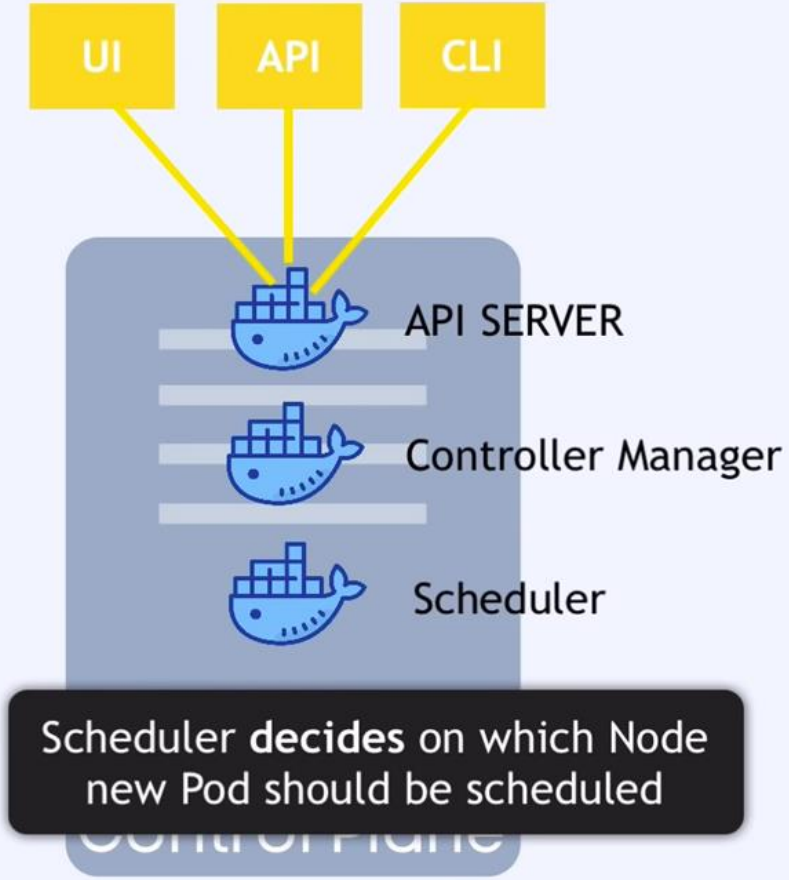
60% used



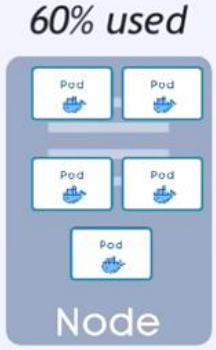
API Server =
Entrypoint to K8s cluster

Controller Manager = keeps track of whats happening in the cluster

Scheduler =
ensures Pods placement



Where to put the Pod?

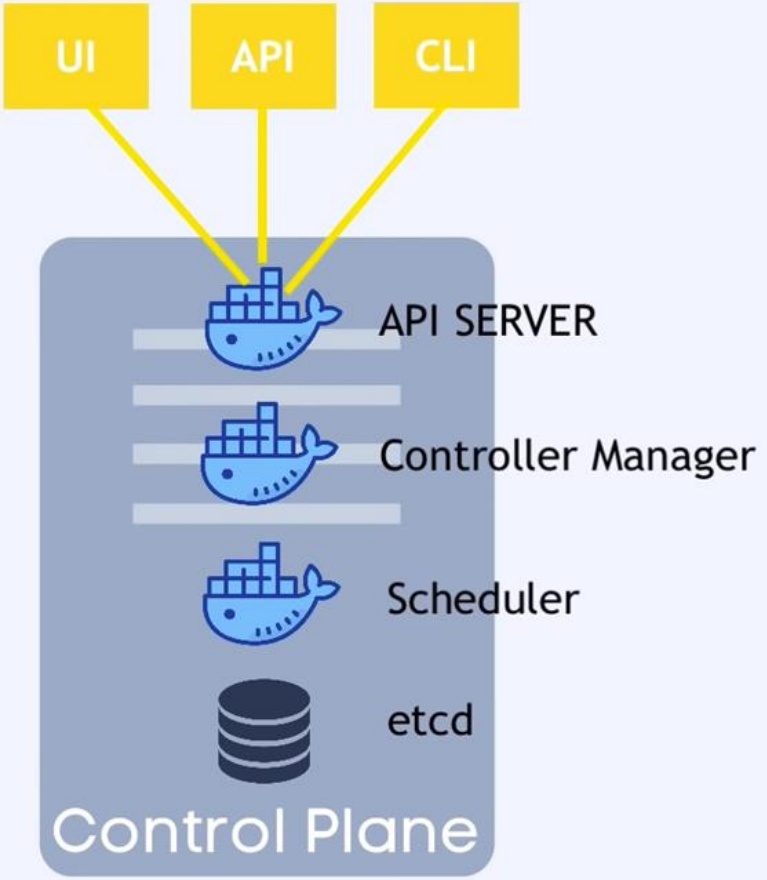


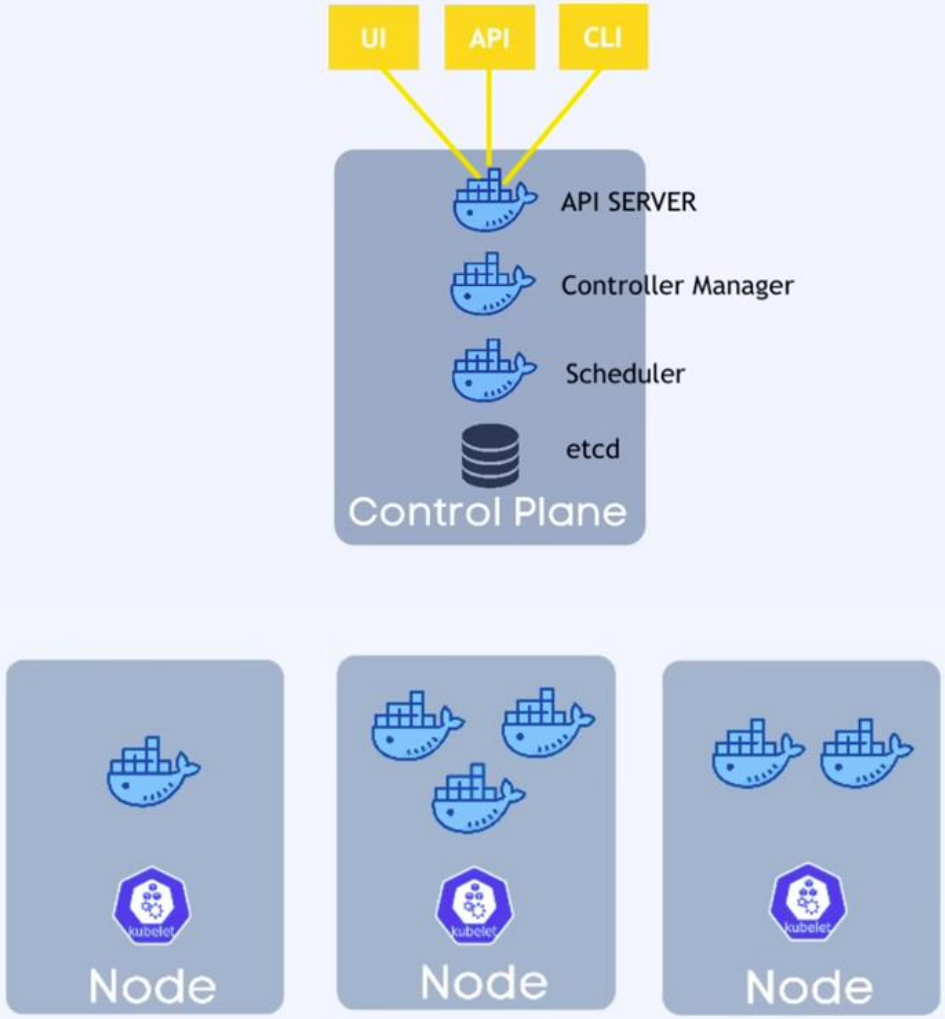
API Server =
Entrypoint to K8s cluster

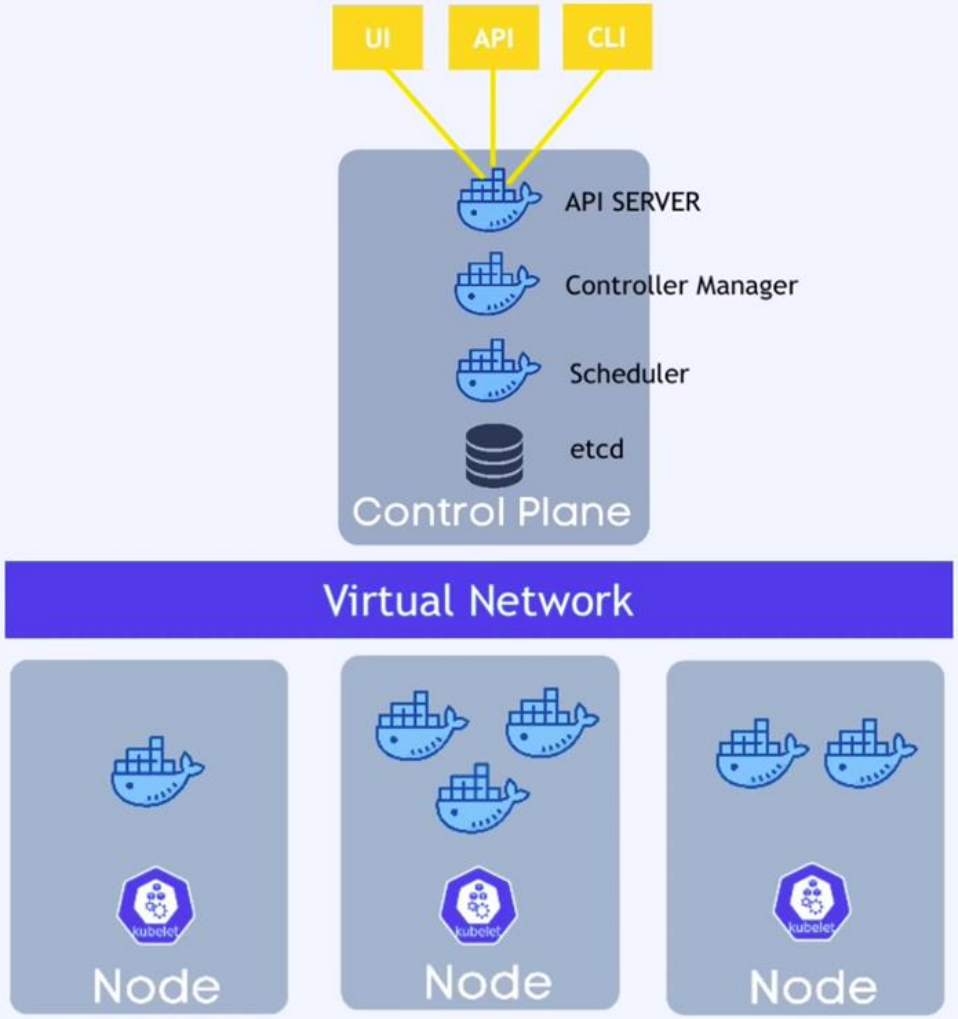
Controller Manager = keeps track of whats happening in the cluster

Scheduler =
ensures Pods placement

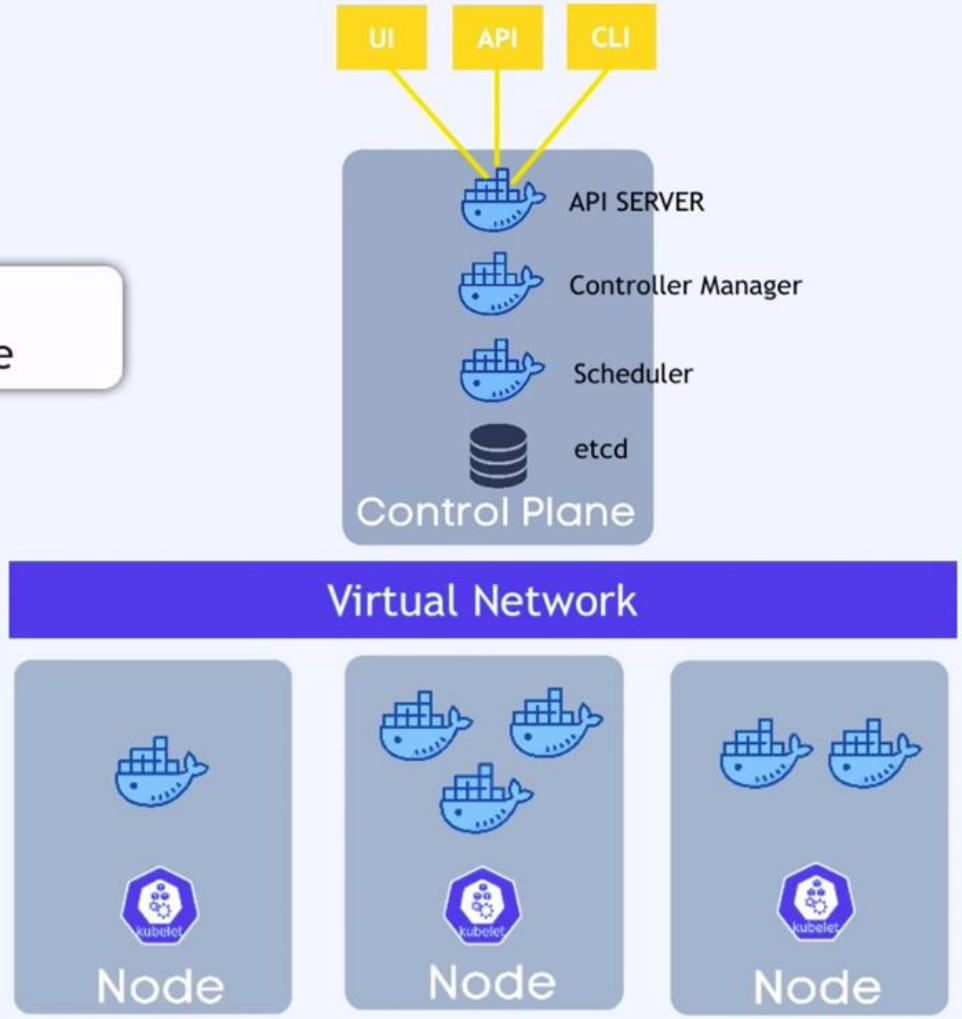
etcd =
Kubernetes backing store

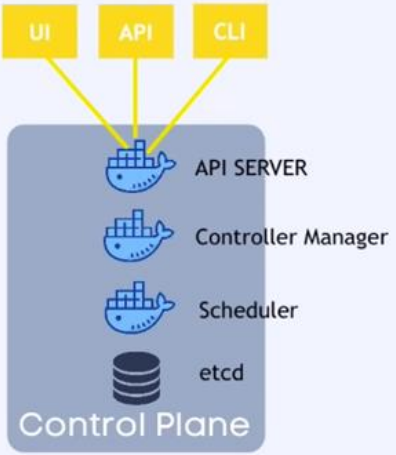



UI
API
CLI
API SERVER
Controller Manager
Scheduler
etcd
Control Plane
kubelet
Node
kubelet
Node
kubelet
Node


UI
API
CLI
API SERVER
Controller Manager
Scheduler
etcd
Control Plane
Virtual Network
kubelet
Node
kubelet
Node
kubelet
Node

Virtual Network =
Creates one unified machine





Worker Nodes

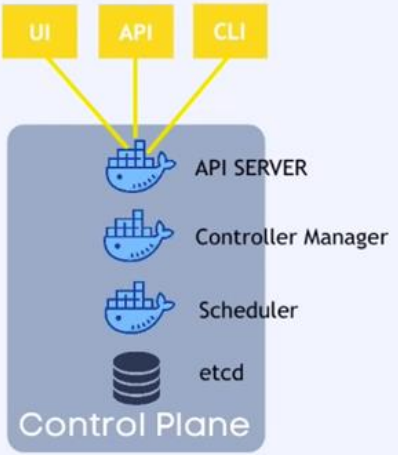
higher workload

much bigger and more resources

Control Plane Nodes

handful of master processes

much **more important**



Worker Nodes

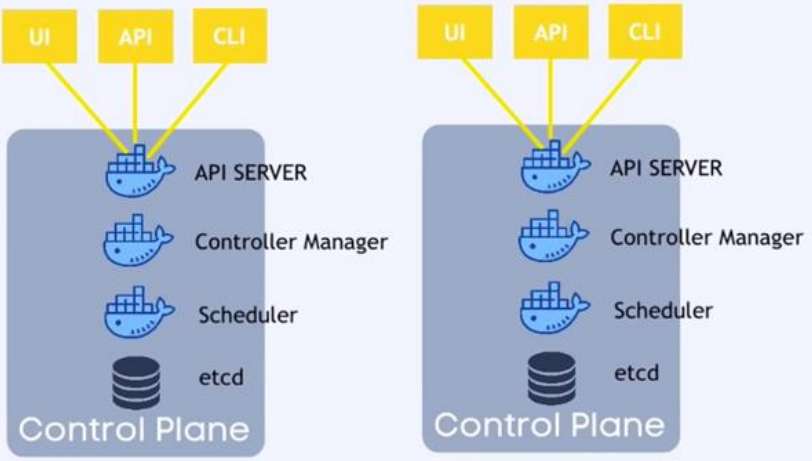
higher workload

much bigger and more resources

Control Plane Nodes

handful of master processes

much **more important**



Worker Nodes

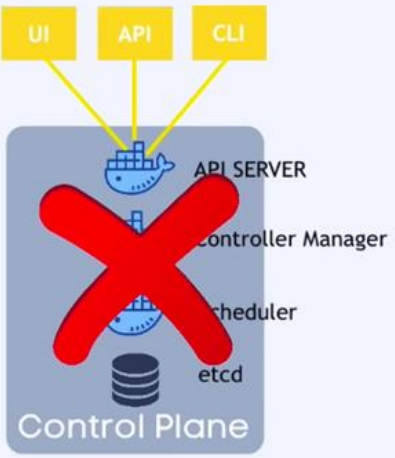
higher workload

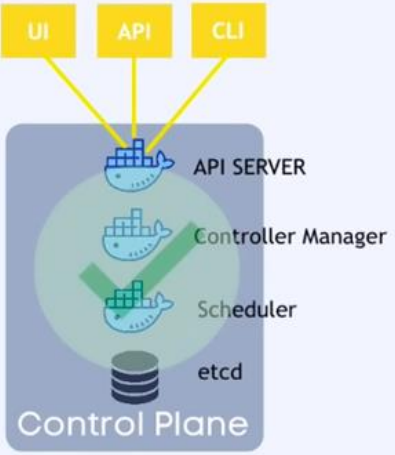
much bigger and more resources

Control Plane Nodes

handful of master processes

much **more important**



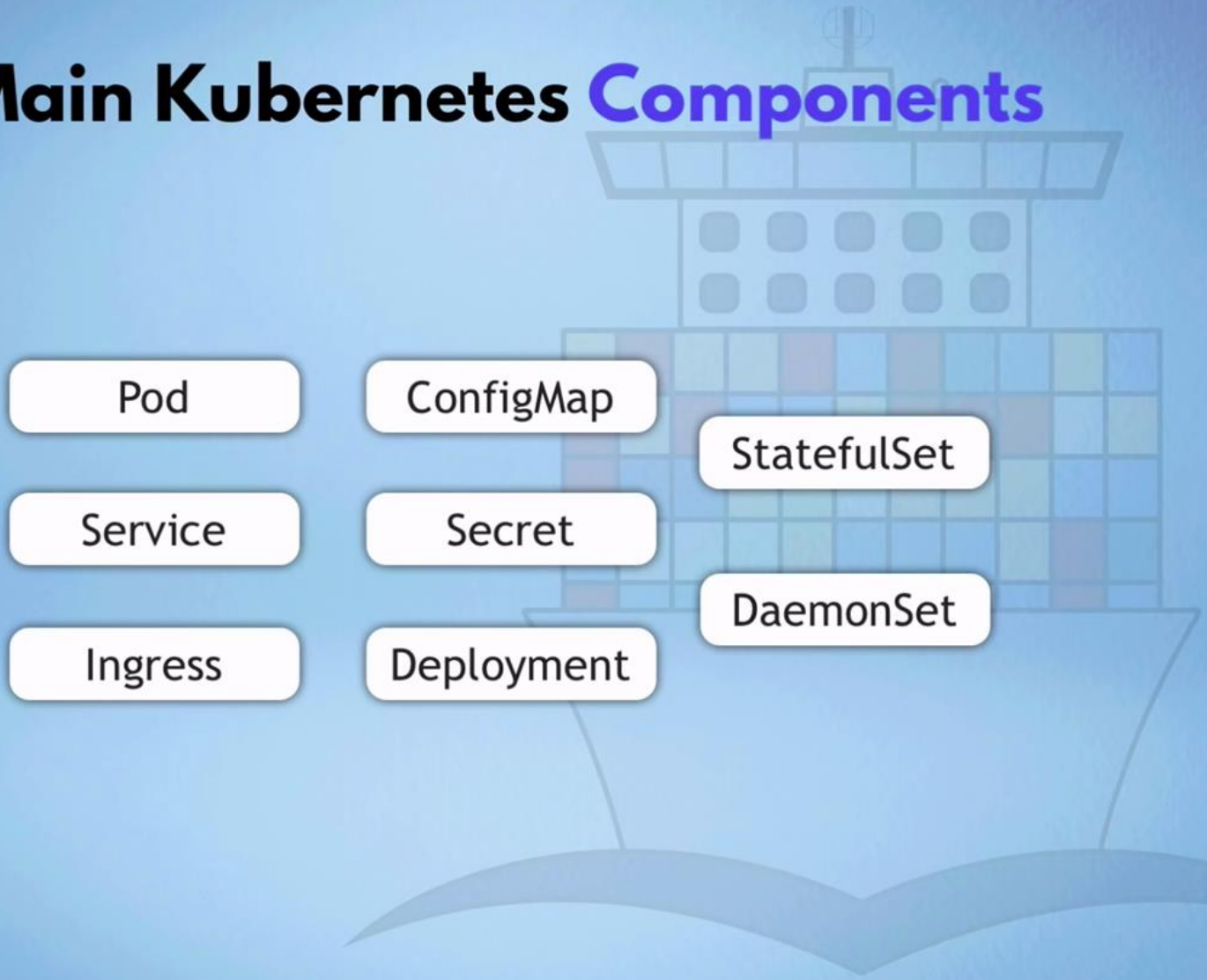


Worker Nodes

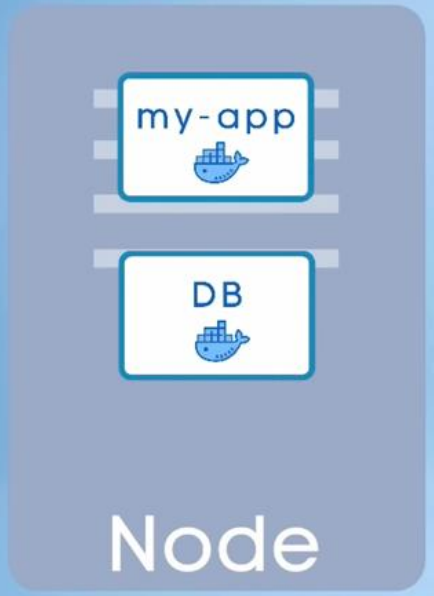
higher workload

much bigger and more resources

Main Kubernetes Components



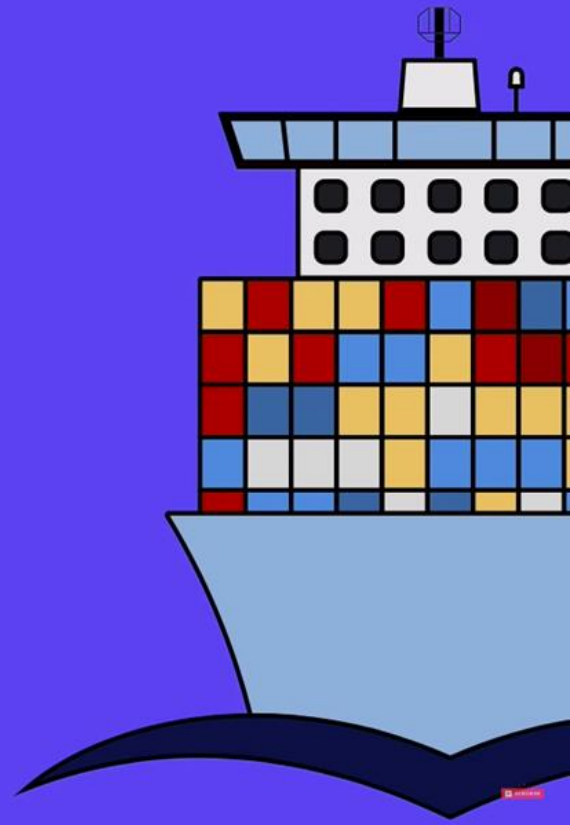
Main Kubernetes Components

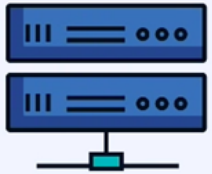


Role of each component

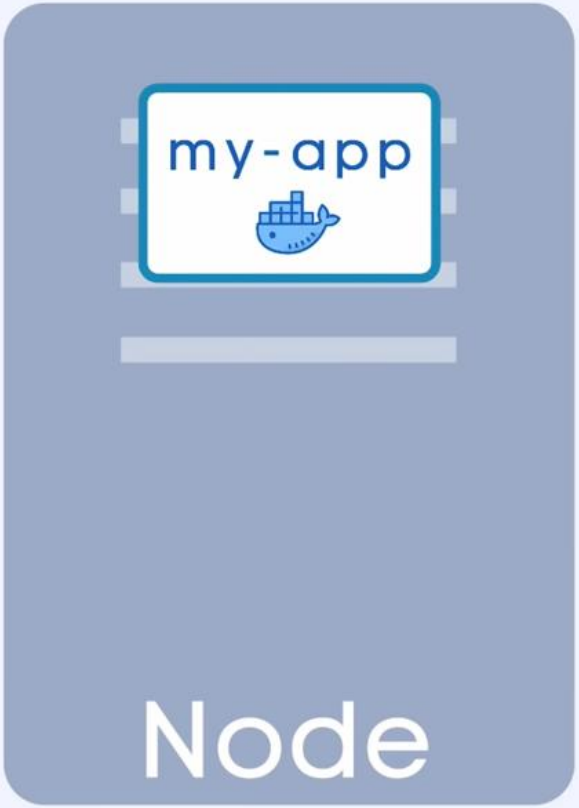
- Pod
- ConfigMap
- StatefulSet
- Service
- Secret
- DaemonSet
- Ingress
- Deployment

Node and Pod

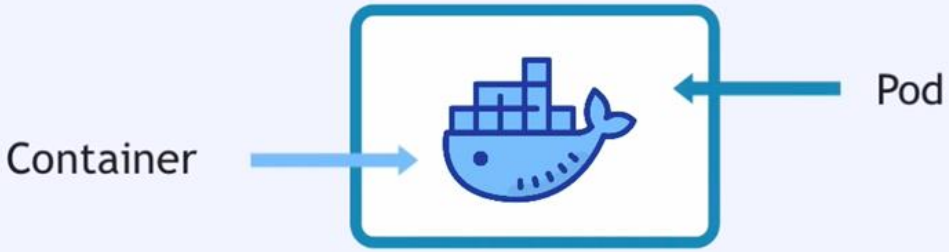


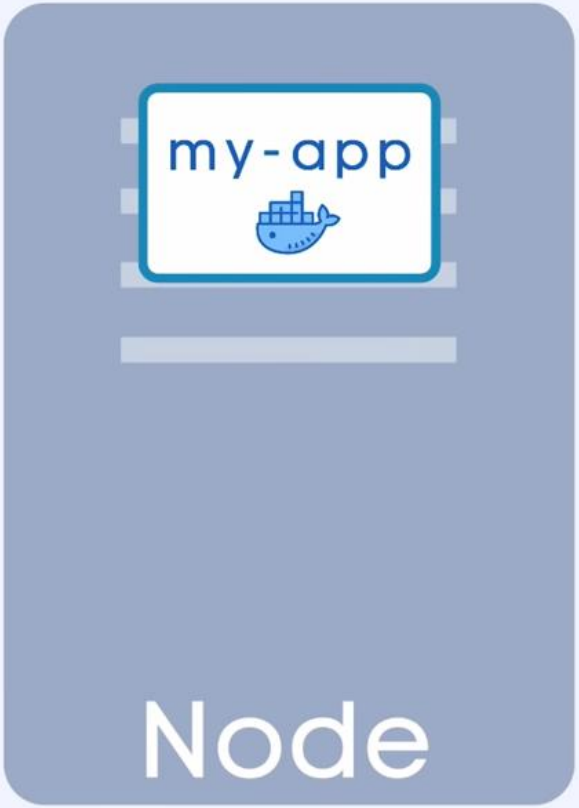


Node =
virtual or physical machine

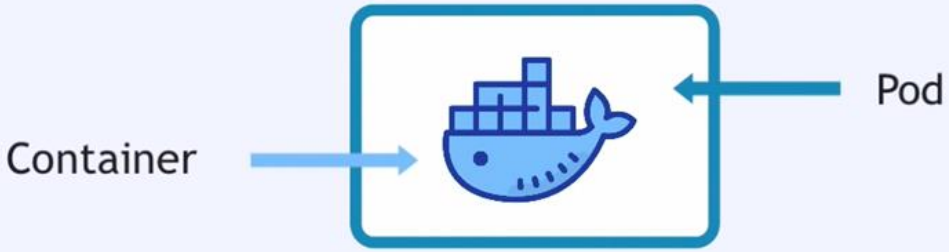


- **Smallest** unit in Kubernetes
- **Abstraction** over container

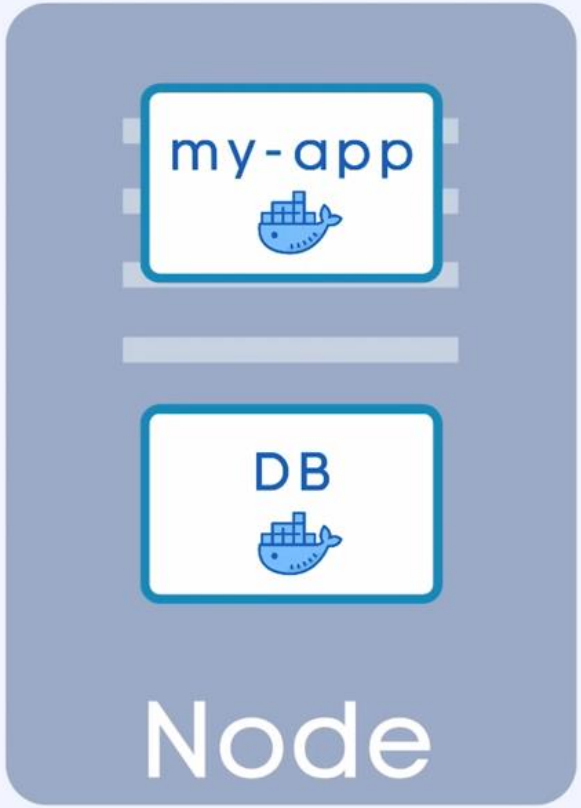




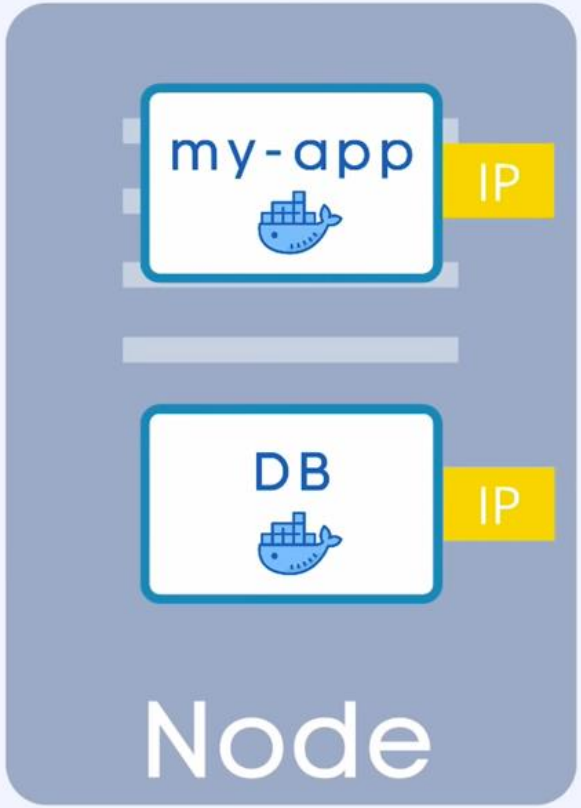
- **Smallest** unit in Kubernetes
- **Abstraction** over container



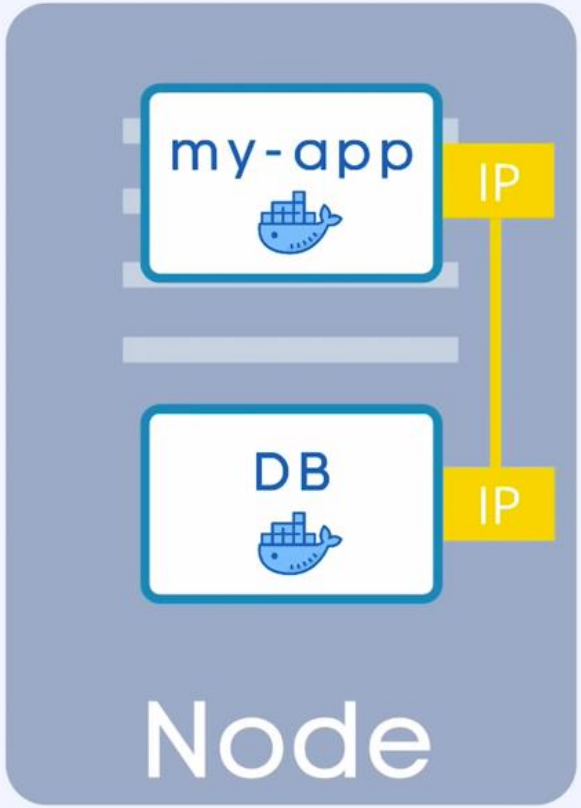
You only interact with the Kubernetes layer



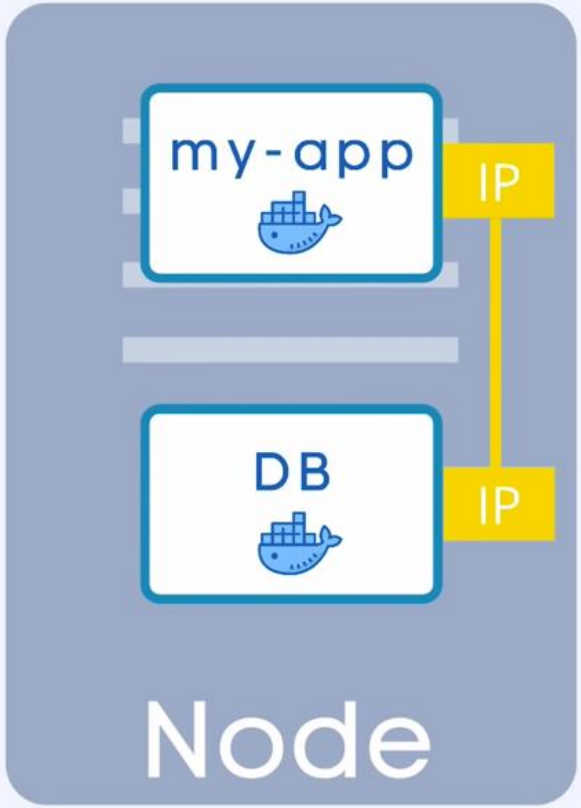
- **Smallest** unit in Kubernetes
- **Abstraction** over container
- Usually **1 Application** per Pod



- **Smallest** unit in Kubernetes
- **Abstraction** over container
- Usually **1 Application** per Pod

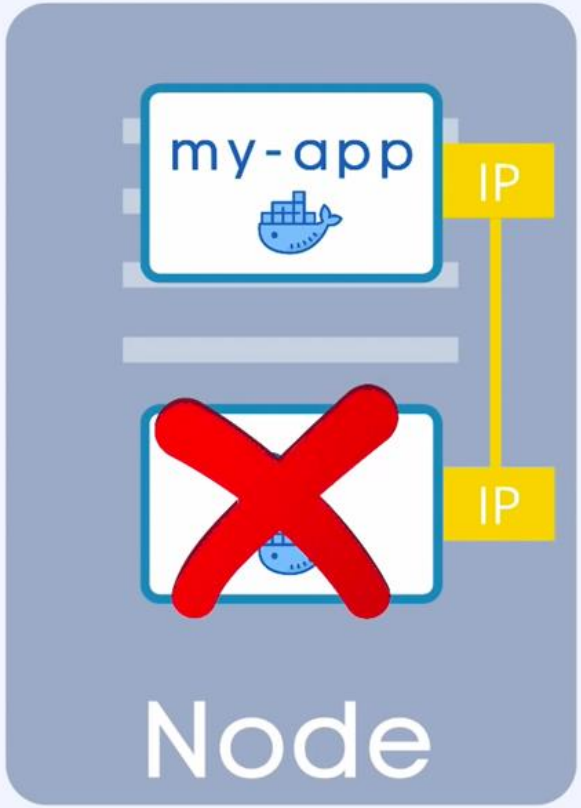


- **Smallest** unit in Kubernetes
- **Abstraction** over container
- Usually **1 Application** per Pod
- Each Pod gets its **own IP address**

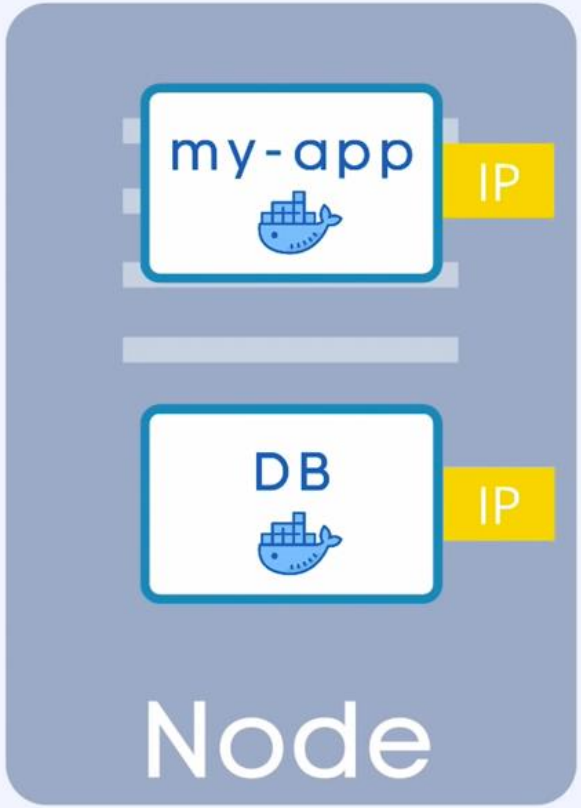


- **Smallest** unit in Kubernetes
- **Abstraction** over container
- Usually **1 Application** per Pod
- Each Pod gets its **own IP address**

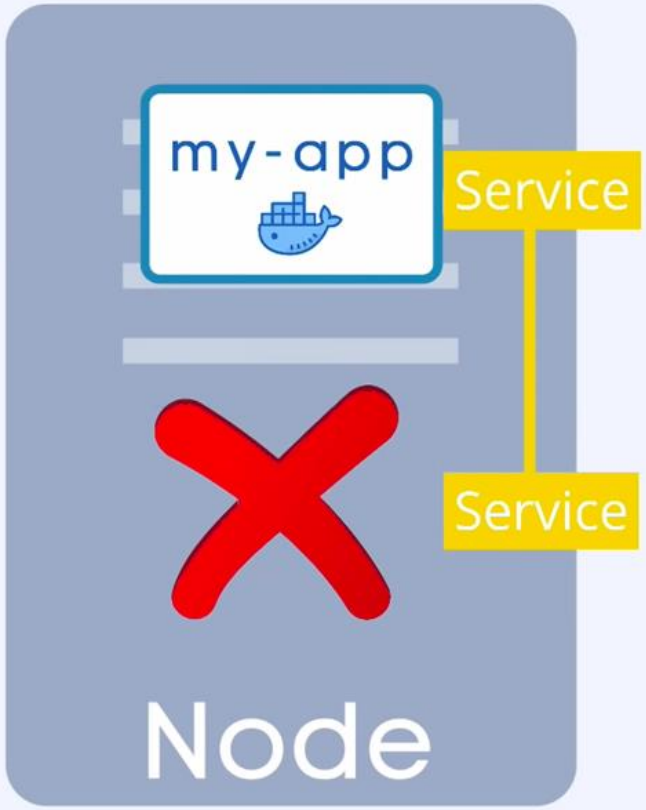
Pods are **ephemeral!**



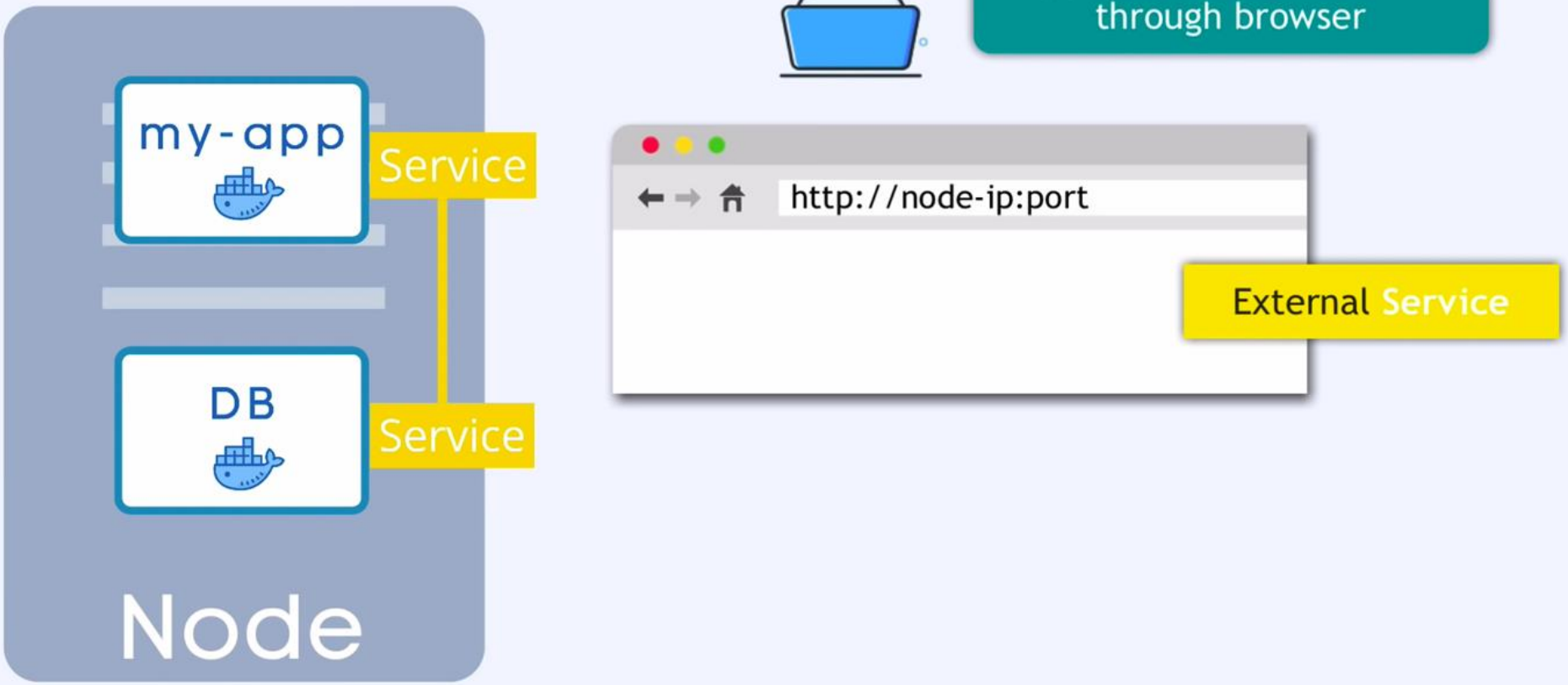
- **Smallest** unit in Kubernetes
- **Abstraction** over container
- Usually **1 Application** per Pod
- Each Pod gets its **own IP address**

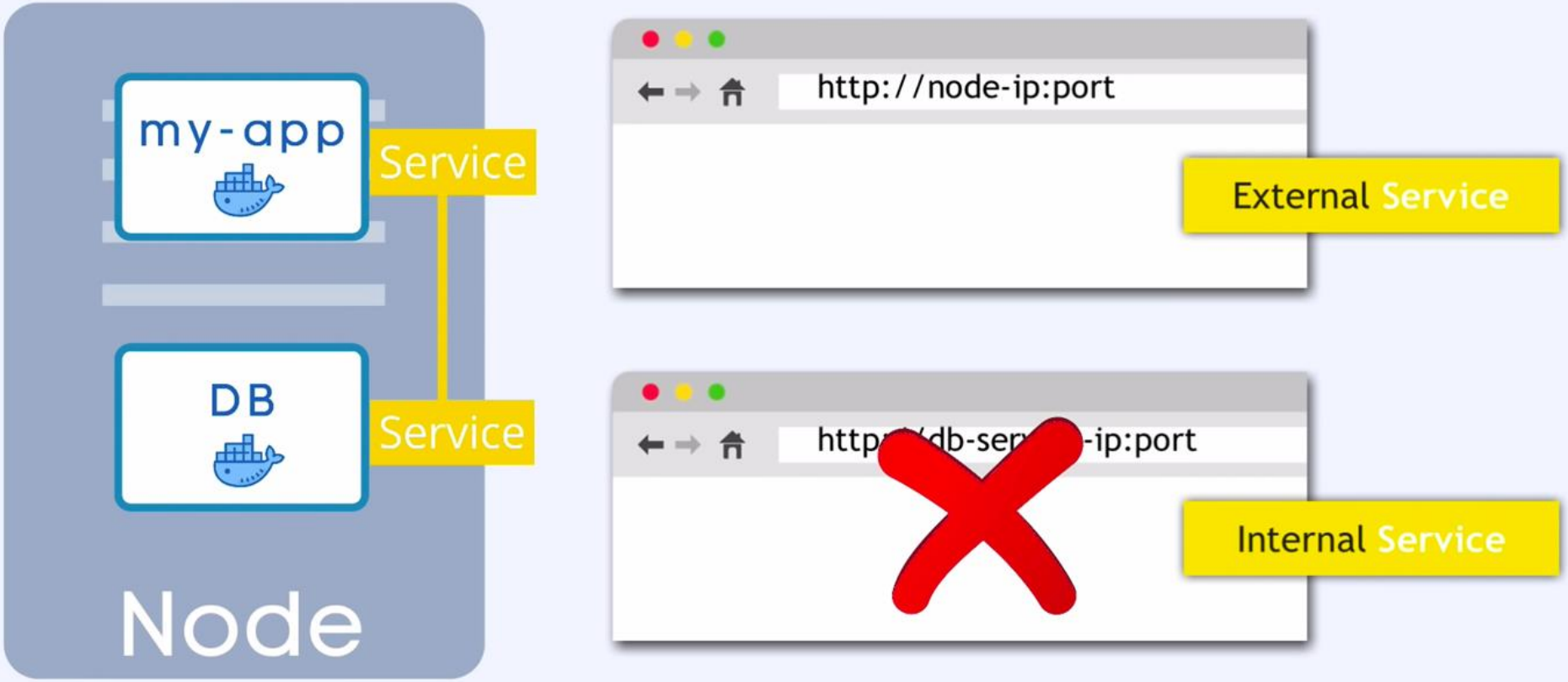


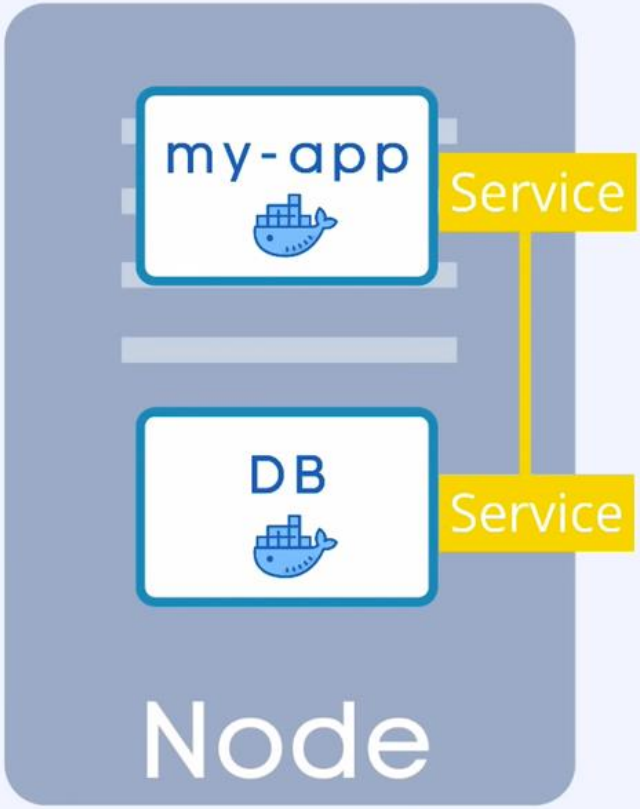
- **Smallest** unit in Kubernetes
- **Abstraction** over container
- Usually **1 Application** per Pod
- Each Pod gets its **own IP address**
- **New IP address** on re-creation

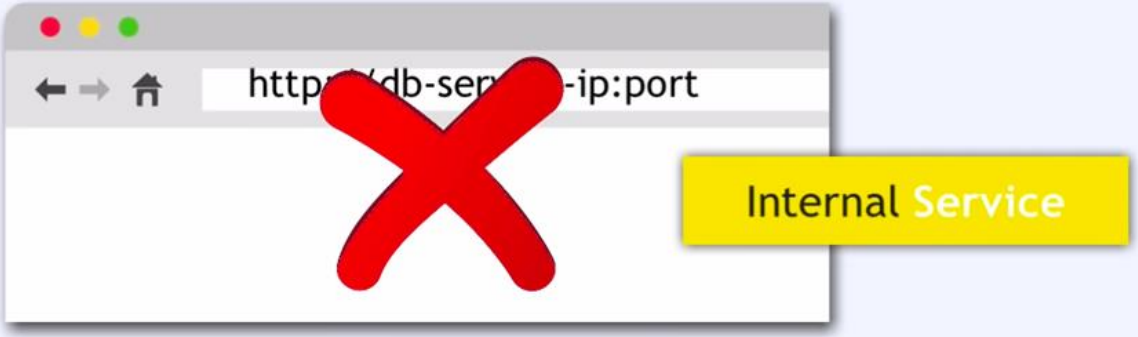


- **Permanent** IP address
- Lifecycle of Pod and Service **not connected**

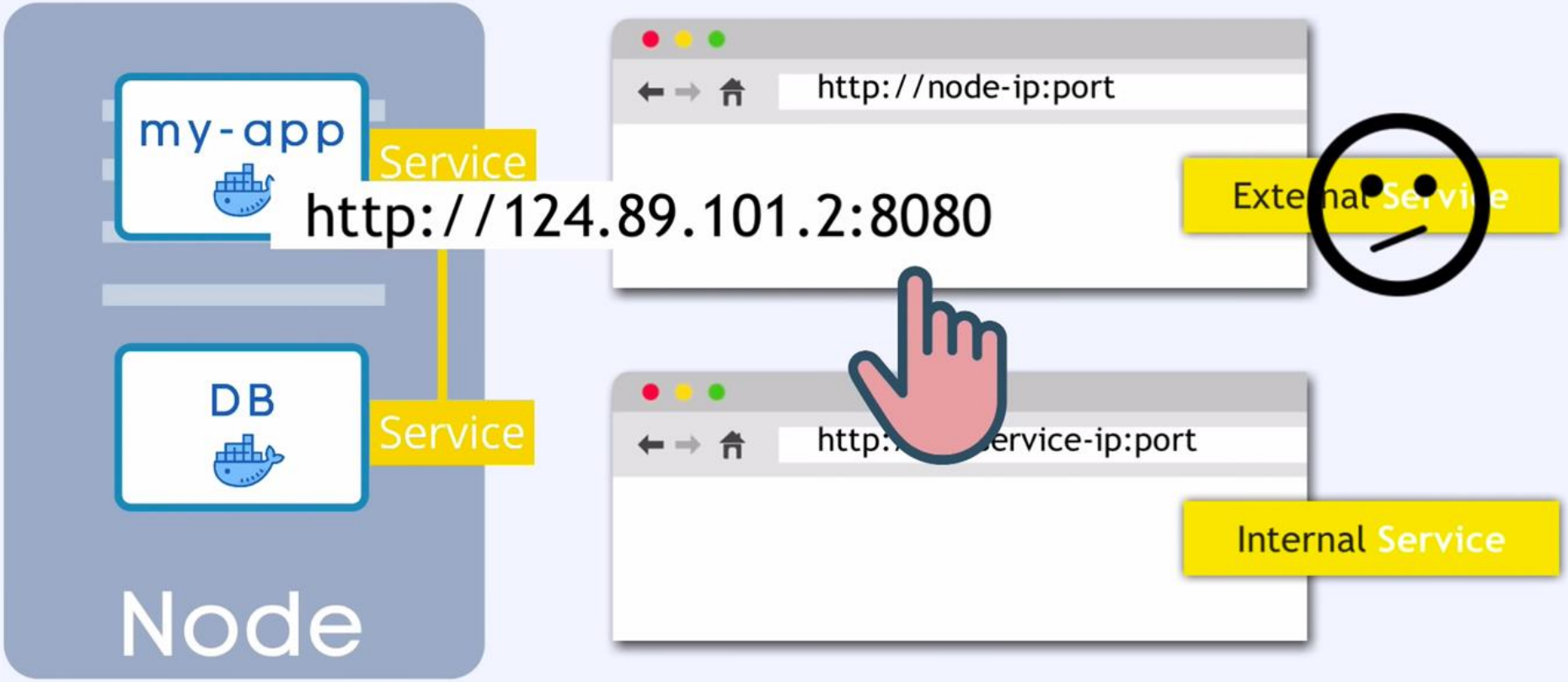

App should be accessible through browser
my-app
Service
DB
Service
Node
http://node-ip:port
External Service

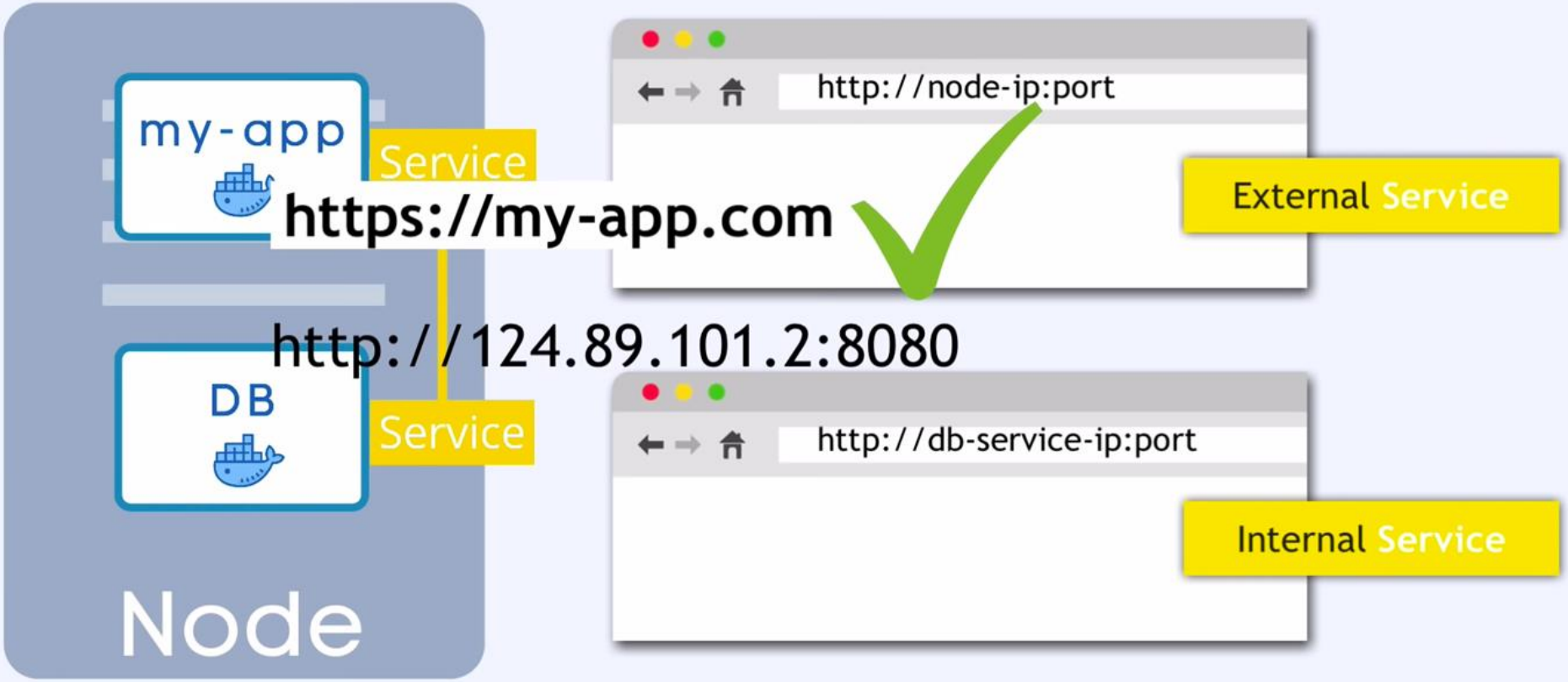

my-app
Service
DB
Service
Node
http://node-ip:port
External Service
Internal Service

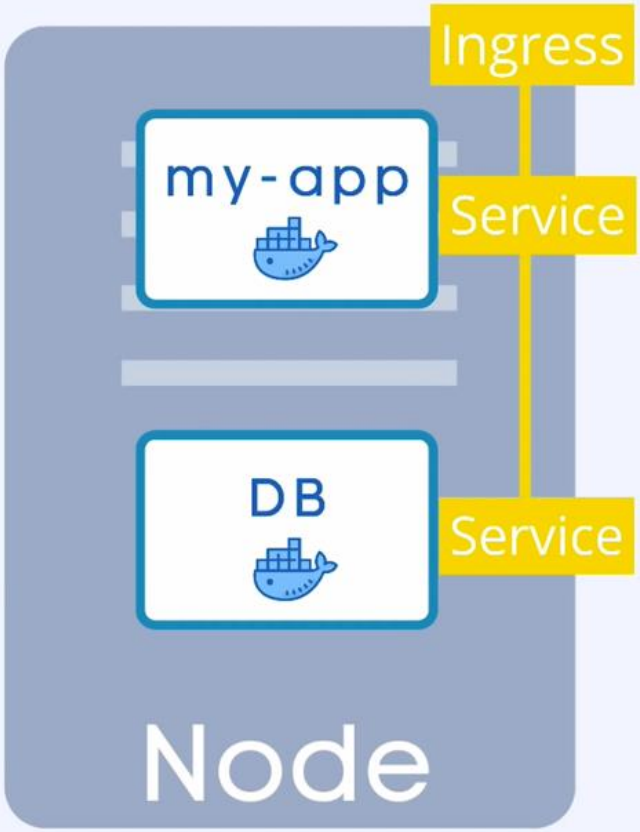




- You specify the **type of Service** on creation
- Internal Service is the **default** type


my-app
Service
http://124.89.101.2:8080
DB
Service
Node
http://node-ip:port
External Service
Internal Service

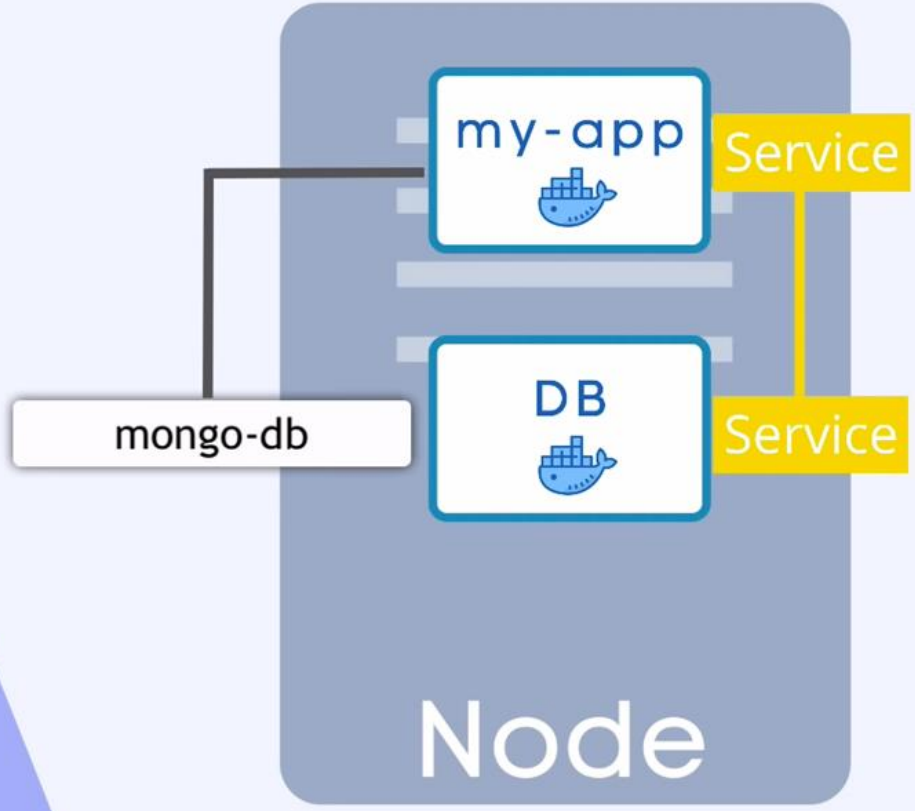

my-app
Service
https://my-app.com
http://node-ip:port
External Service
http://124.89.101.2:8080
DB
Service
http://db-service-ip:port
Internal Service
Node

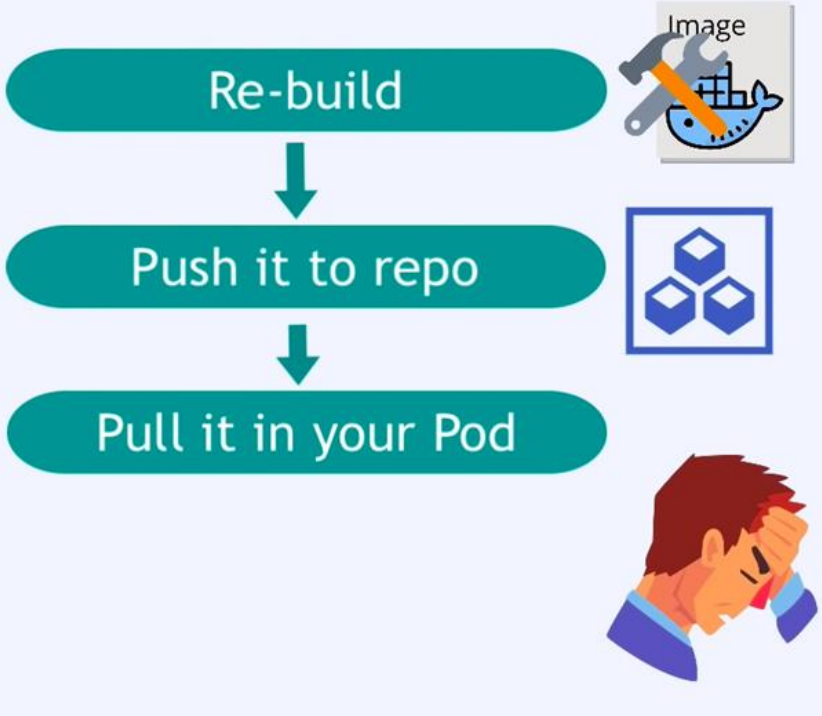

Ingress
my-app
Service
DB
Service
Node

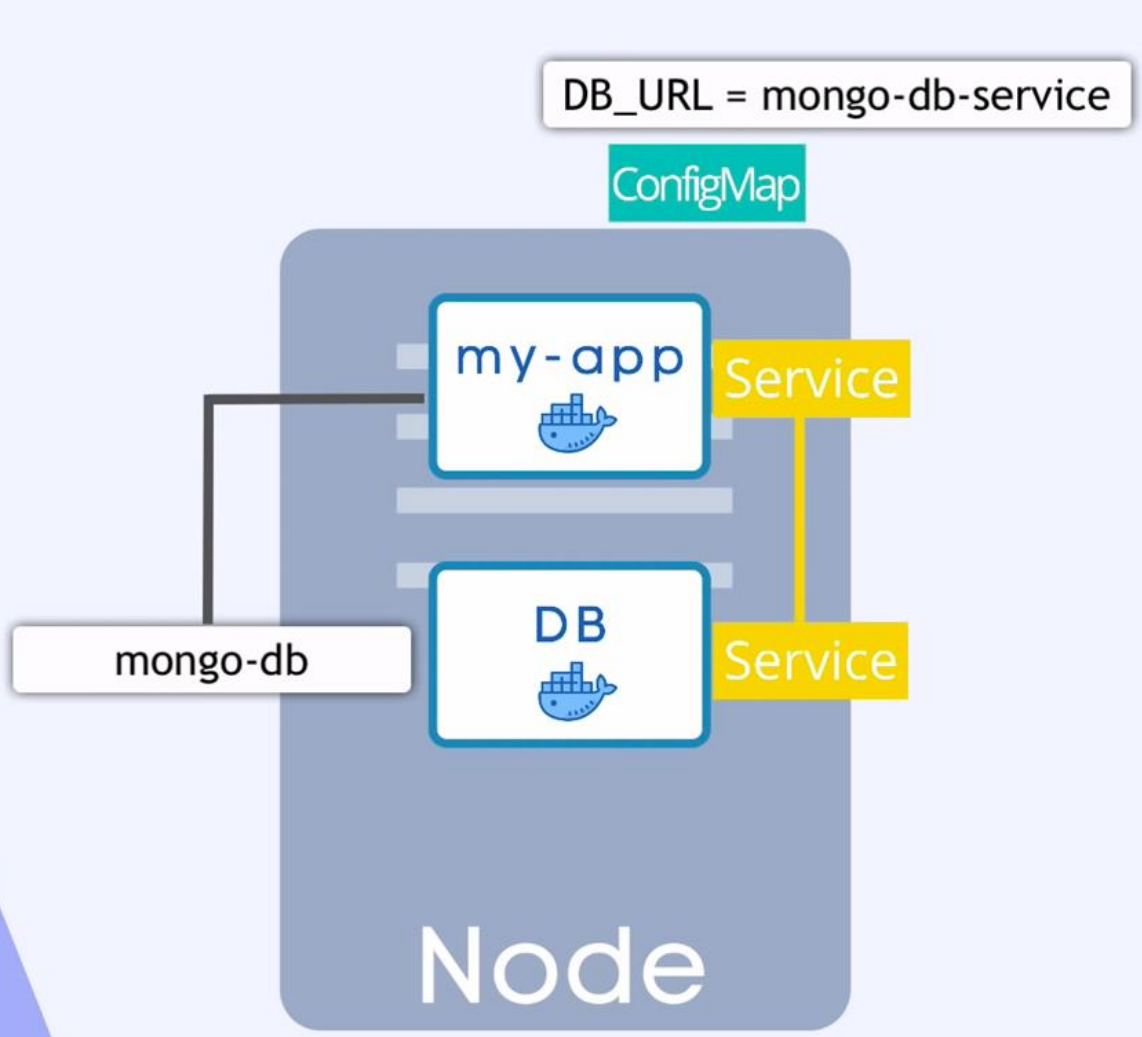

ing


https://my-app.com
Ingress

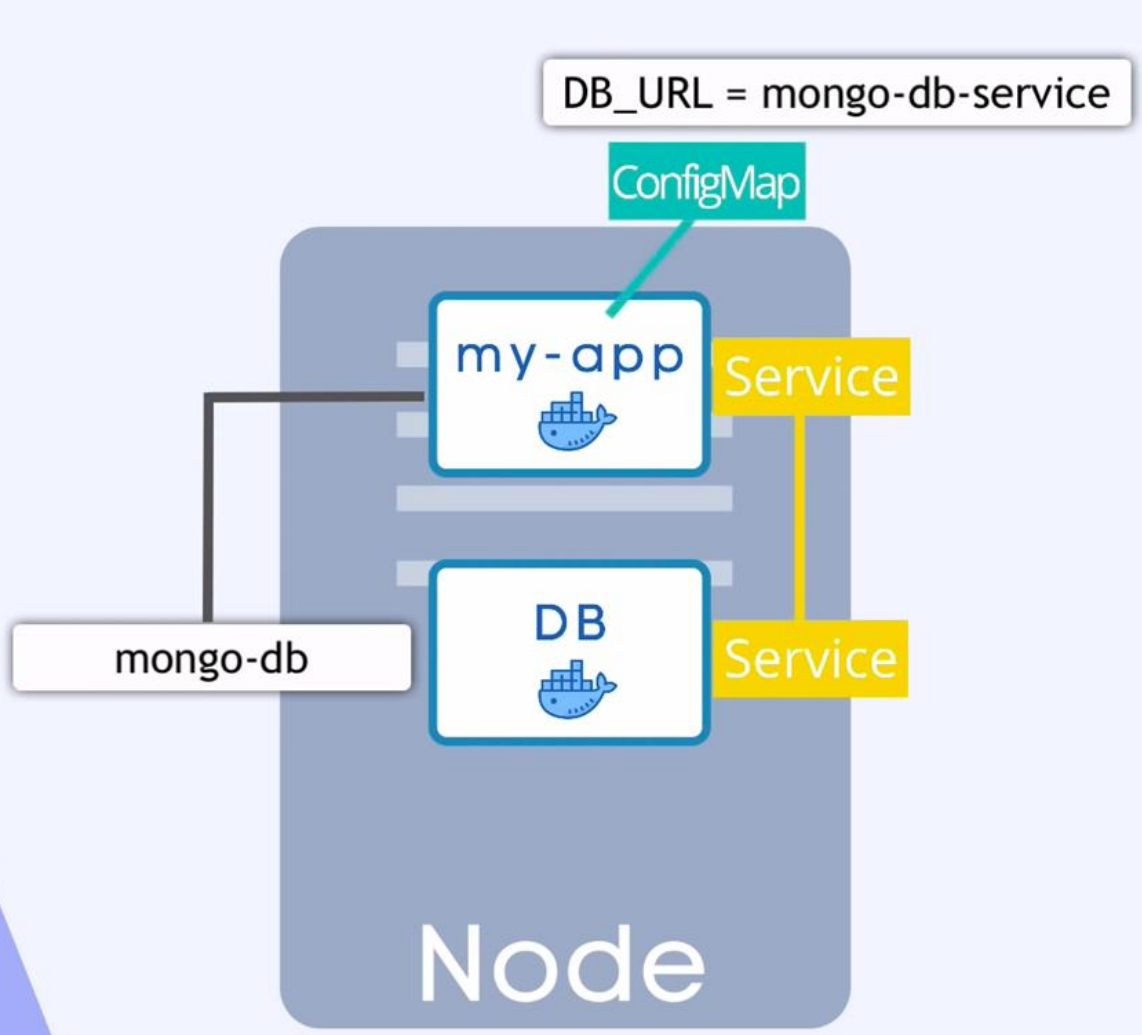
Database URL usually in the **built** application!



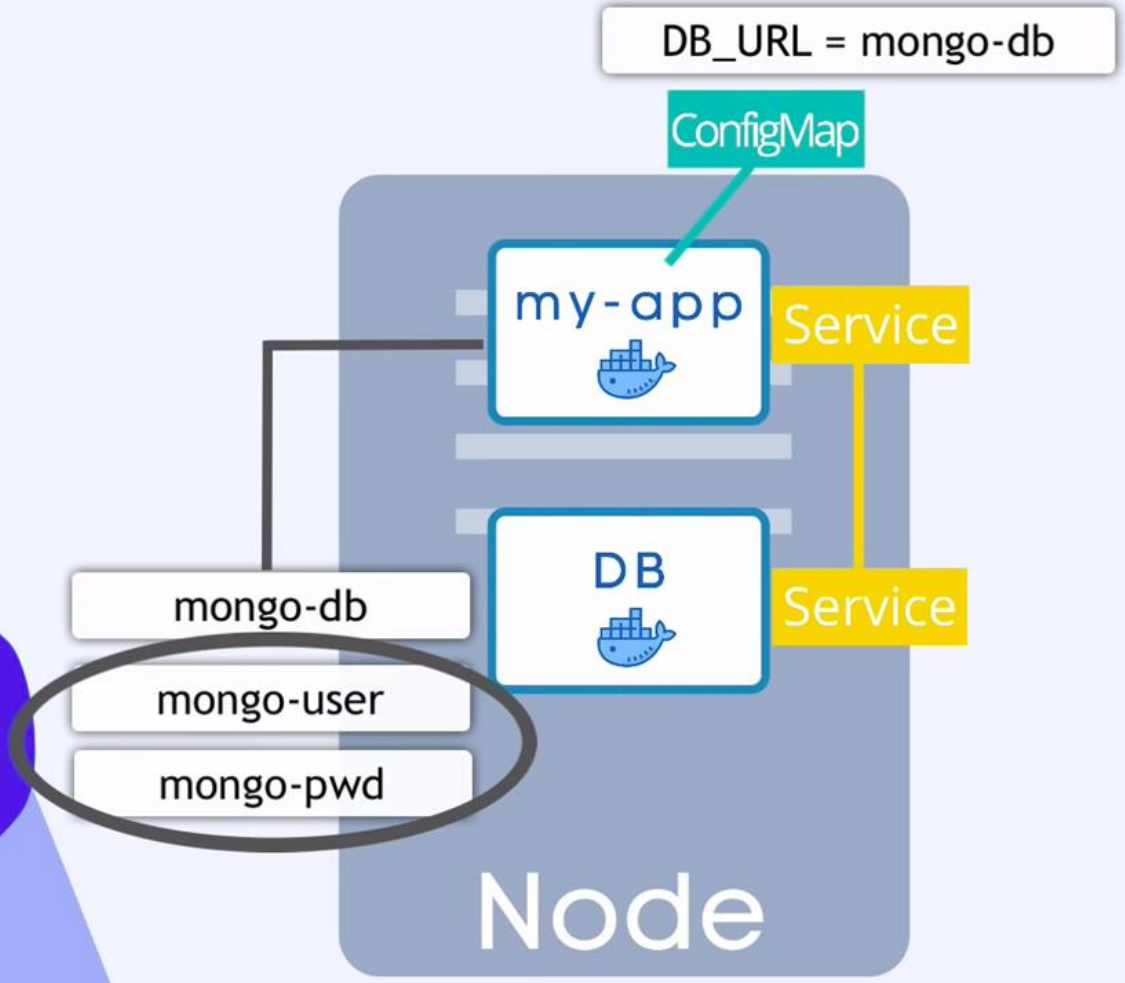


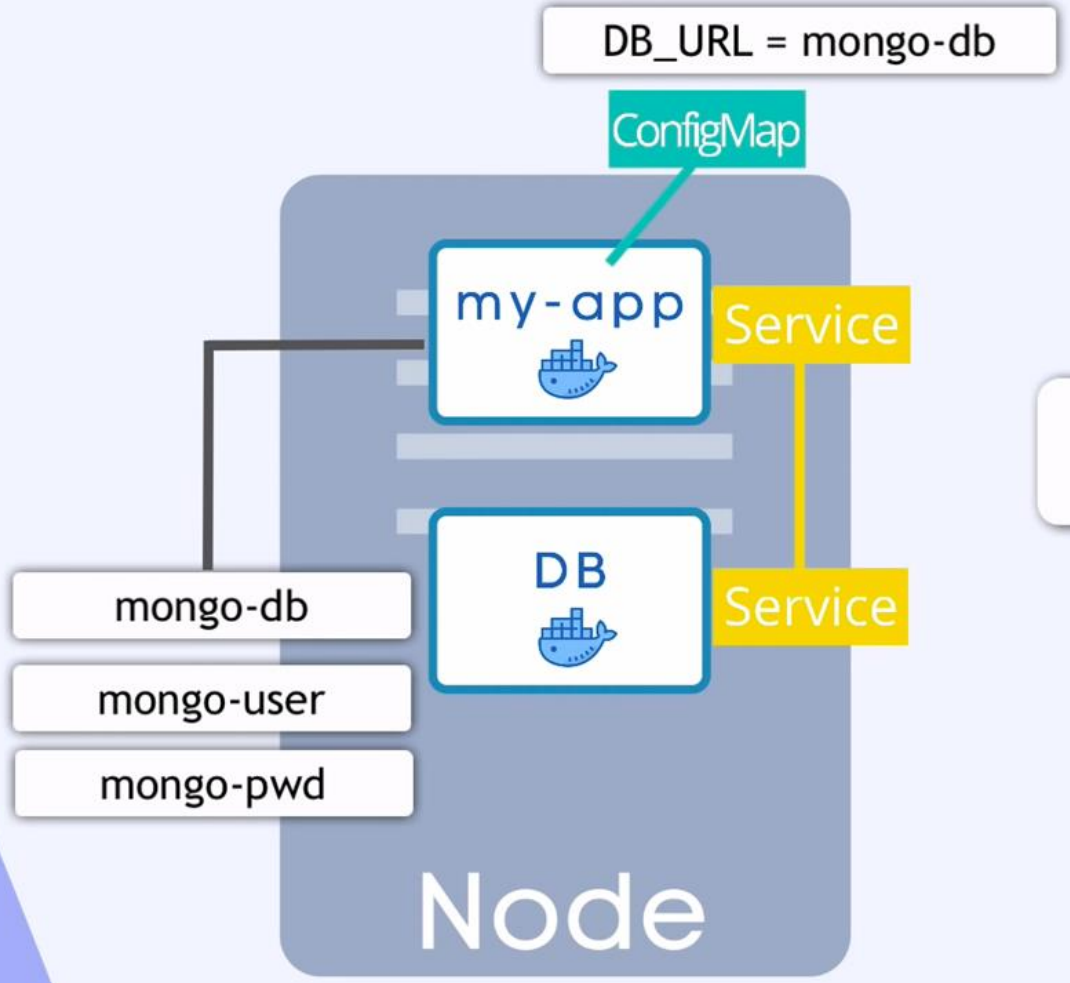


- **External** Configuration of your application

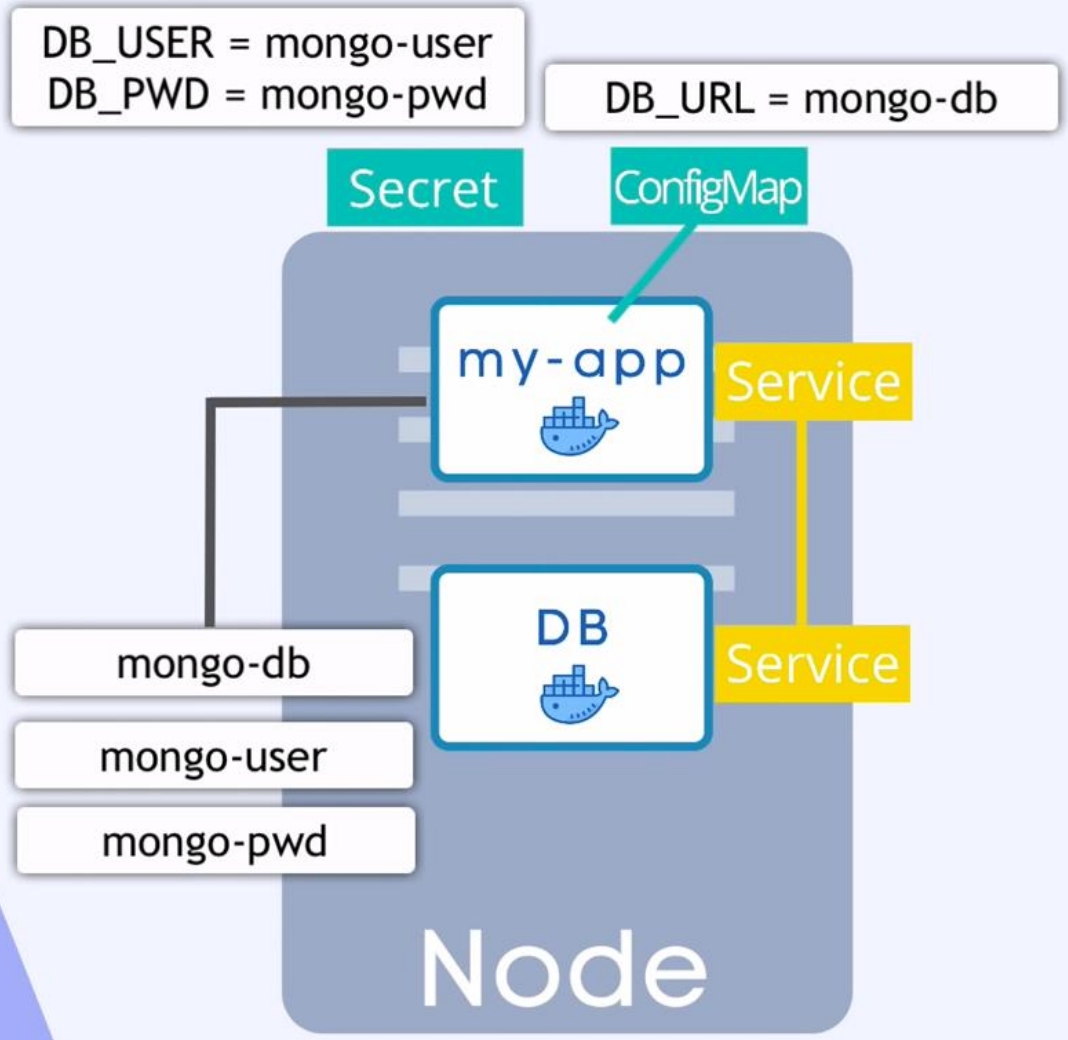


- **External** Configuration of your application


DB_URL = mongo-db
ConfigMap
my-app
Service
DB
Service
mongo-db
mongo-user
mongo-pwd
Node



! ConfigMap is for non-confidential data only!



- Used to store **secret data**

DB_USER = mongo-user
DB_PWD = mongo-pwd

DB_URL = mongo-db

Secret

mongo-db

mongo-user

mongo-pwd

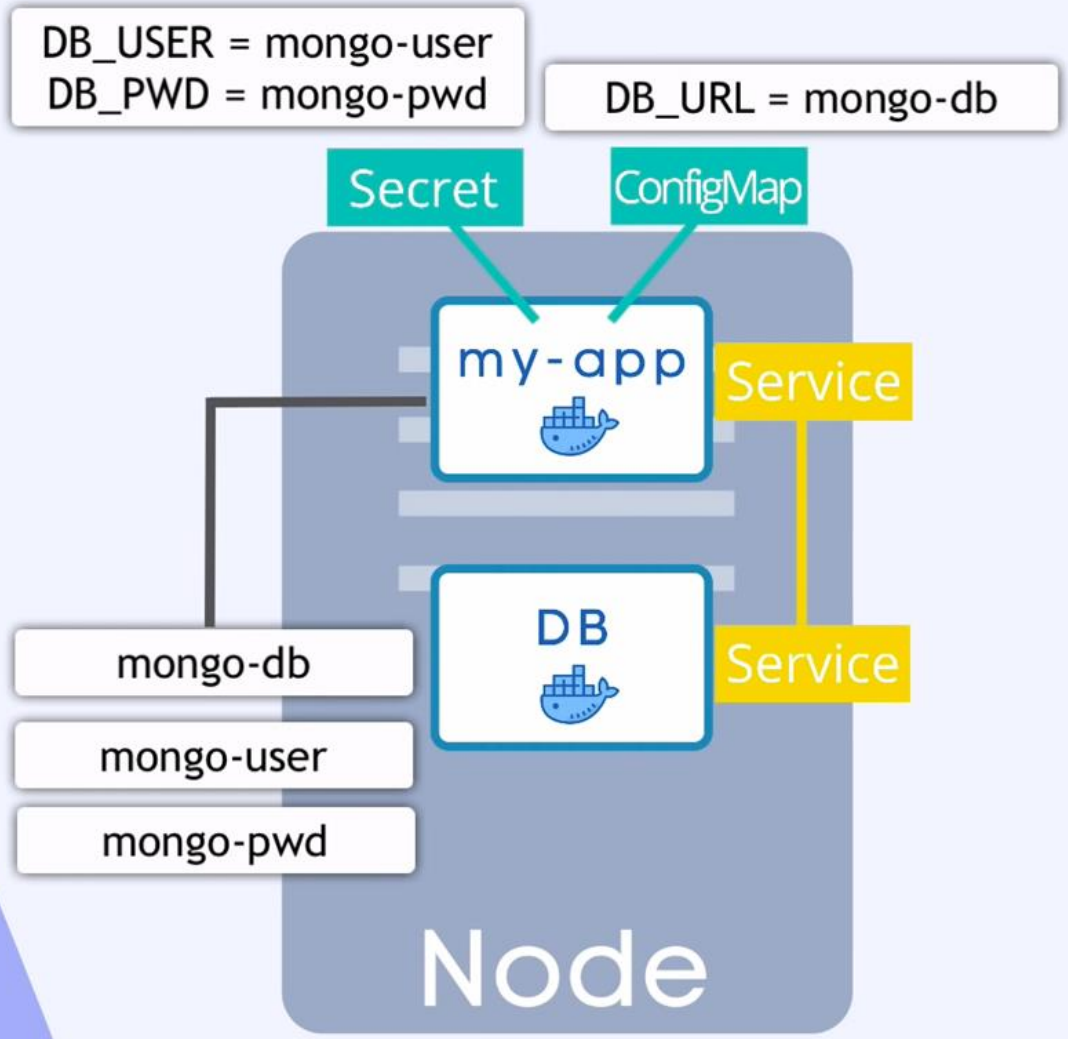
❗ The built-in security mechanism is not enabled by default!

Check out the official docs for more info on this:

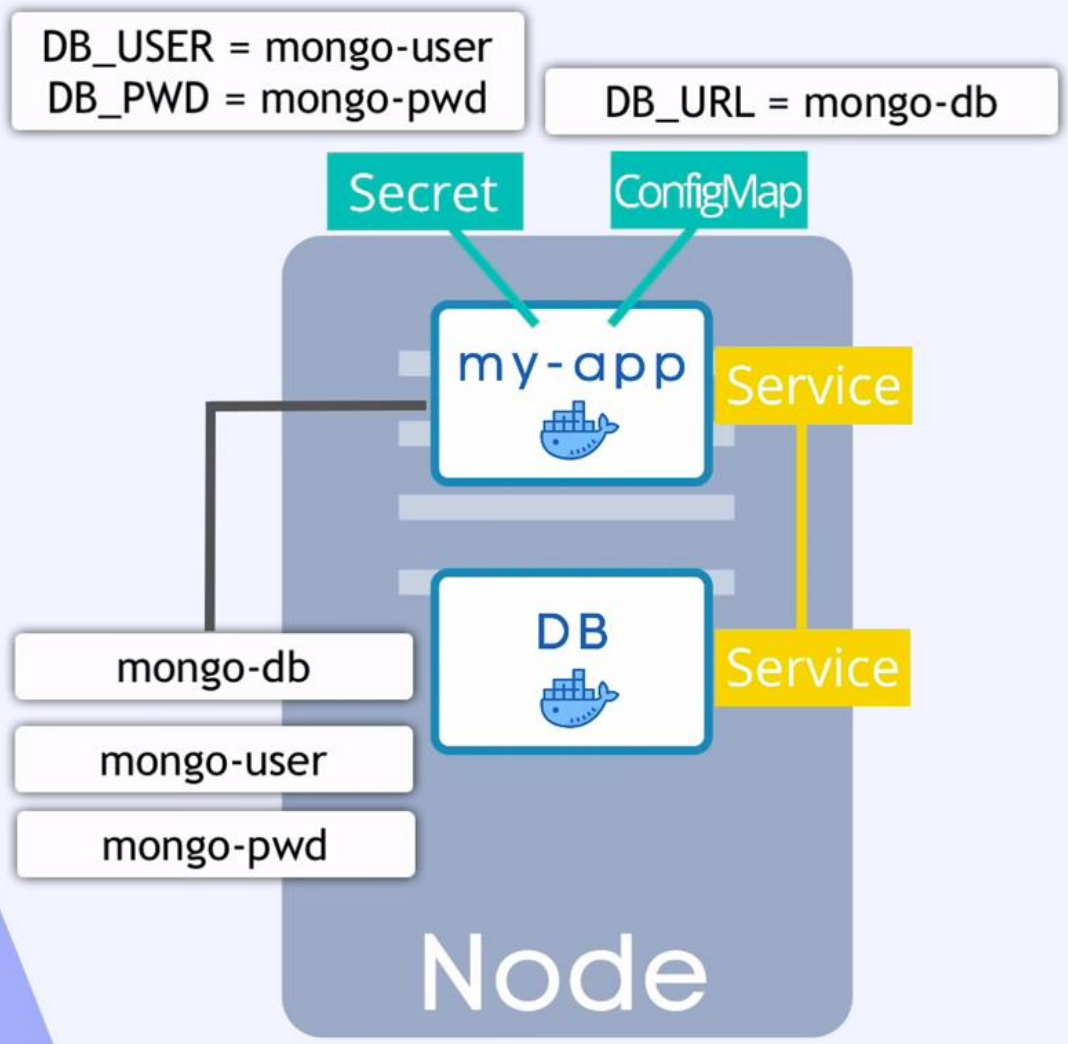
Caution:
Kubernetes Secrets are, by default, stored unencrypted in the API server's underlying data store (etcd). Anyone with API access can retrieve or modify a Secret, and so can anyone with access to etcd. Additionally, anyone who is authorized to create a Pod in a namespace can use that access to read any Secret in that namespace; this includes indirect access such as the ability to create a Deployment.

In order to safely use Secrets, take at least the following steps:

1. Enable Encryption at Rest for Secrets.
2. Enable or configure RBAC rules that restrict reading data in Secrets (including via indirect means).
3. Where appropriate, also use mechanisms such as RBAC to limit which principals are allowed to create new Secrets or replace existing ones.

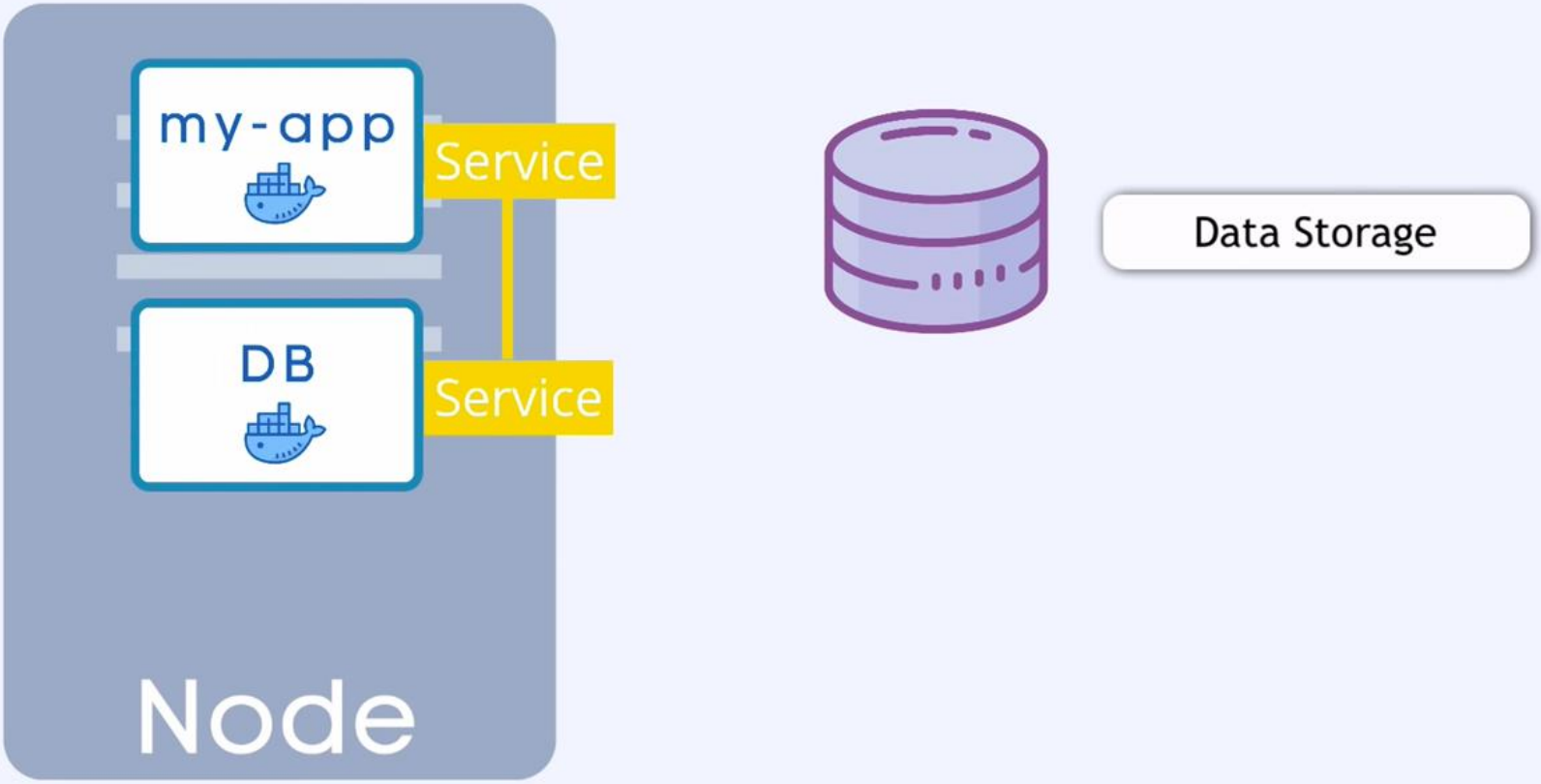


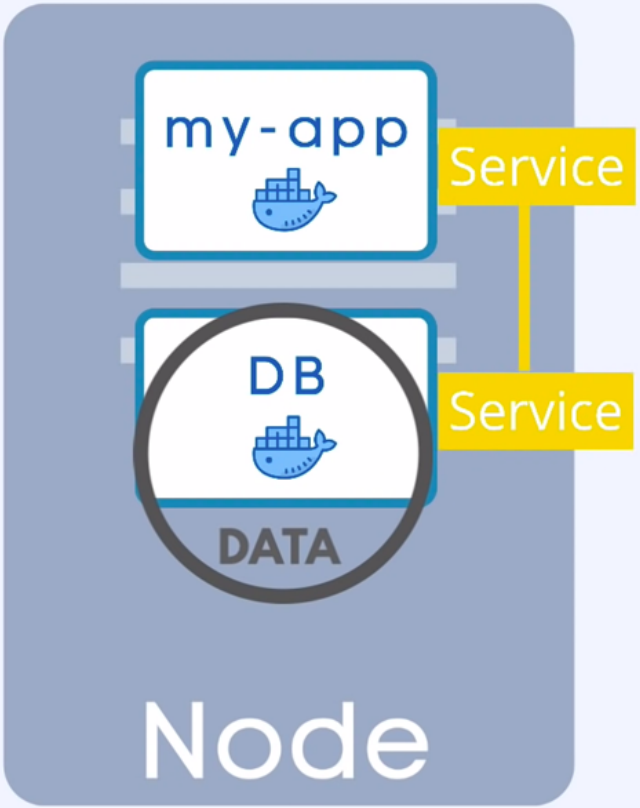
- Used to store **secret data**
- Reference Secret in Deployment/Pod

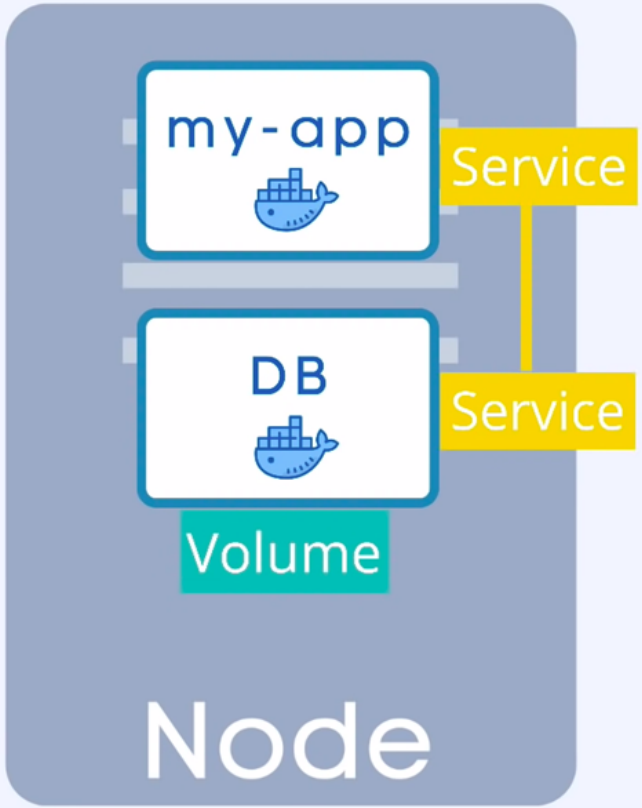


- Used to store **secret data**
- Reference Secret in Deployment/Pod

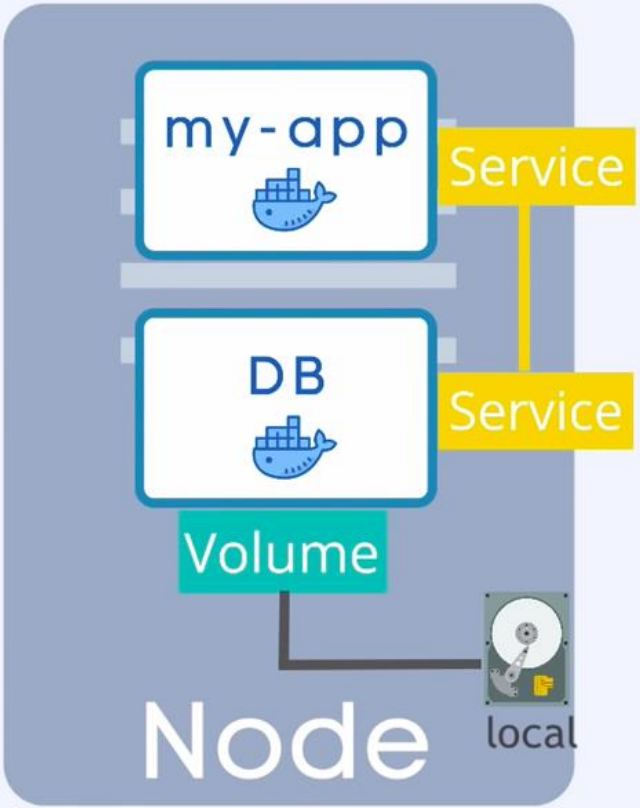
💡 Use it as environment variables or as a properties file


my-app
Service
DB
Service
Node
Data Storage

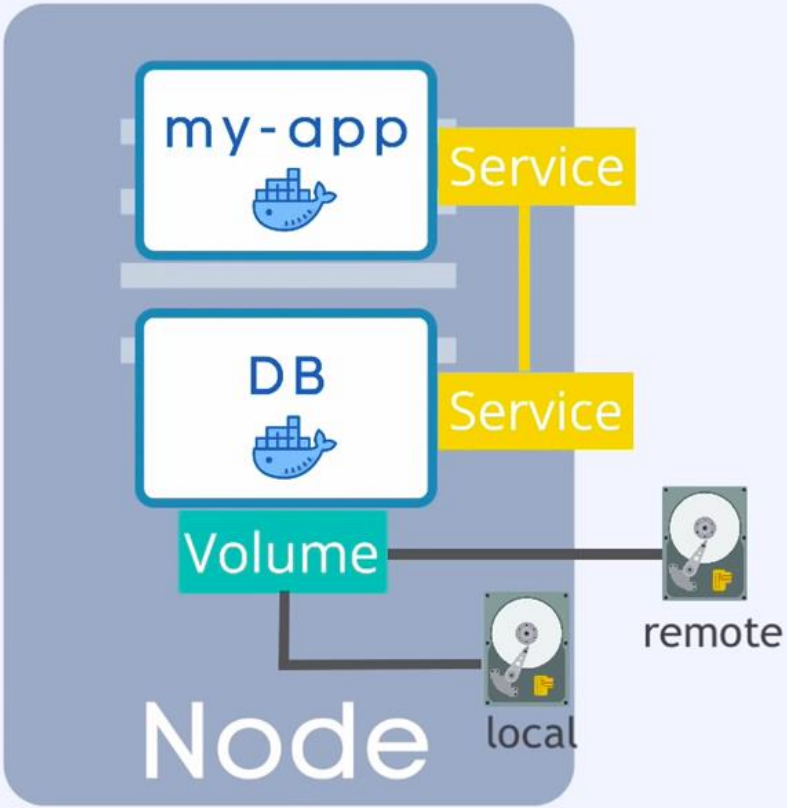

my-app
Service
DB
Service
DATA
Node


my-app
Service
DB
Service
Volume
Node

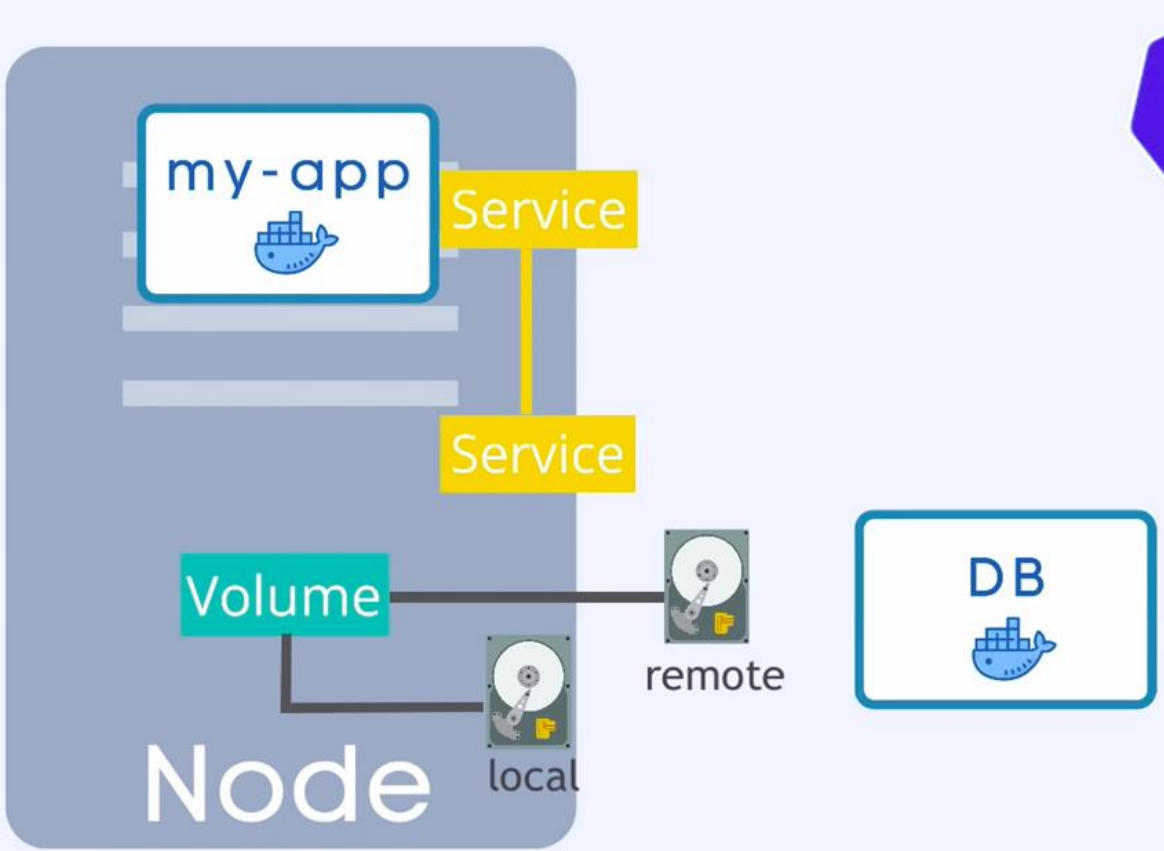

vol

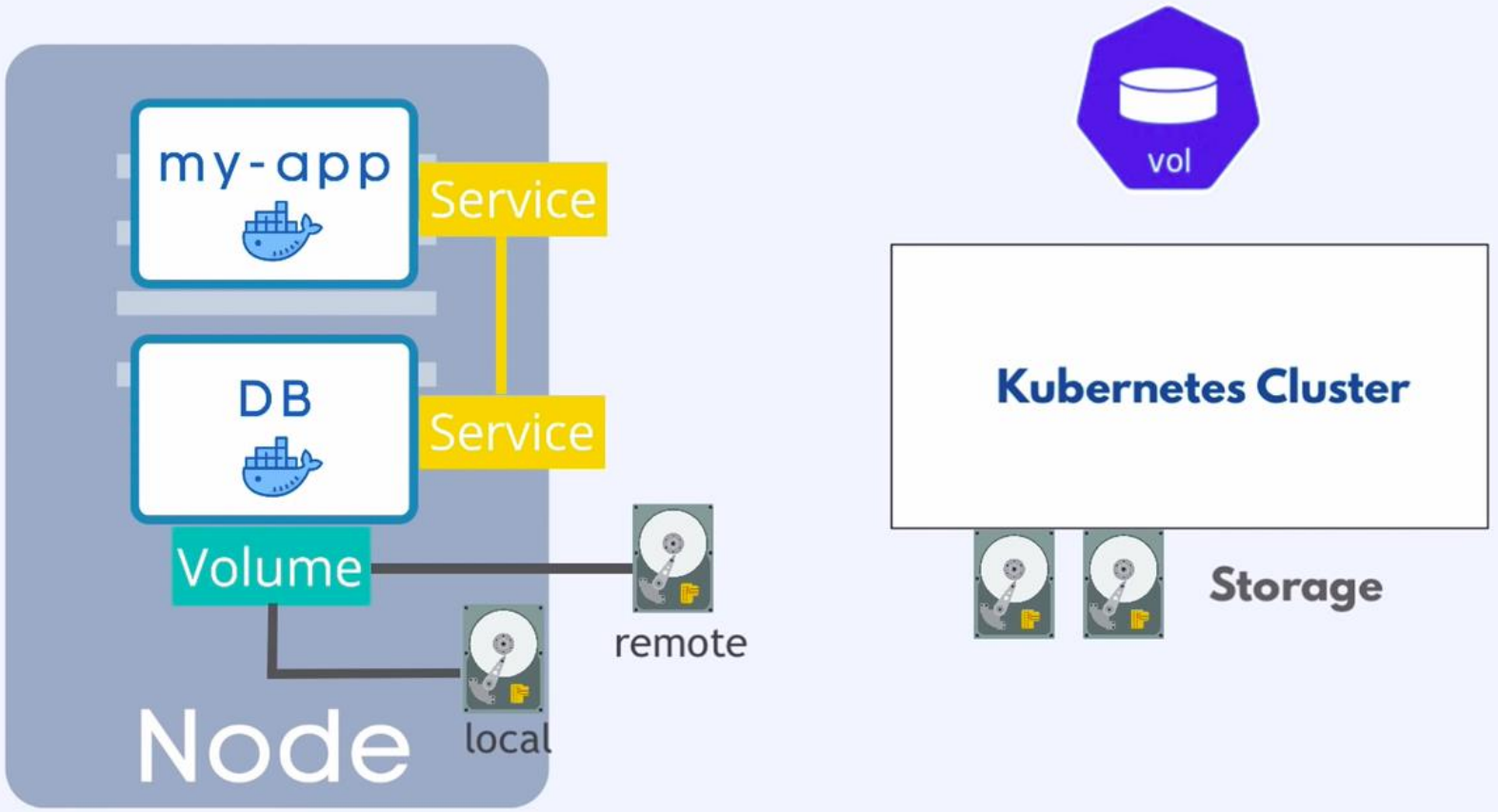


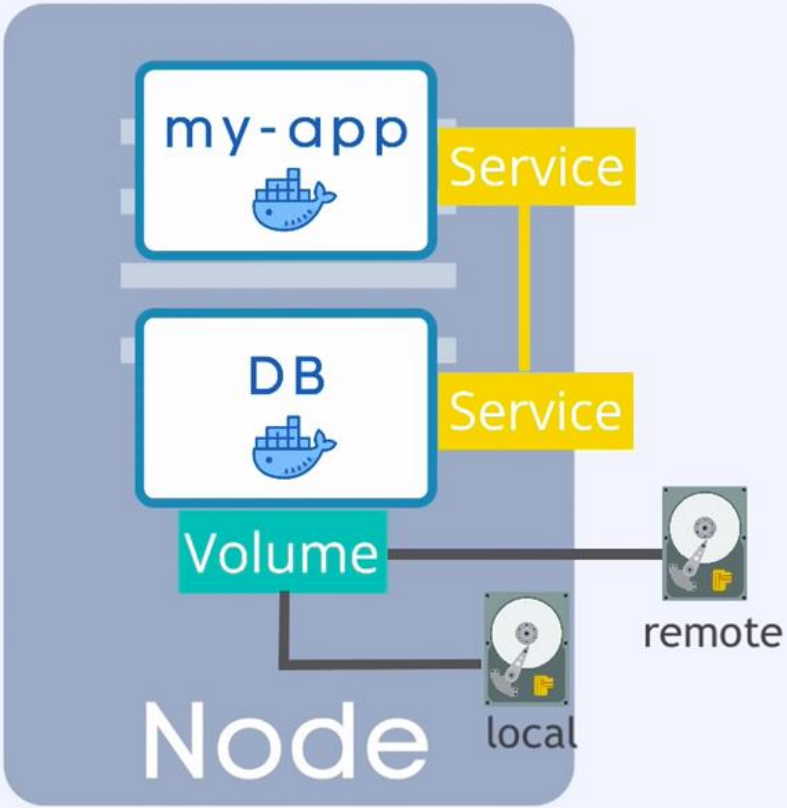
- Storage on **local** machine



- Storage on **local** machine
- Or **remote,** outside of the K8s cluster


my-app
Service
Service
Volume
remote
local
Node
DB
vol

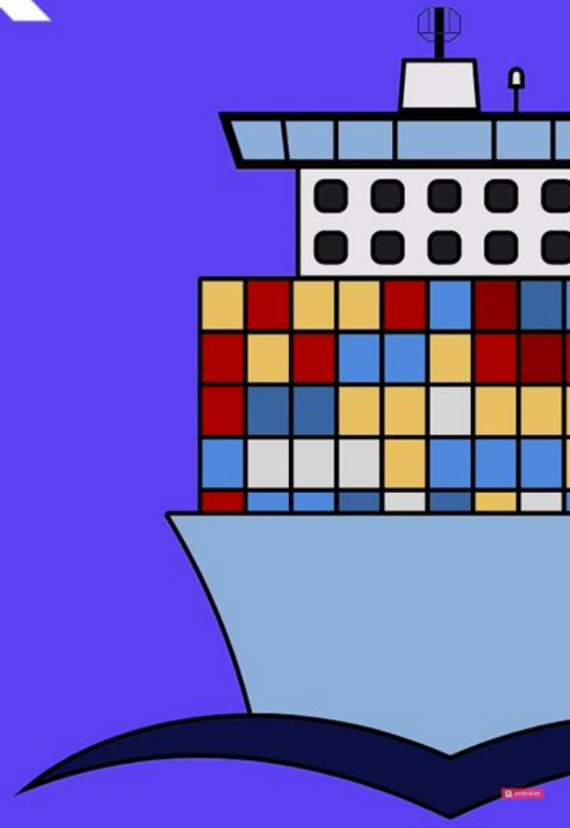

my-app
Service
DB
Service
Volume
remote
local
Node
vol
Kubernetes Cluster
Storage

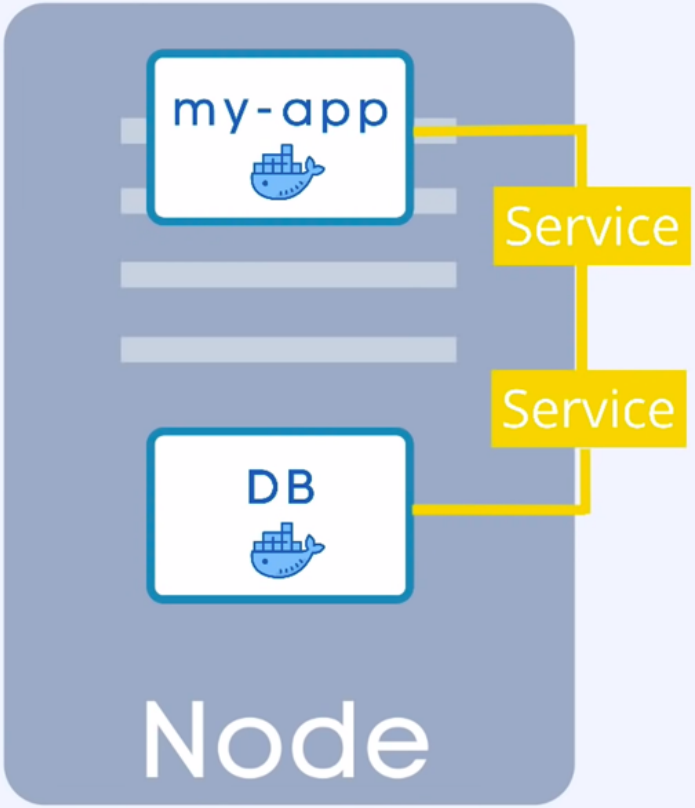


❗ Kubernetes doesn't manage data persistance!

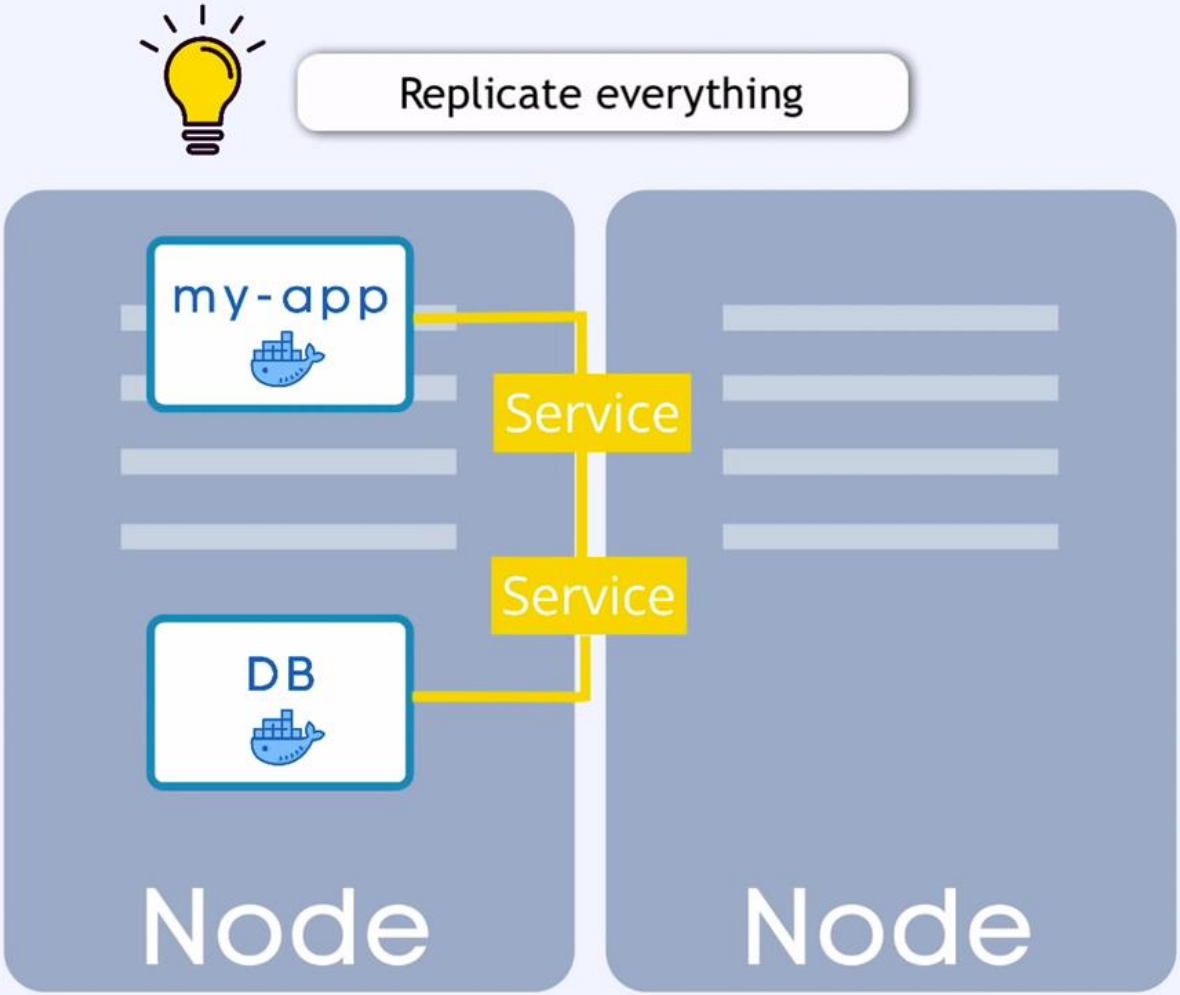
Deployment &

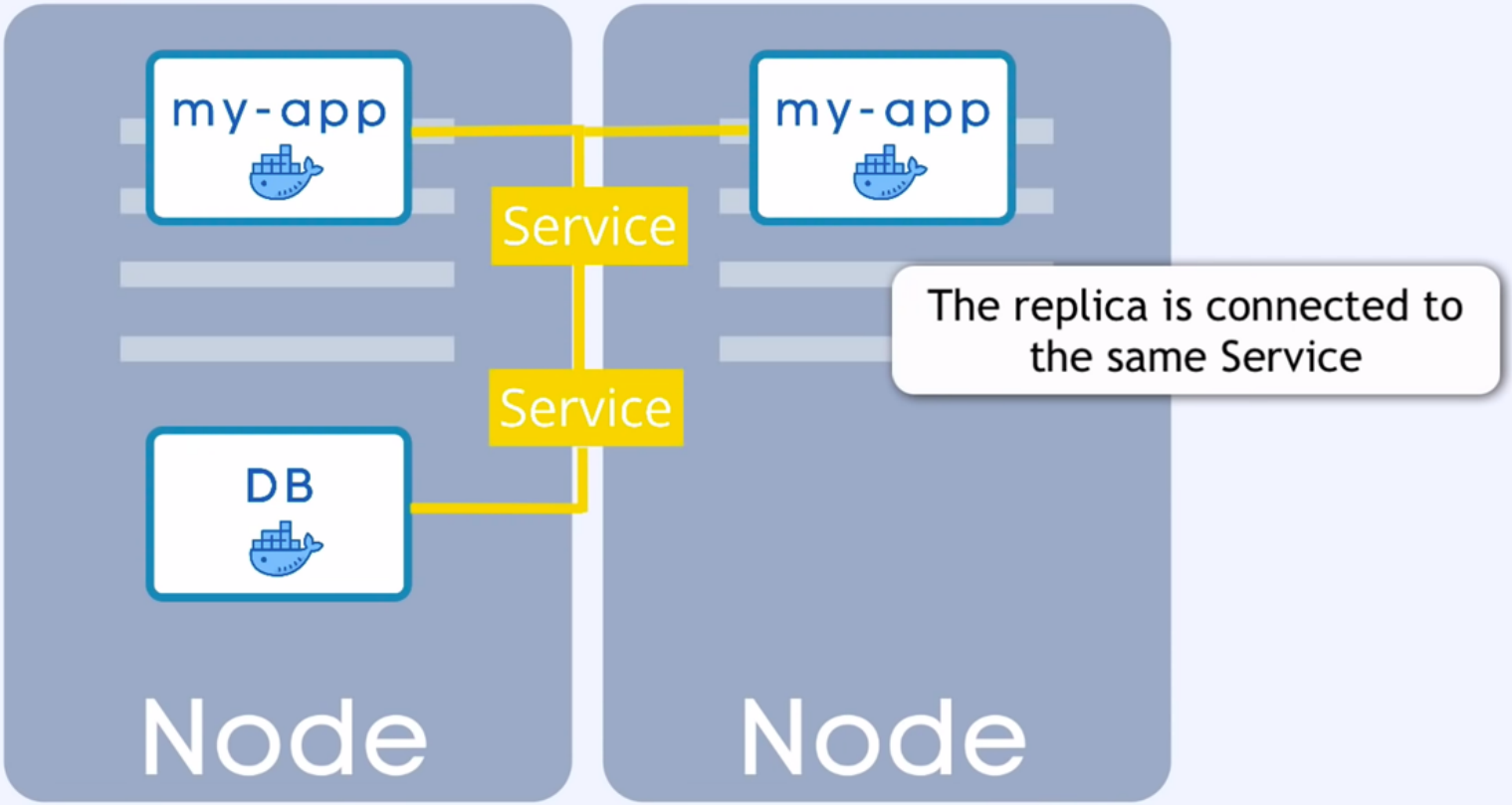
StatefulSet

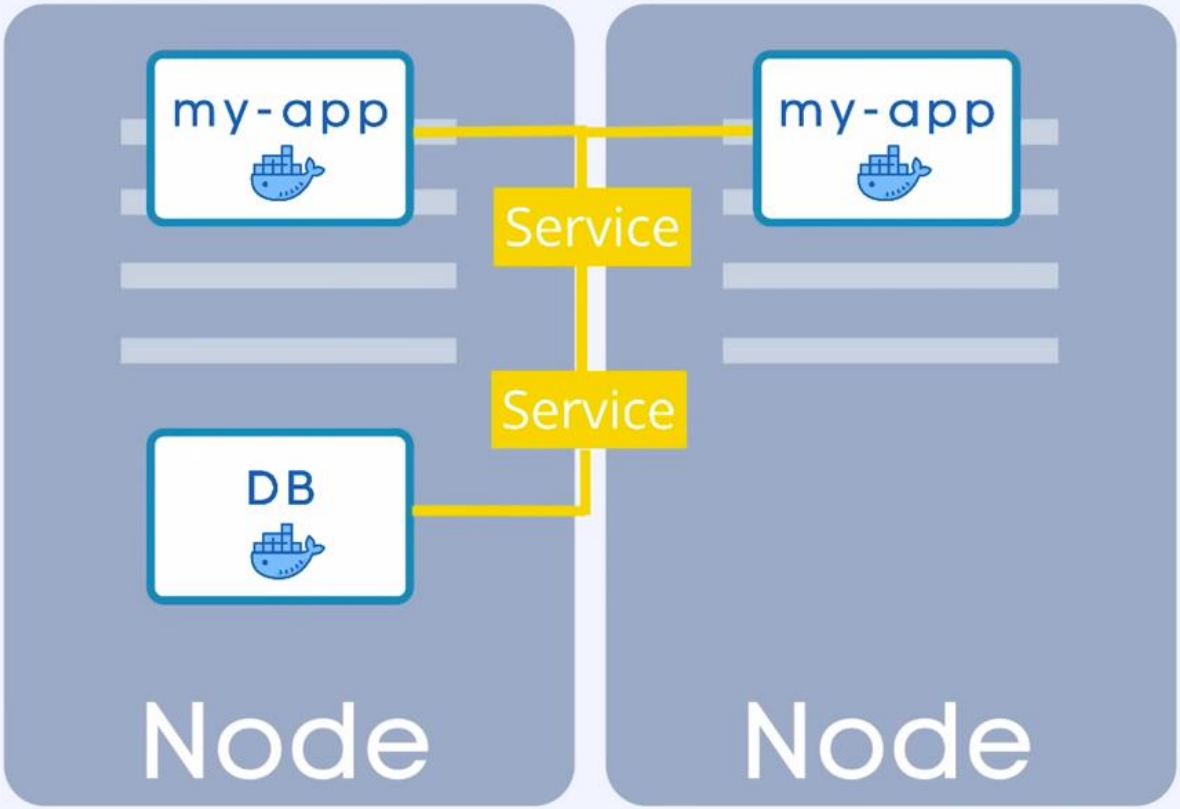




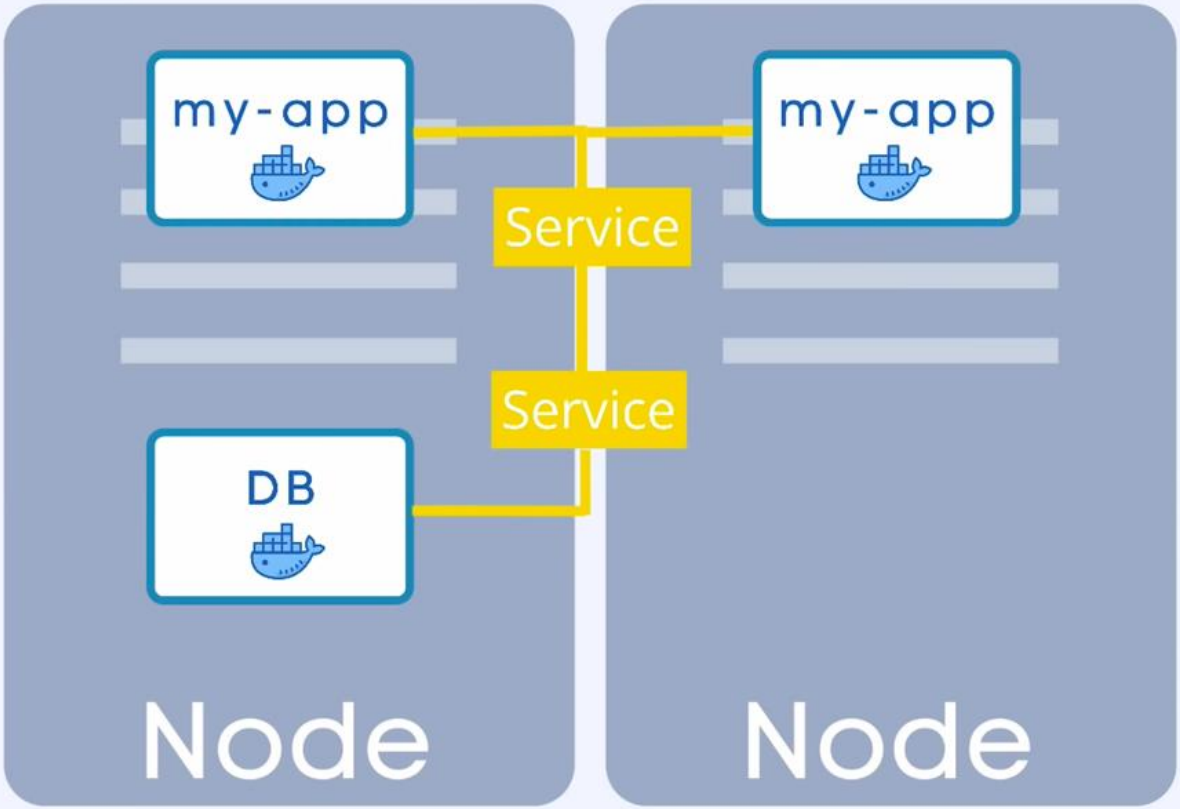
Distributed Systems


Replicate everything
my-app
DB
Service
Service
Node
Node

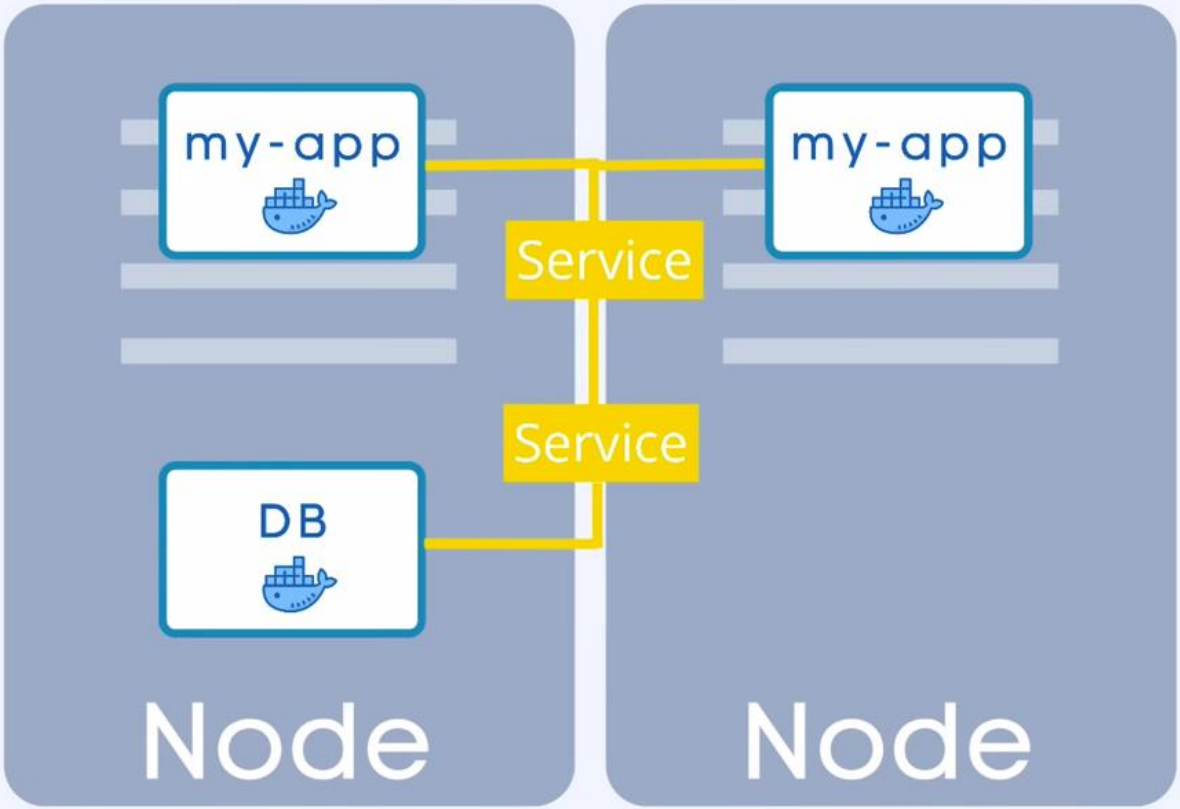

my-app
my-app
Service
Service
DB
Node
Node
The replica is connected to the same Service



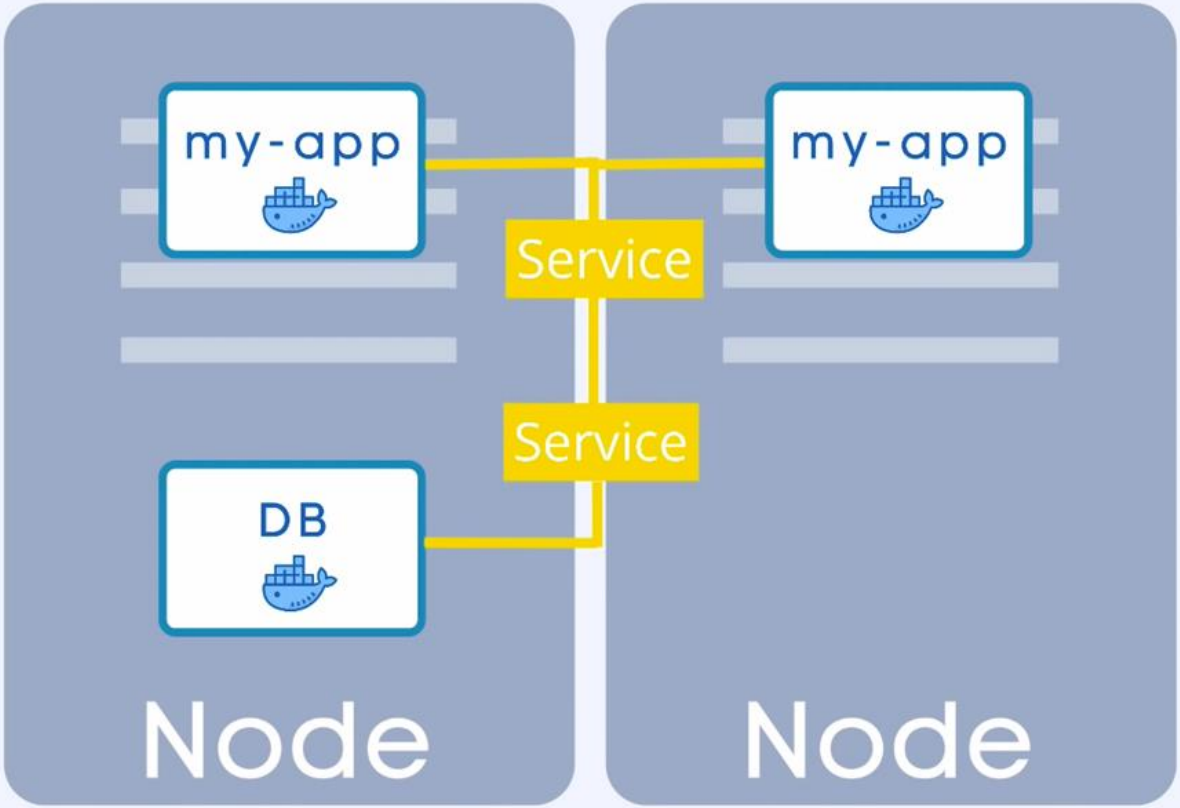
- permanent IP



- permanent IP
- load balancer

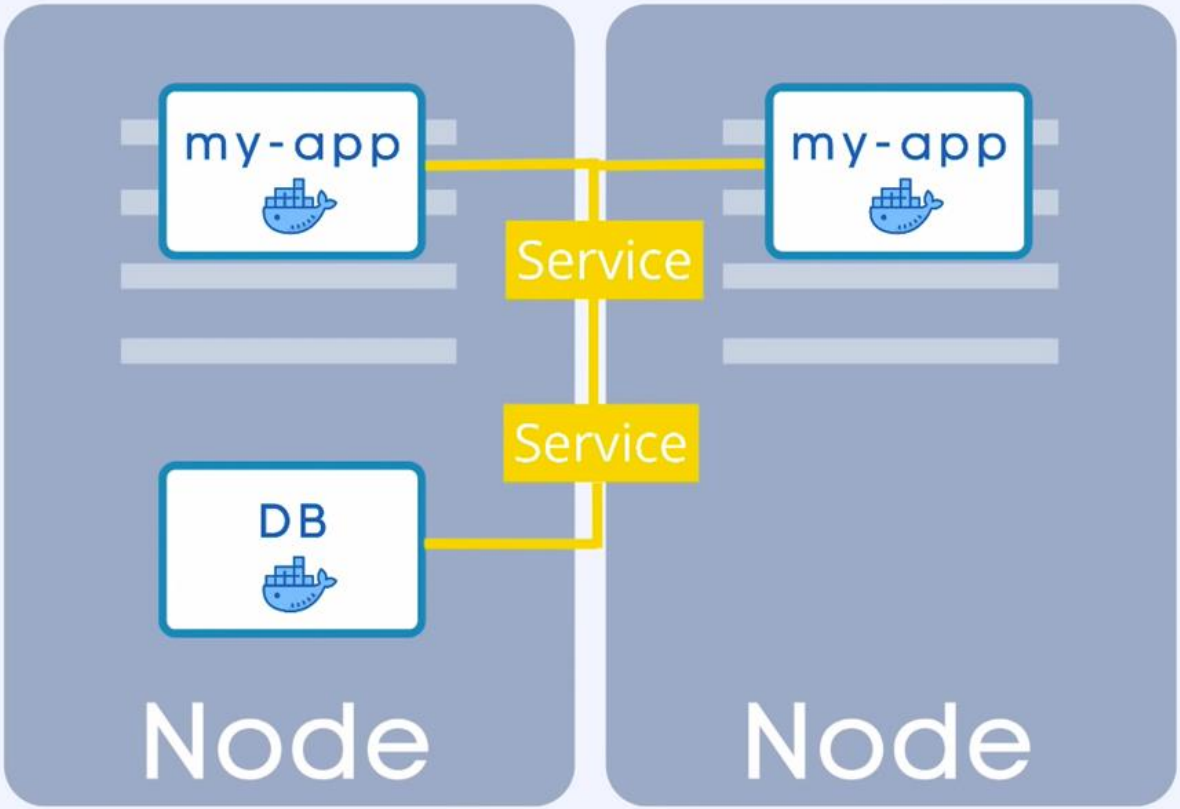


Define **blueprint** for Pods



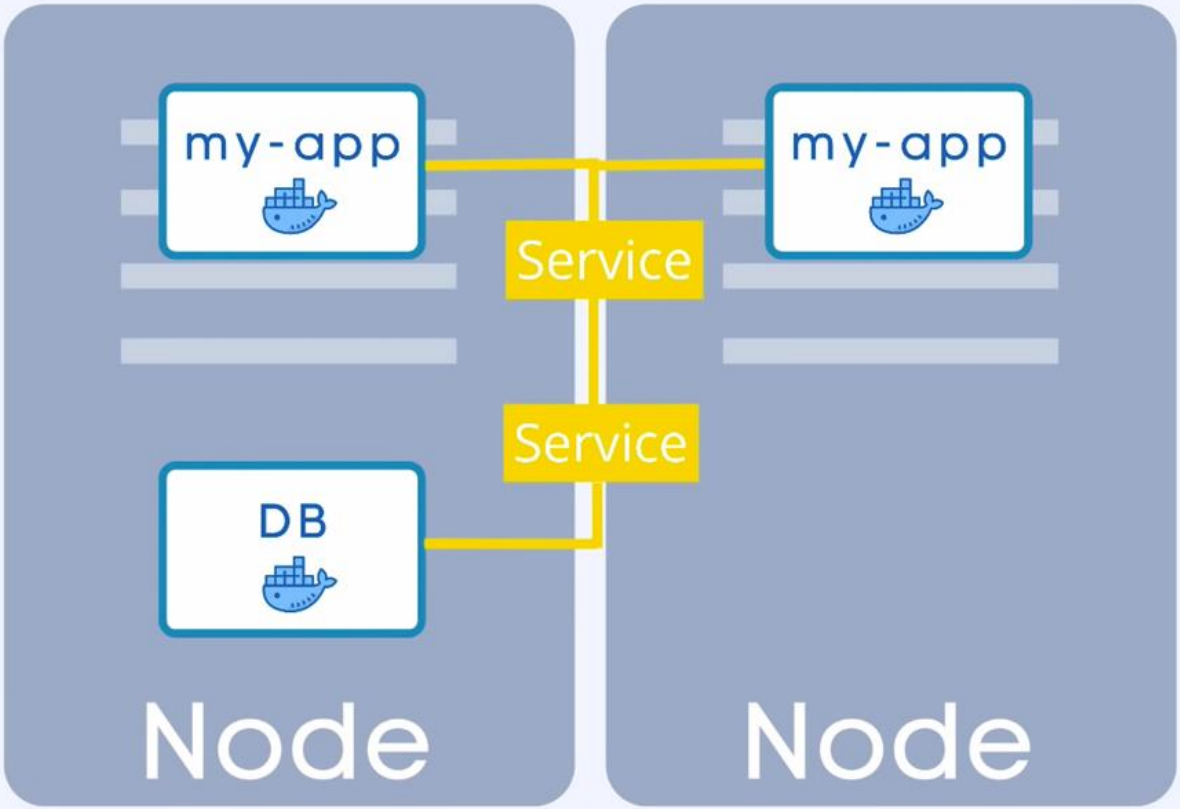
Define **blueprint** for Pods

- Specify how many replicas you want to have



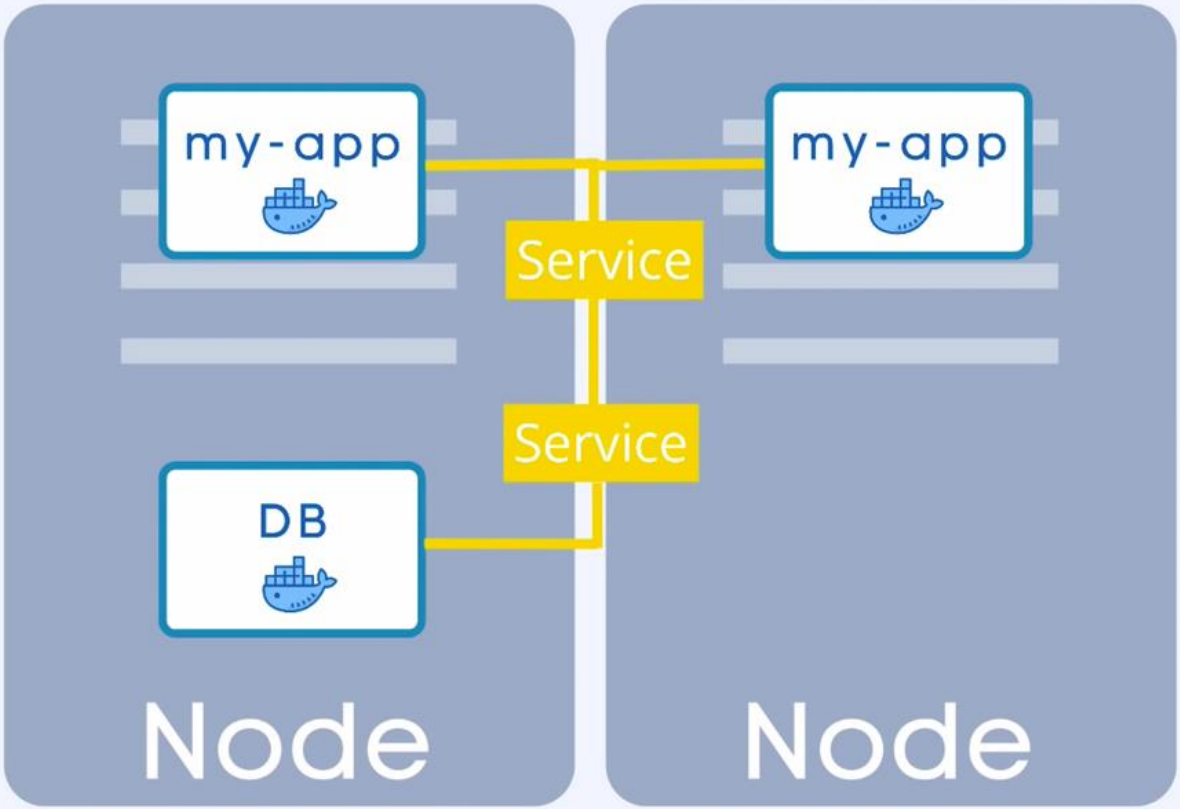
DEPLOYMENT

- **Blueprint** for "my-app" Pods



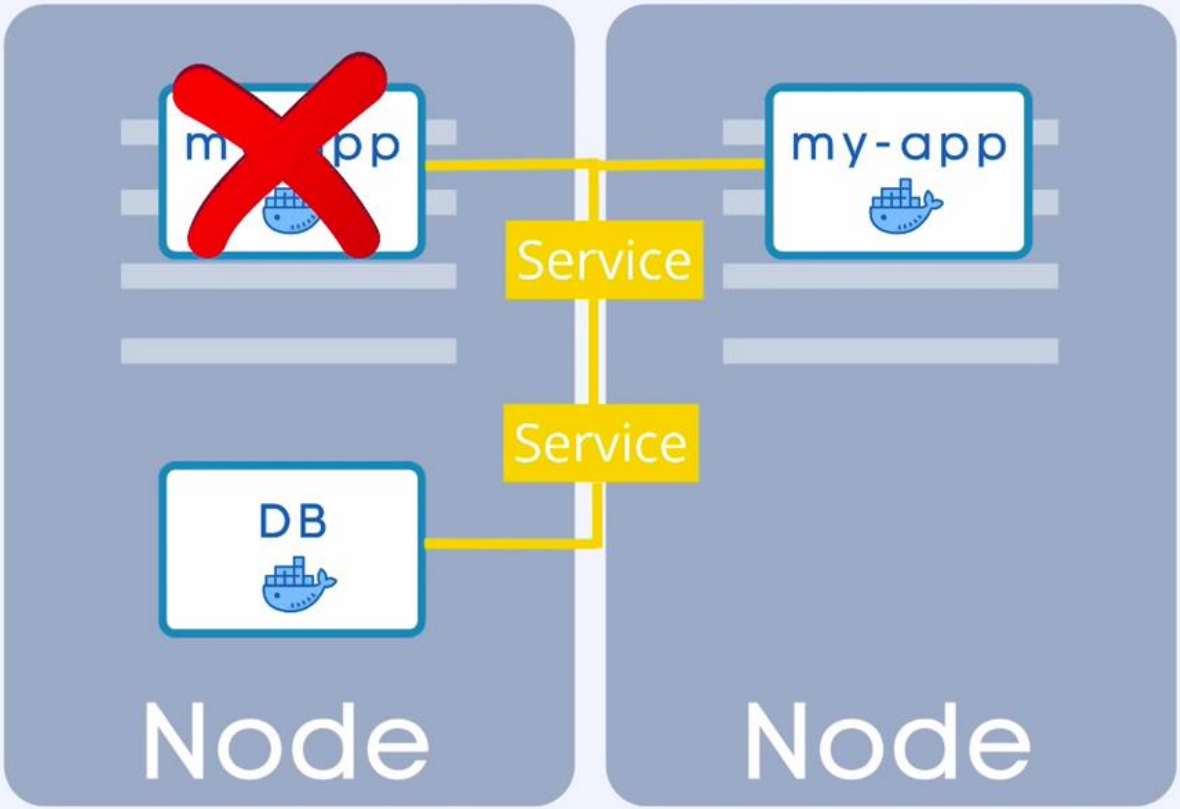
DEPLOYMENT

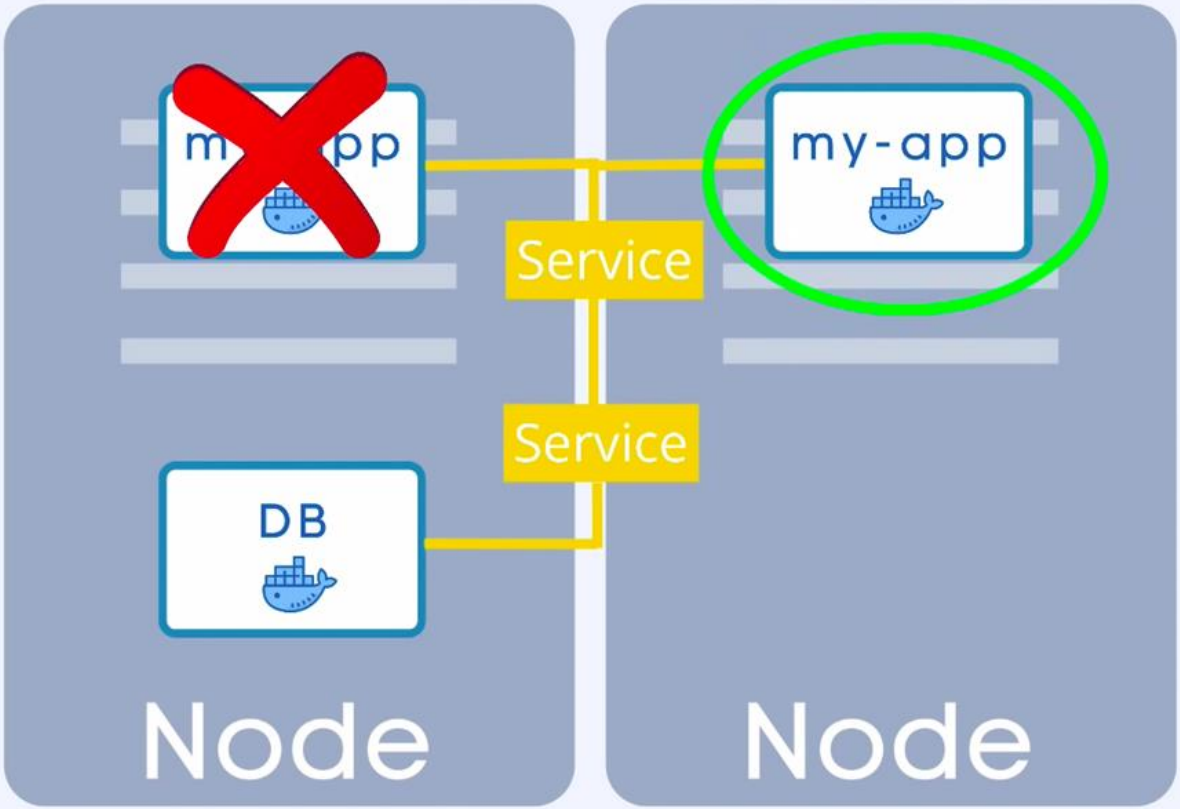
- **Blueprint** for "my-app" Pods
- You create **Deployments**

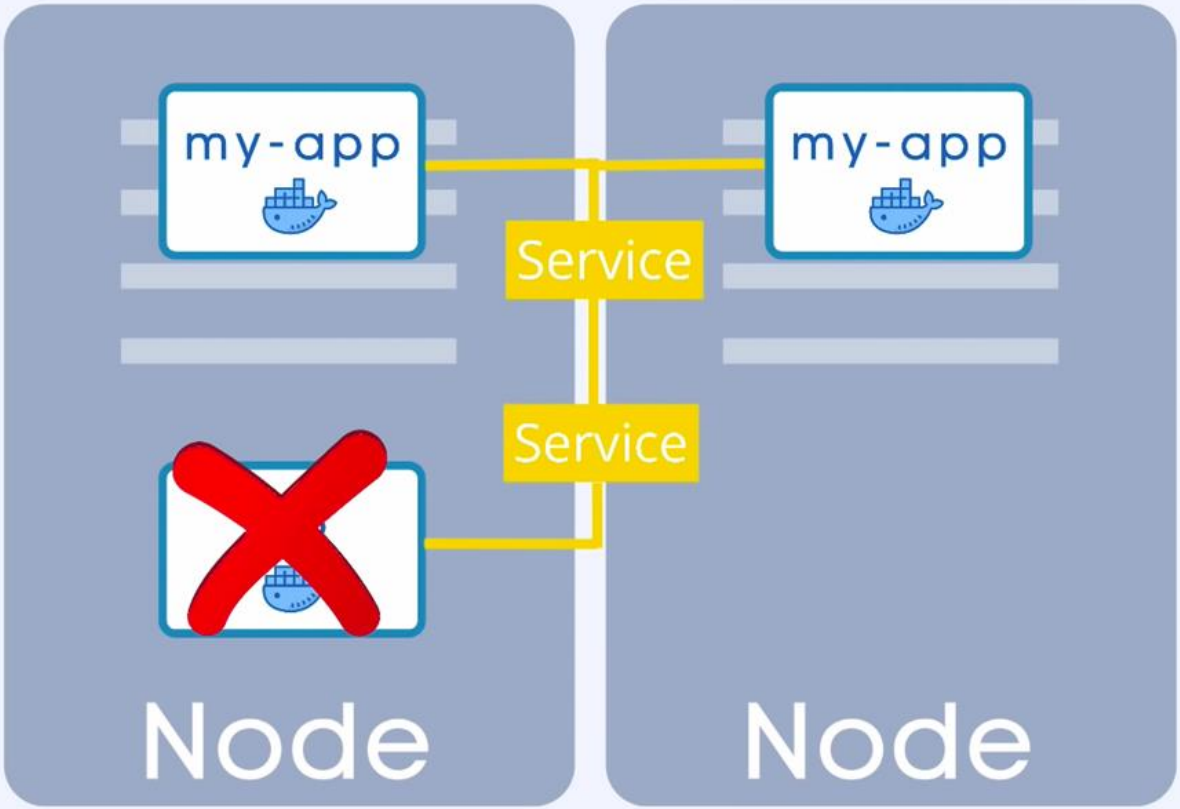


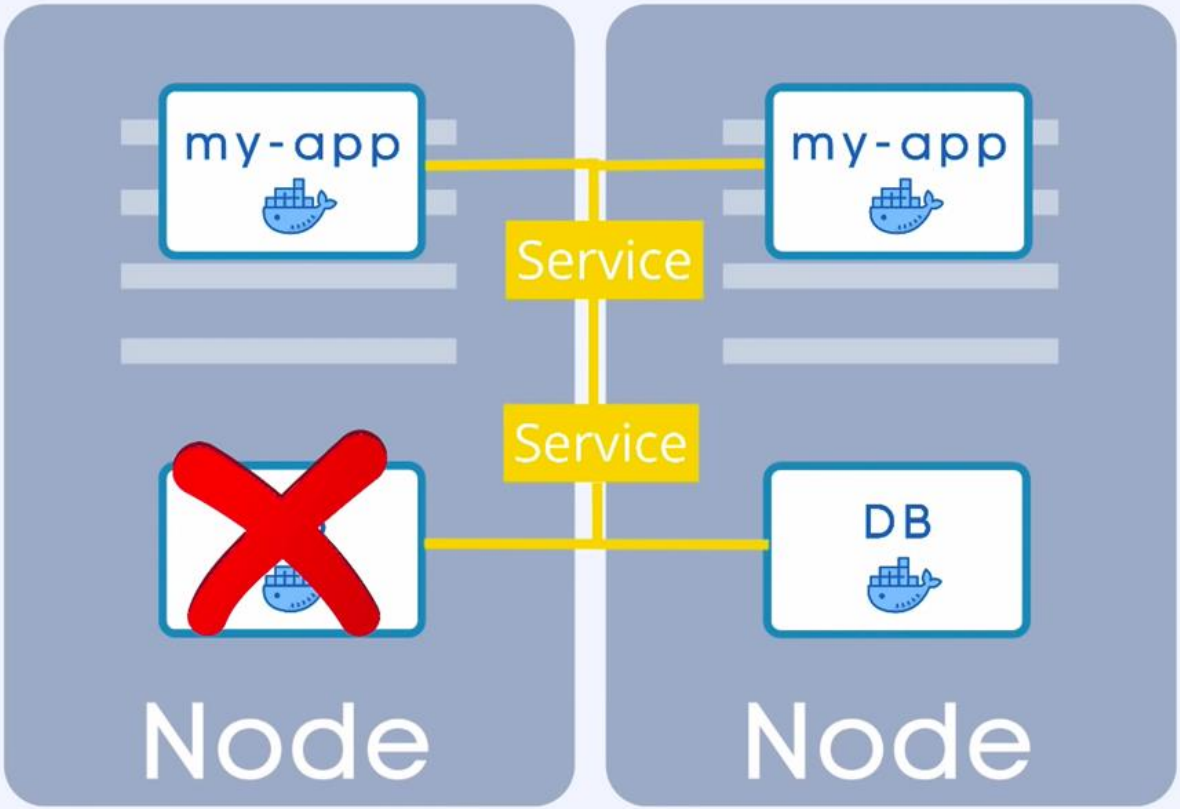
DEPLOYMENT

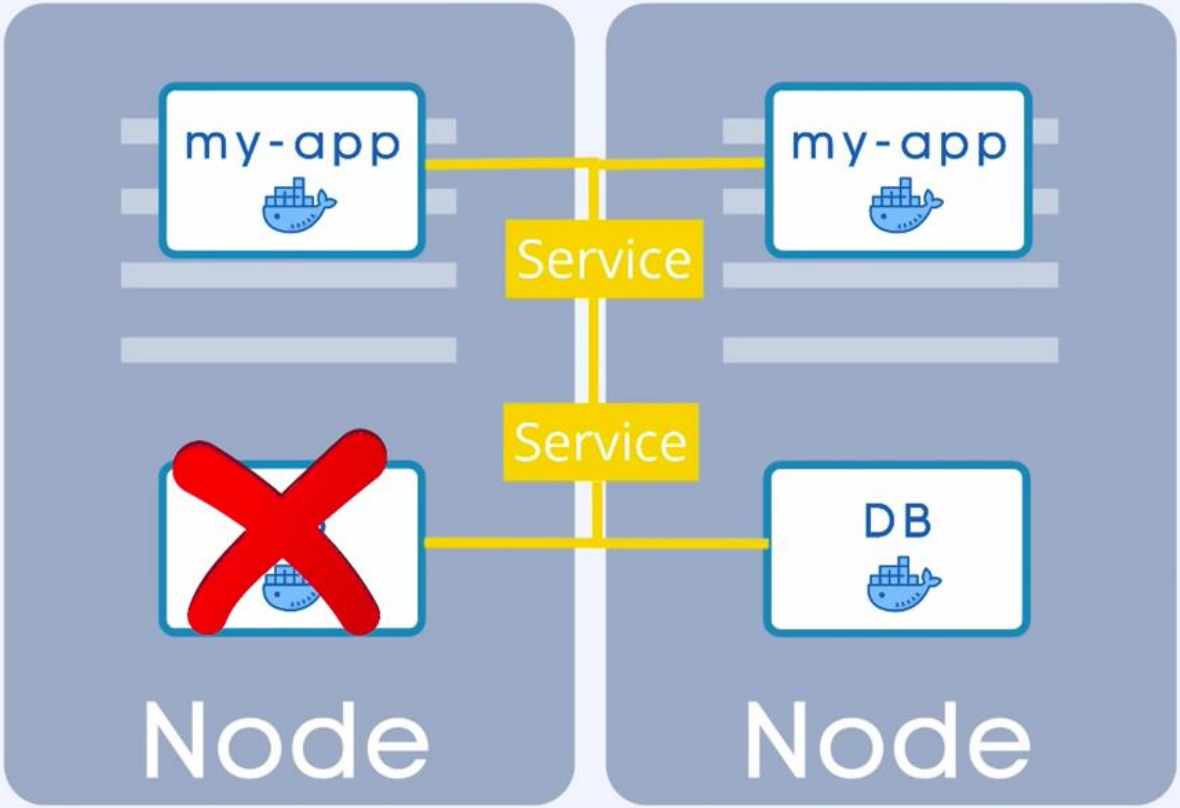
- **Blueprint** for "my-app" Pods
- You create **Deployments**
- Abstraction of Pods


my-app
Service
Service
DB
Node
Node

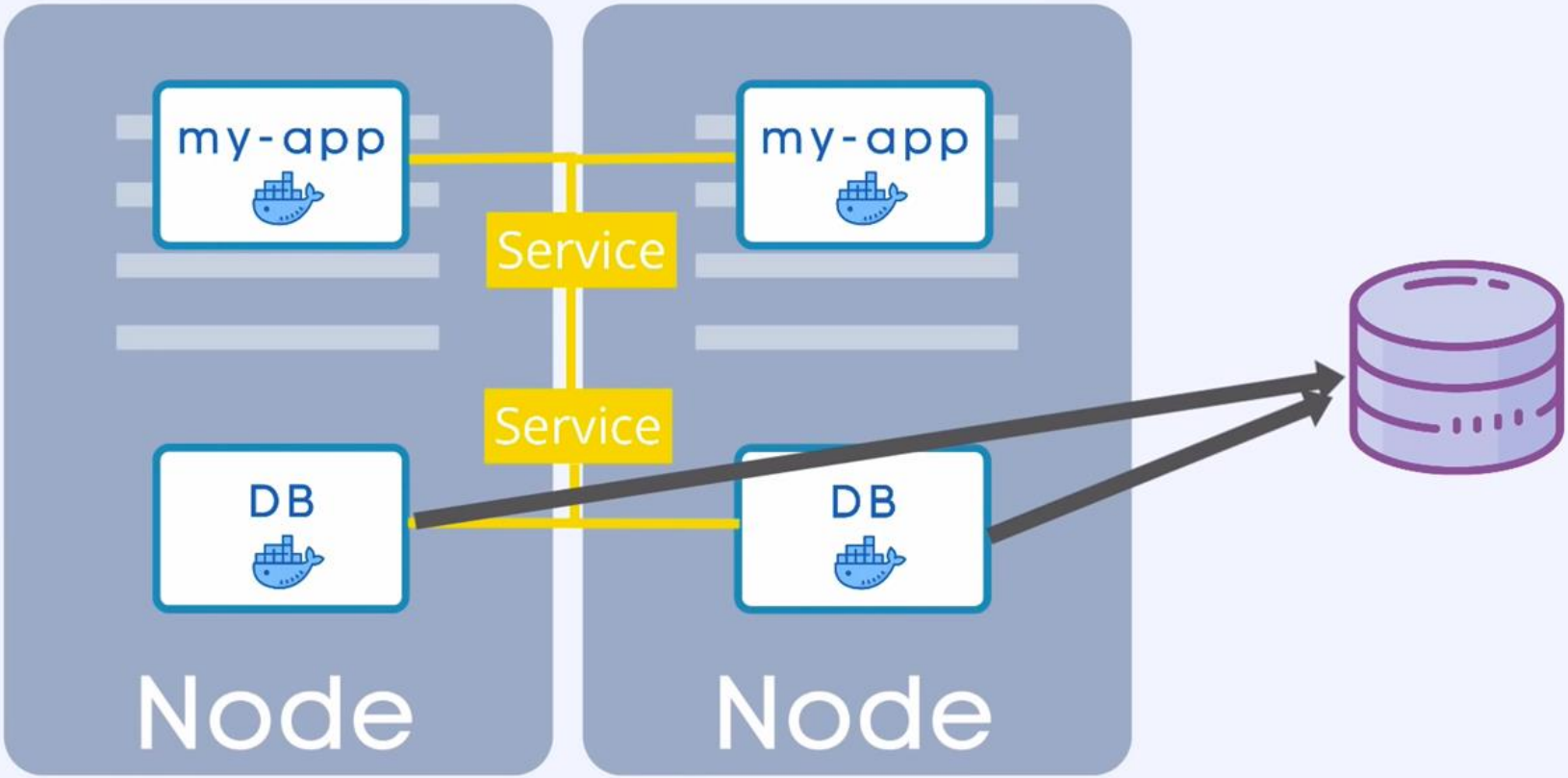

my-app
Service
Service
DB
Node
Node

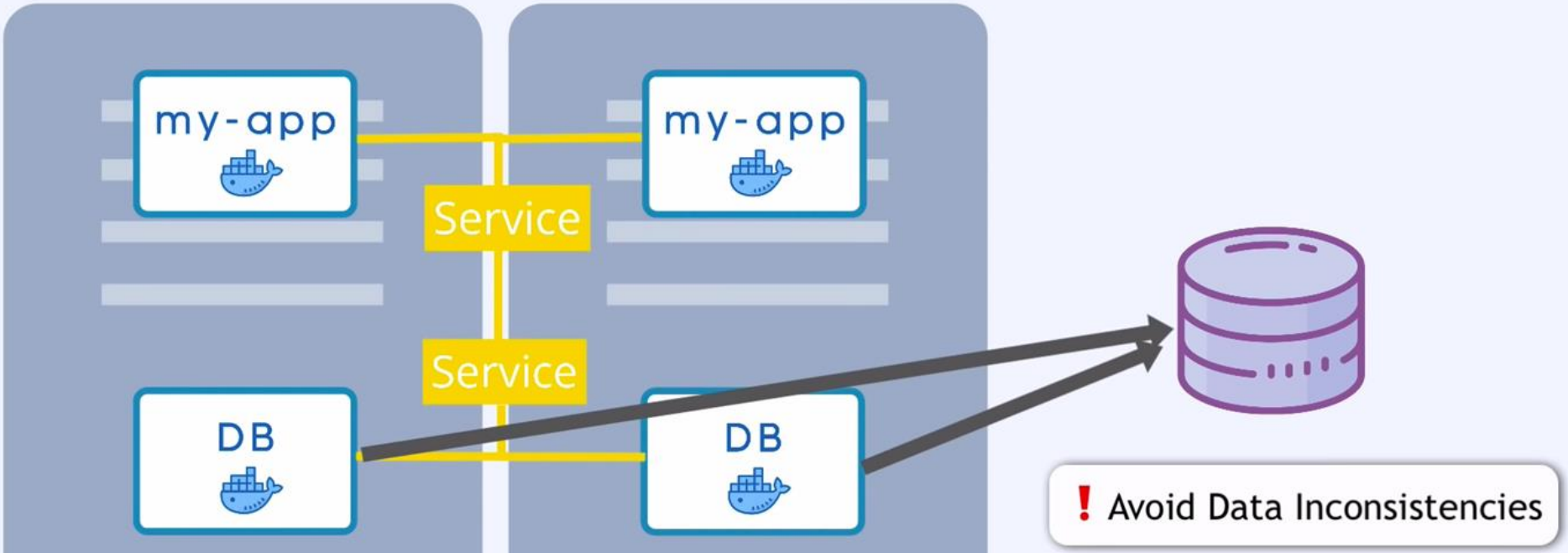

my-app
my-app
Service
Service
Node
Node

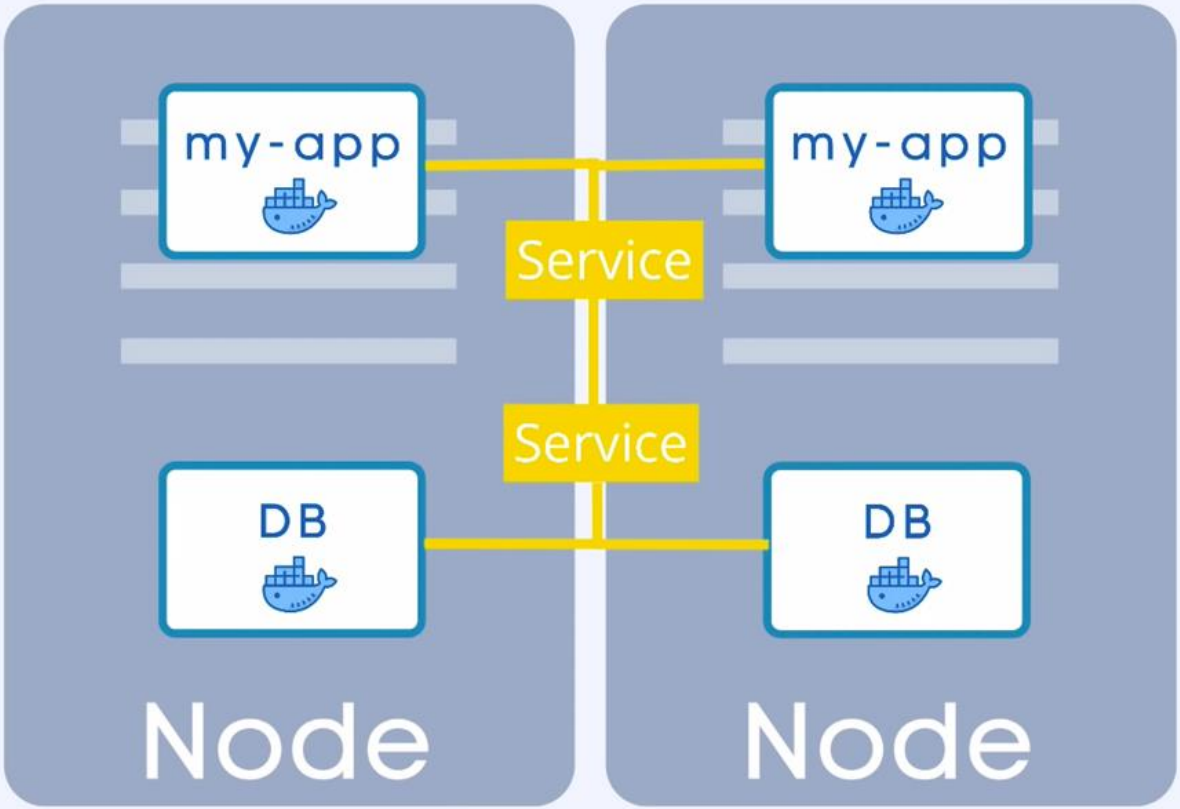

my-app
my-app
Service
Service
DB
Node
Node



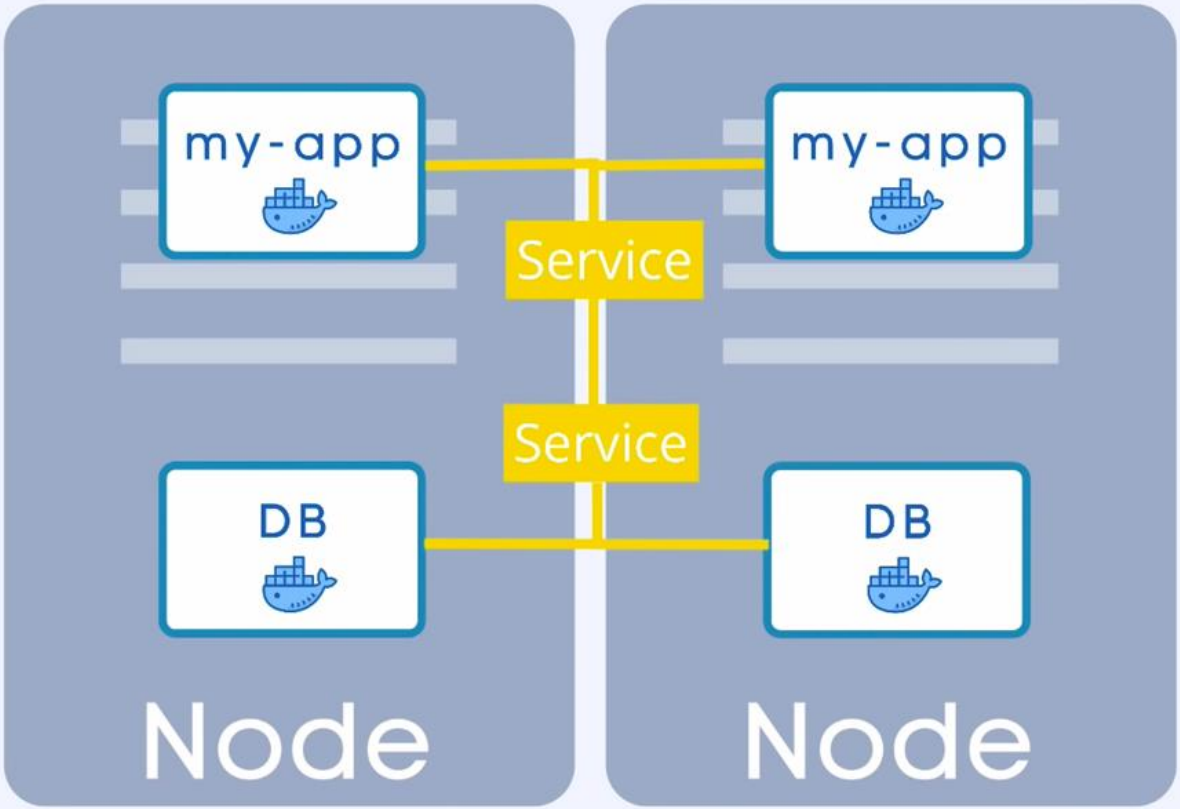
❗ DB can't be replicated via Deployment!


my-app
my-app
Service
Service
DB
DB
Node
Node


my-app
my-app
Service
Service
DB
DB
Node
Node
! Avoid Data Inconsistencies

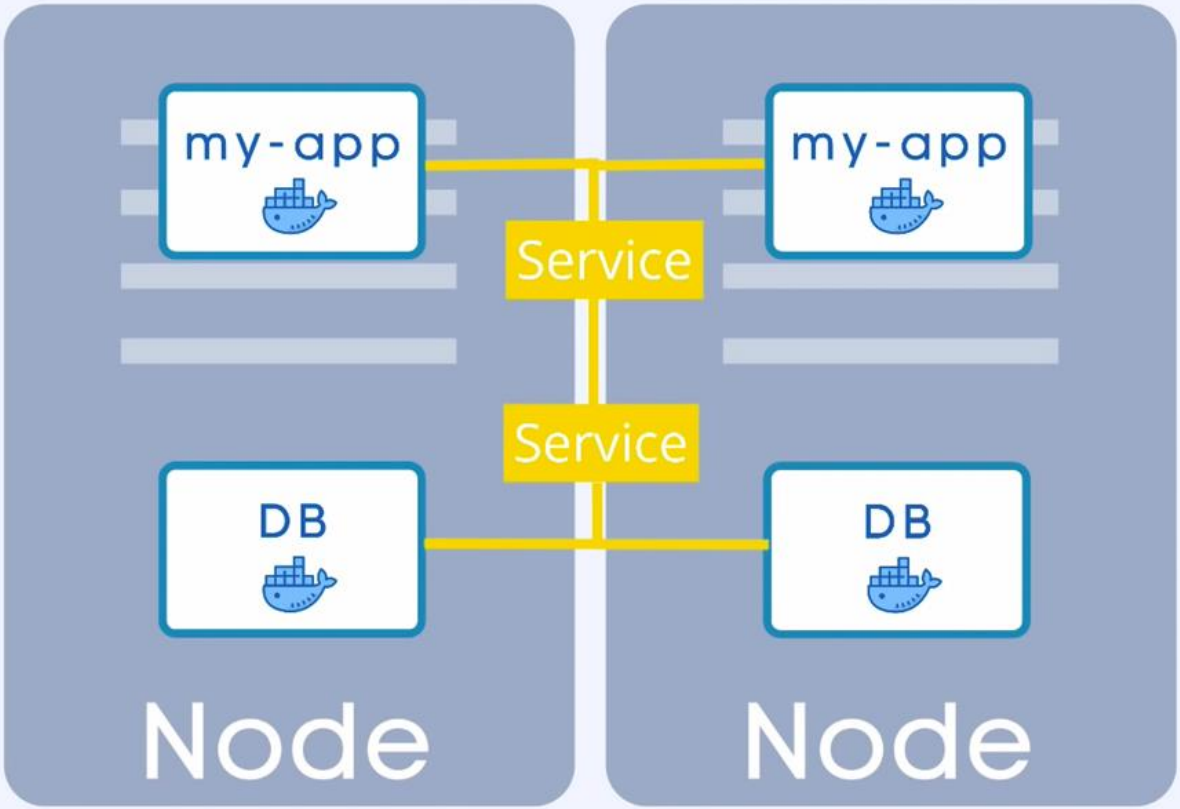


STATEFULSET



STATEFULSET

- for STATEFUL apps



STATEFULSET

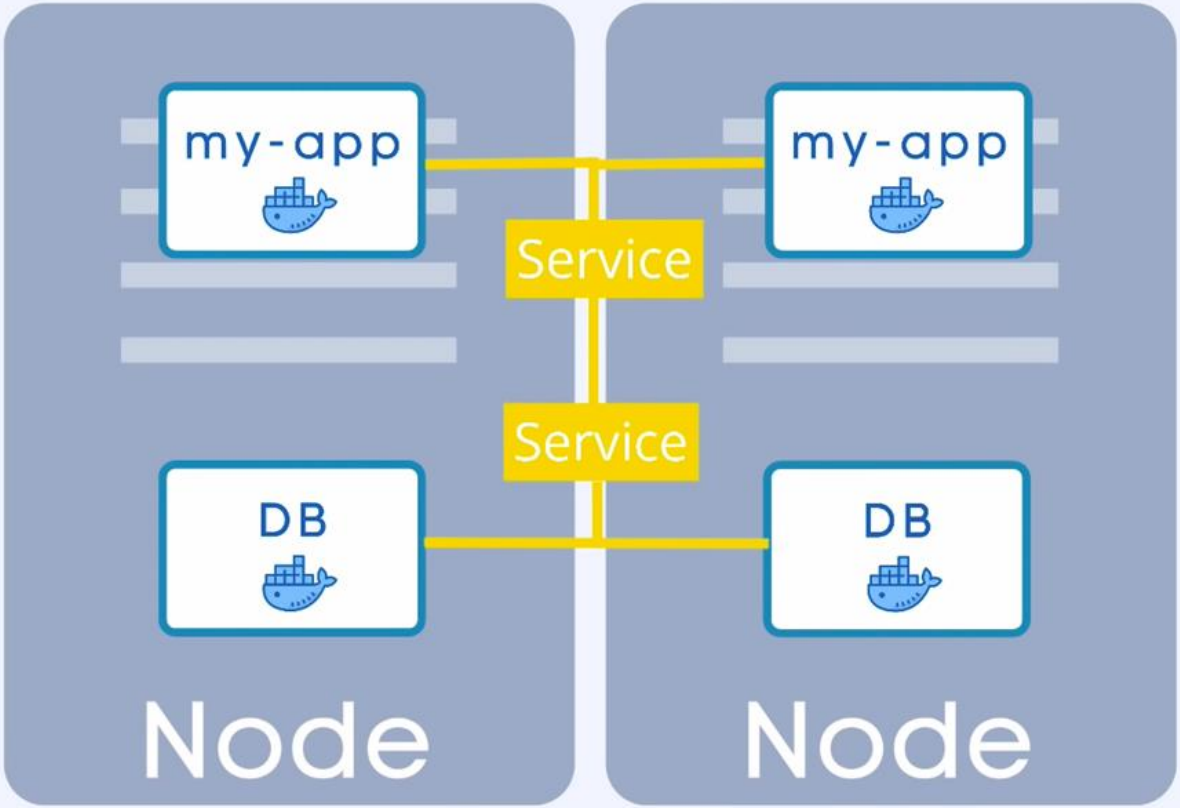
sts

- for STATEFUL apps

MySQL

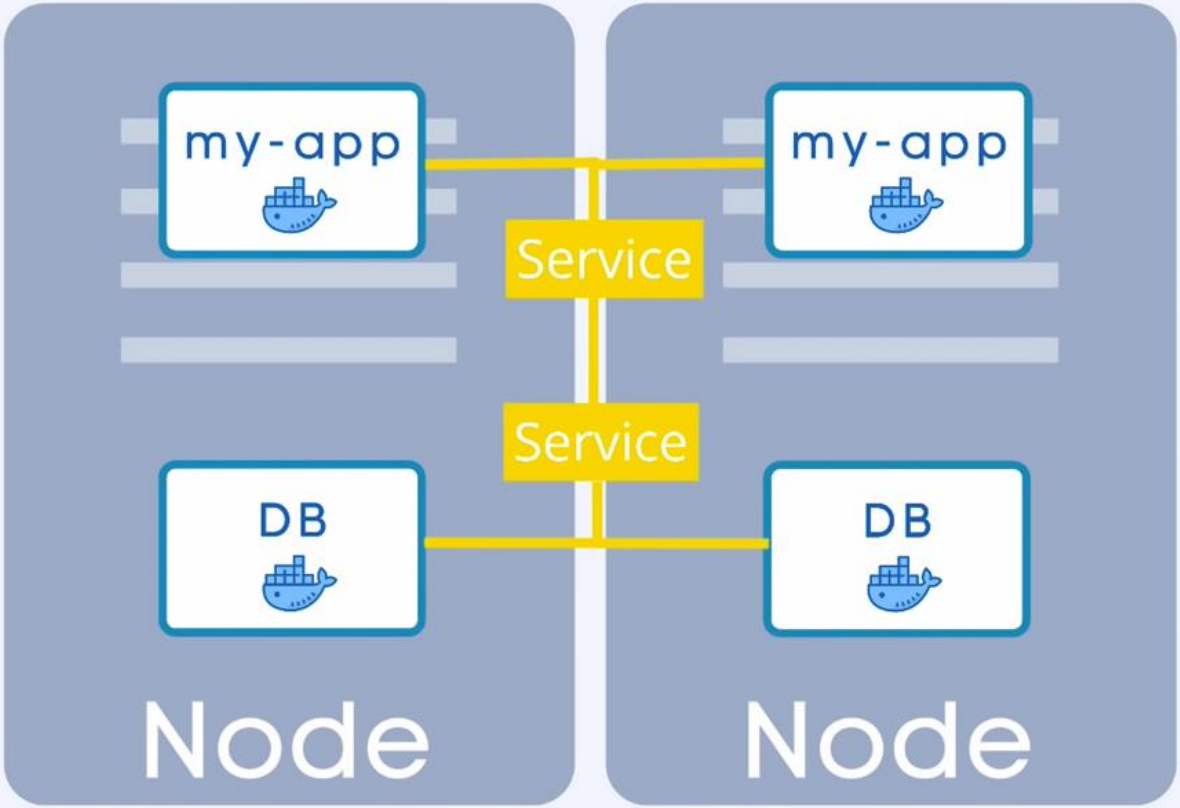
elastic

mongoDB

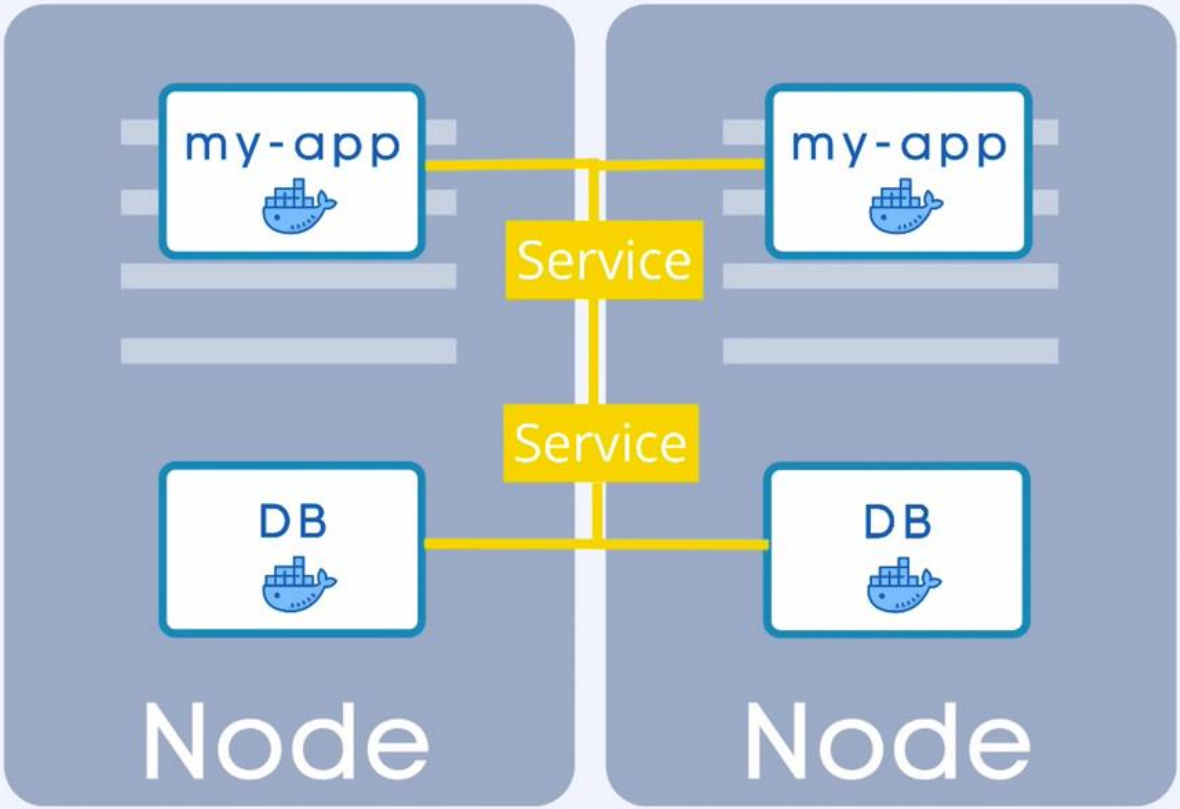


Deployment =
for stateLESS Apps

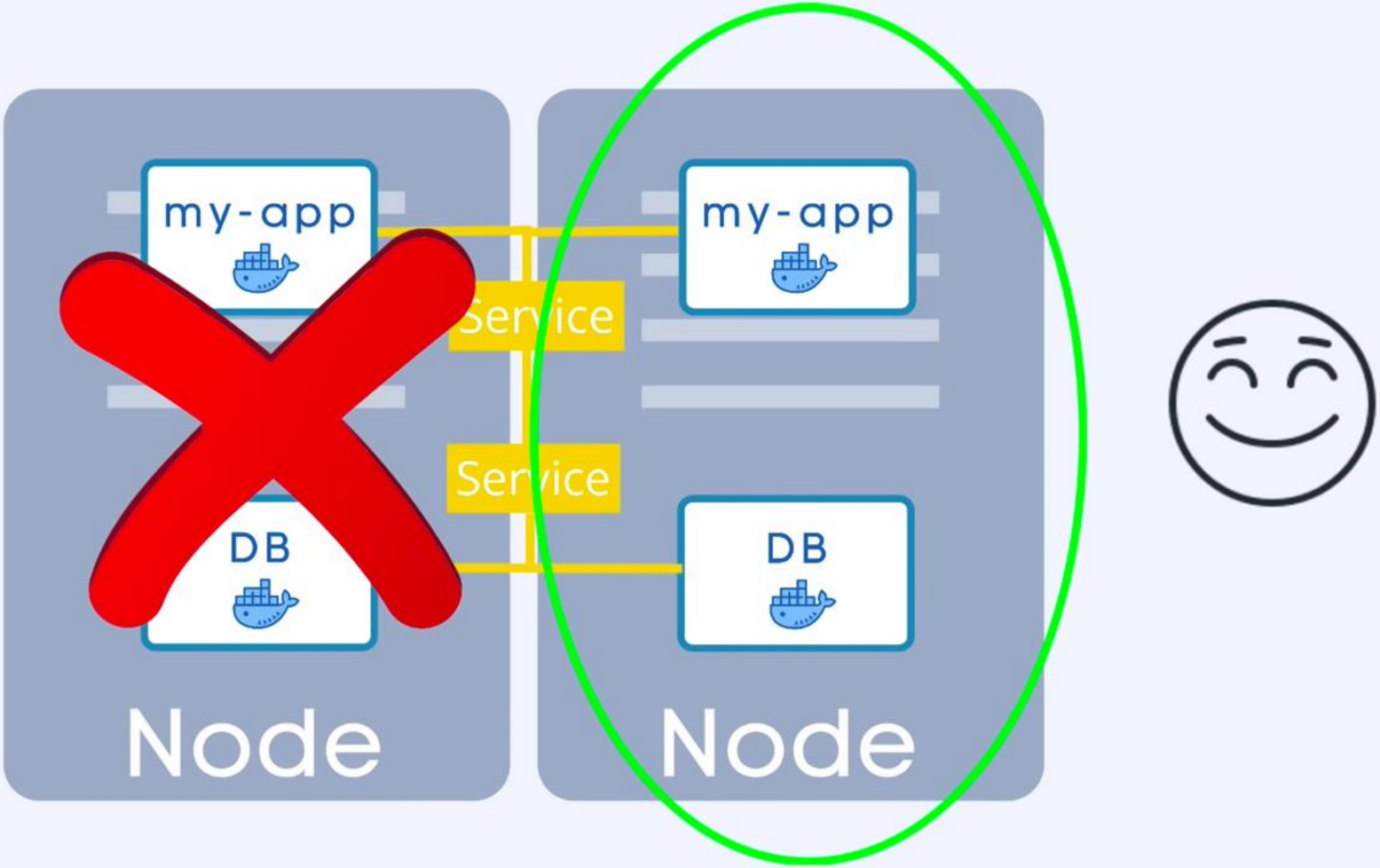
StatefulSet =
for stateFUL Apps or Databases



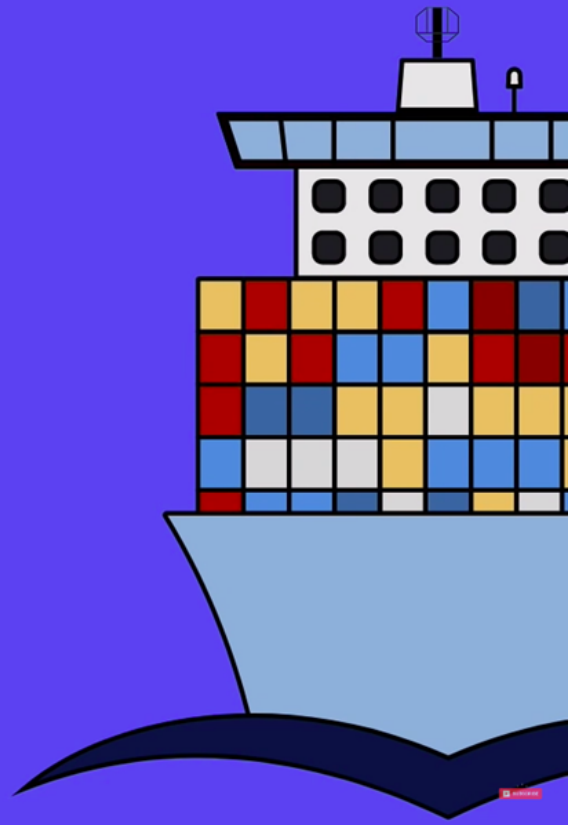
Deploying StatefulSet
not easy



DB are often hosted outside of Kubernetes cluster


my-app
DB
Service
Service
my-app
DB
Node
Node

Wrap Up



Main Kubernetes Components

summarized

Pod — ⭐ abstraction of containers

Service — ⭐ communication

Ingress — ⭐ route traffic into cluster

Main Kubernetes Components

summarized

ConfigMap

Secret

⭐ external configuration

Main Kubernetes Components

summarized

Volume

⭐ data persistence

Main Kubernetes Components

summarized

Deployment

StatefulSet

⭐ replication