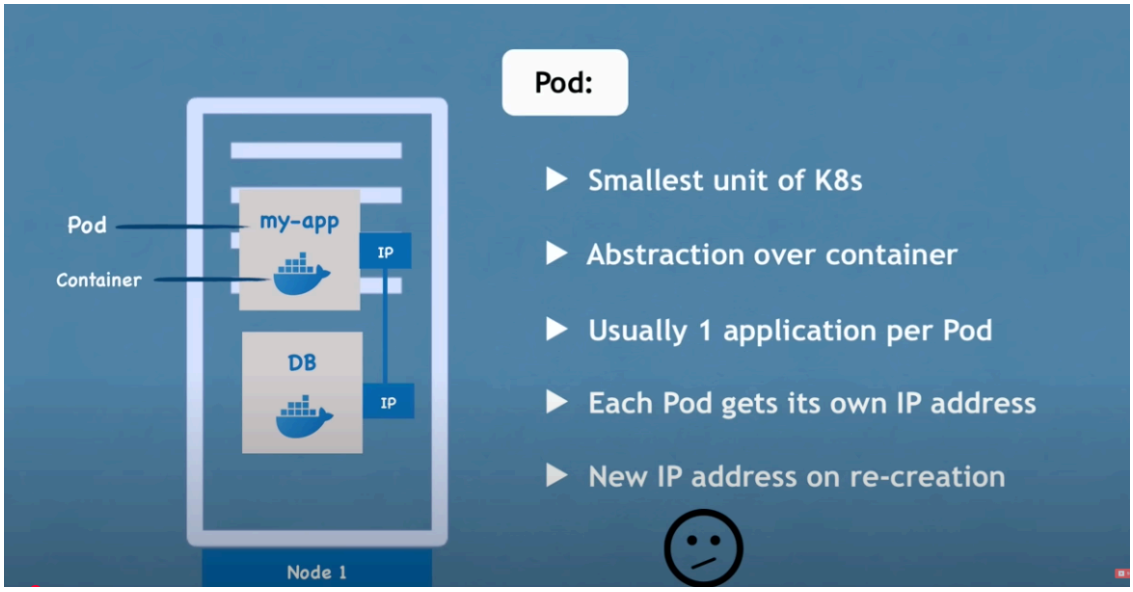
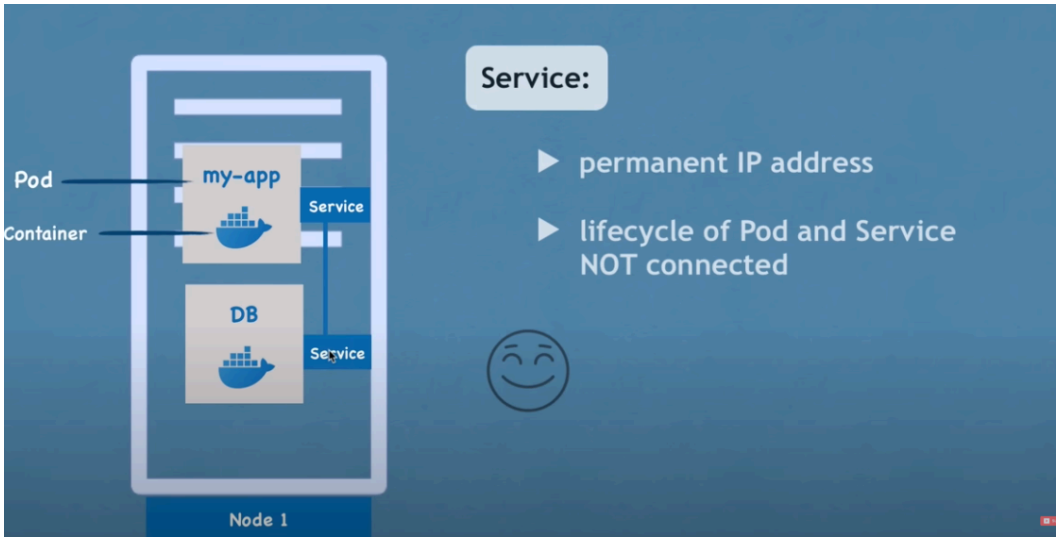
Kubernetes



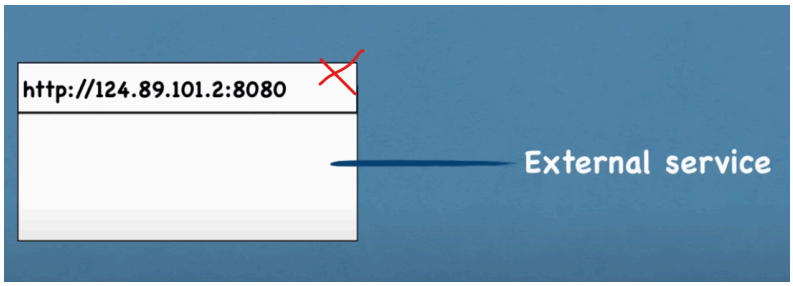
Why do we need services

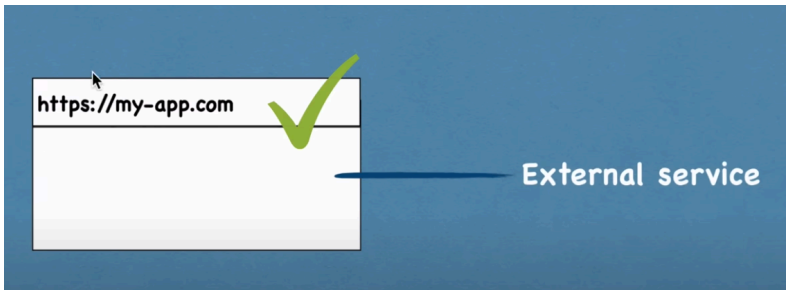
If the pods communicate with ip addresses when a pod dies then it could be a problem, because each creation we will have a new ip address. So why we need services

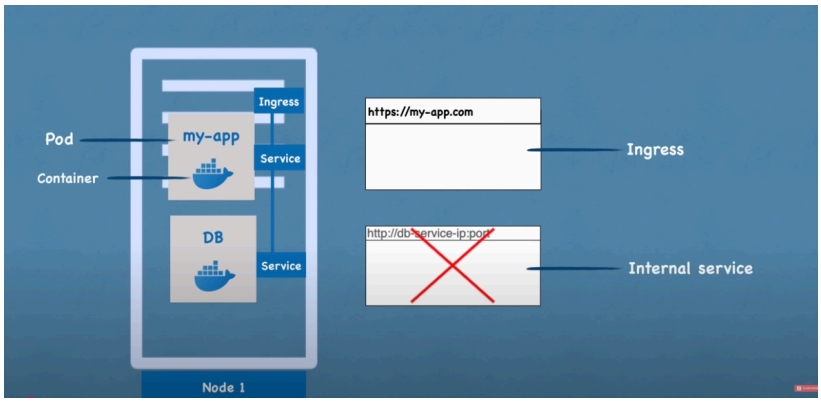


Why do we need ingress:

You need **Ingress** in Kubernetes for **external HTTP(S) access** to services within your cluster. Ingress is an API object that manages external access to services, typically HTTP traffic.



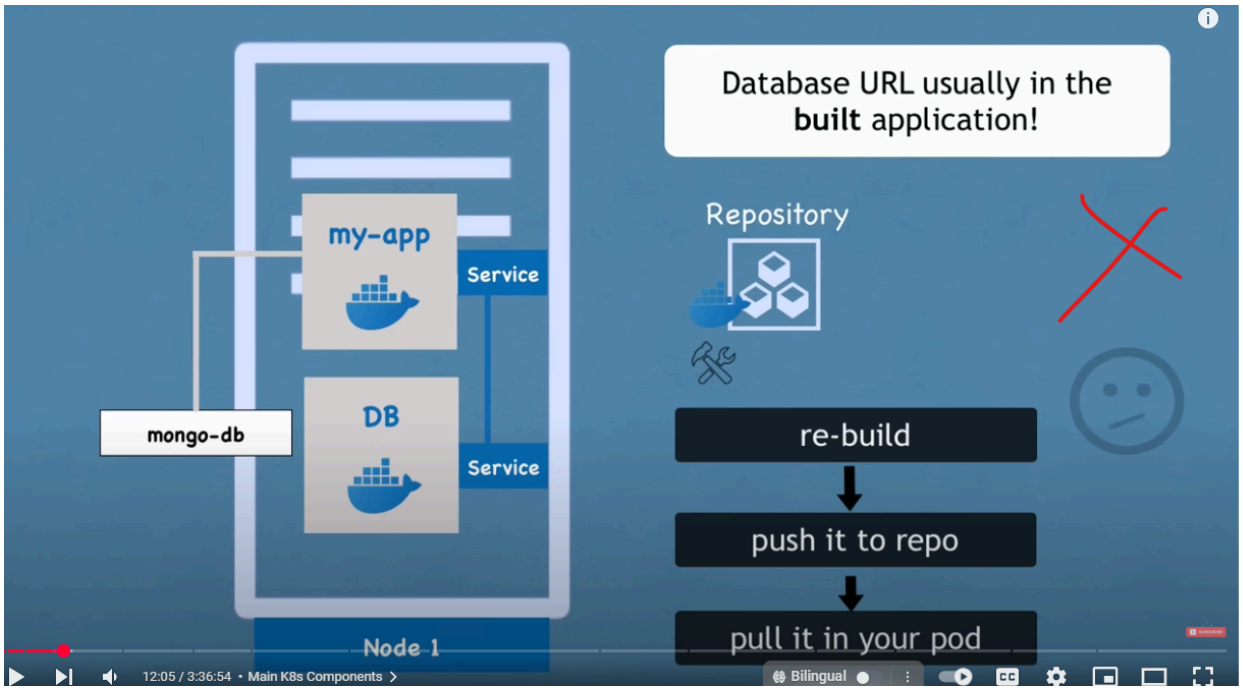




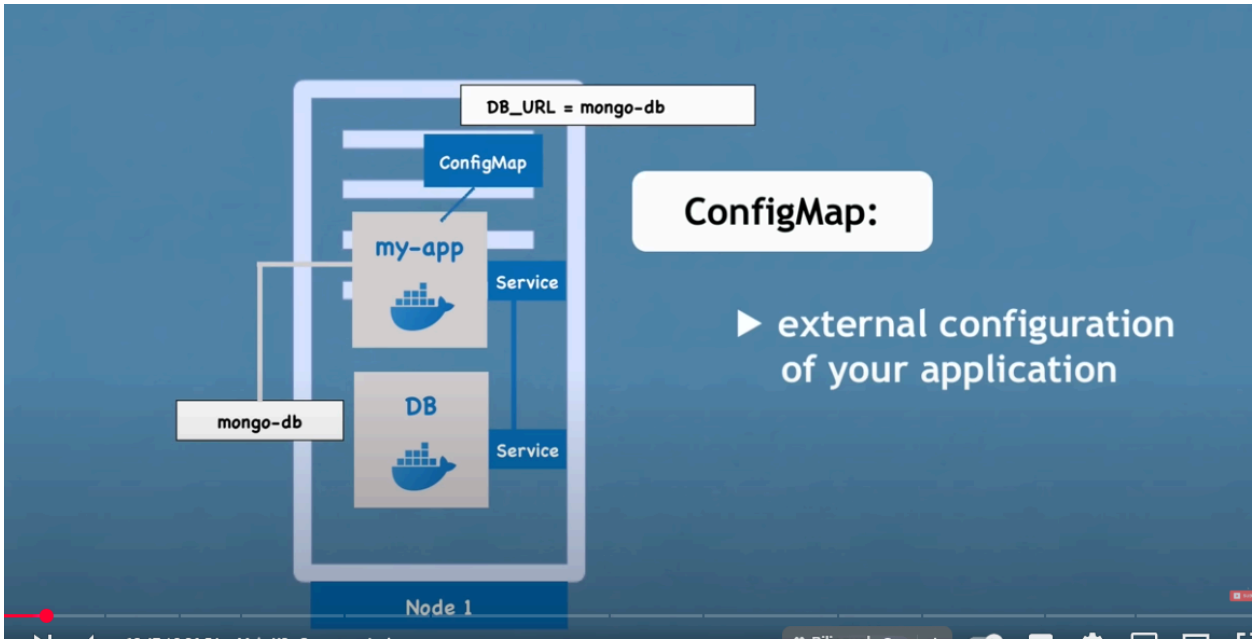
- **Ingress** provides a way to expose services to external traffic.
- It helps with **load balancing**, **SSL termination**, and routing requests to different services based on **URLs** or **hostnames**.
- **Ingress Controller** is required to process the Ingress rules and route traffic accordingly.

Why do we need Config Map

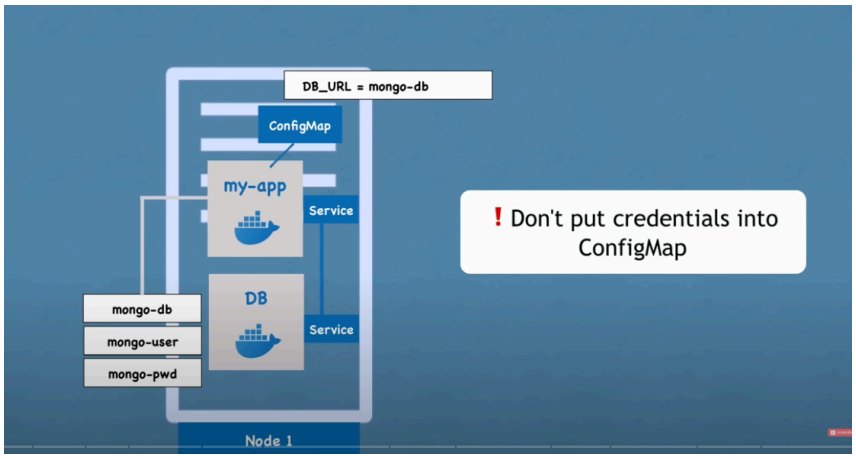
For example if we want to change db url without config map:



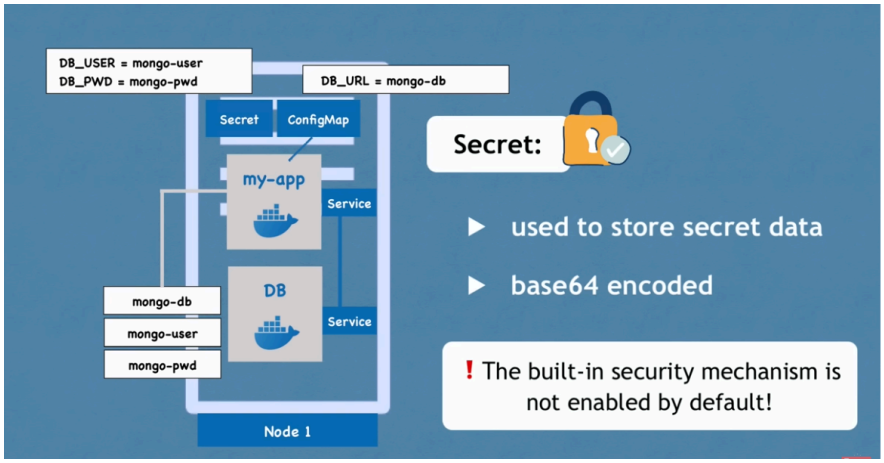
With config map:



Addingfor example db credentials into config map is not secure:



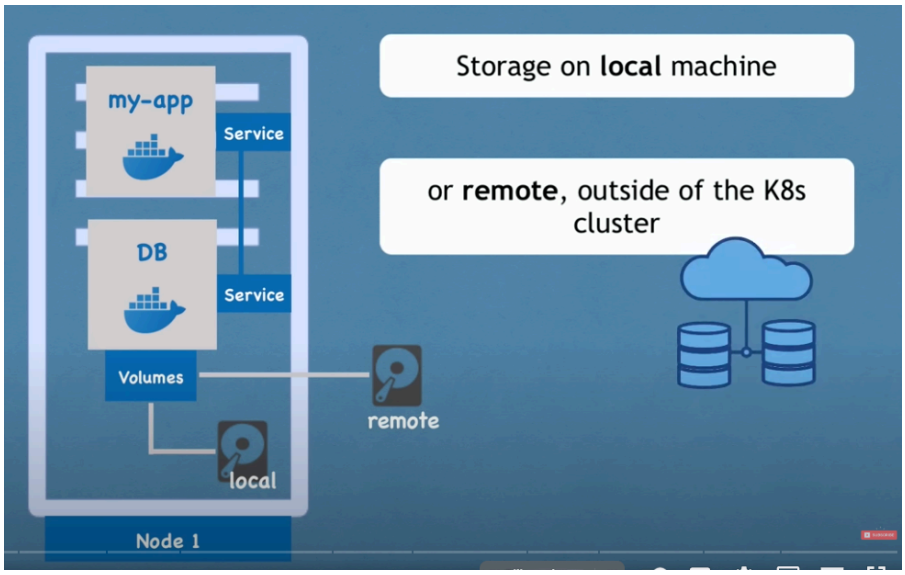
Instead:

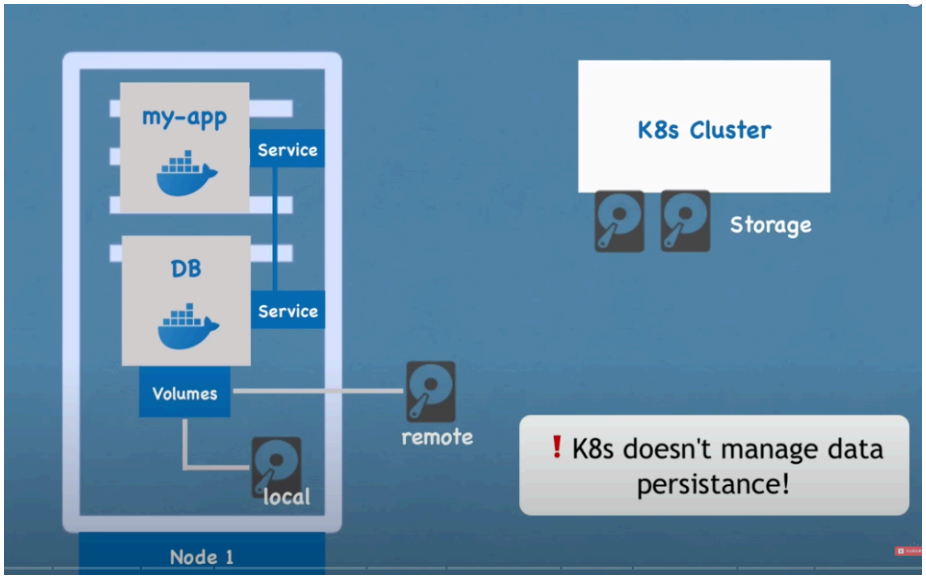


Secrets are **safer than ConfigMaps** because they use `/etc/secrets/`, **file permissions**, and **etcd encryption**.
Base64 **alone is not secure**; always enable **etcd encryption** and **RBAC controls**.

Why do we need volumes?

storing data directly in a Pod is risky because **Pods are ephemeral**. If a Pod dies, any data stored inside it is lost, which can be catastrophic for your application. That's why **persistent storage** is crucial in Kubernetes So why we need volumes





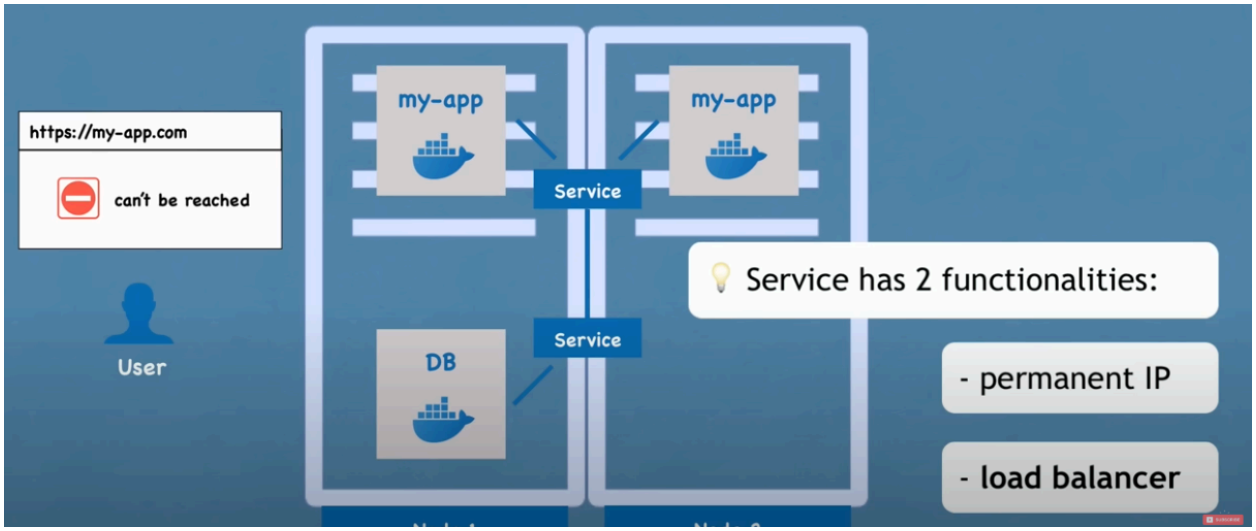
Kubernetes helps with persistent storage and makes sure data is available even if Pods fail, by attaching Persistent Volumes to Pods.
You are responsible for:

- **Backups**: Kubernetes doesn't automatically back up your data. You need to set up a backup strategy.
- **Data Replication**: Kubernetes won't replicate your data automatically unless your application (e.g., a database) handles it.

- **High Availability**: For **data redundancy** and **failover** (especially for databases), you have to set it up within your application or use tools like **Replicas**.

  Replica Sets and services

- **ReplicaSet** continuously ensures the correct number of Pods are running.
- The **Service** just keeps routing traffic to whatever Pods are **available** and match its label selector.
- **One Service** can manage multiple replicas. It load-balances traffic across all available Pods.
- Kubernetes ensures that when a **Pod fails**, it replaces it automatically to maintain the desired replica count.
- The Service routes traffic to all Pods that match its label selectors, without needing to know how many replicas exist.



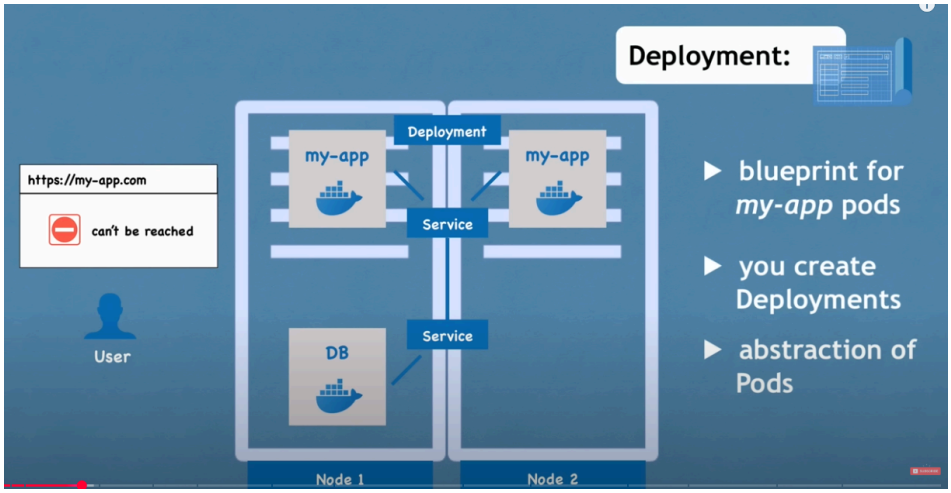
What Does "Blueprint for an App" Mean?

Blueprint for an app is a template or design that defines how the app should be deployed and run (e.g., using YAML files for Deployments, Services).

- In Kubernetes, this **blueprint** is often represented as **YAML files** (for Deployments, Services, etc.).
- It defines things like:
  - The number of replicas.
  - The container image to use.
  - The resources (CPU, memory) required.
  - Environment variables.
  - Networking details (e.g., ports, services).

A blueprint can be thought of as a **template** that tells Kubernetes how to **deploy** and **manage** the application.

Deployment

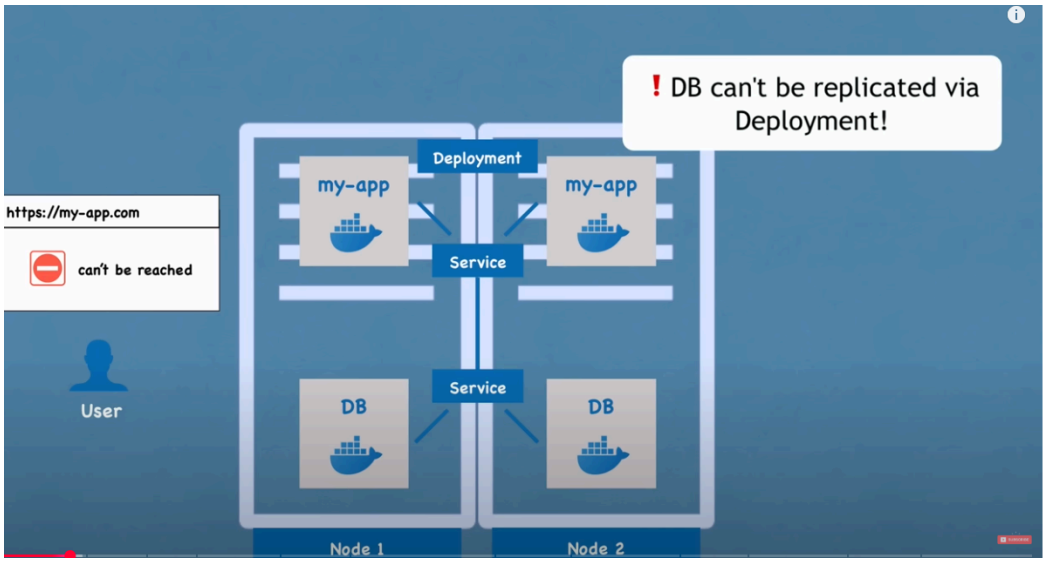


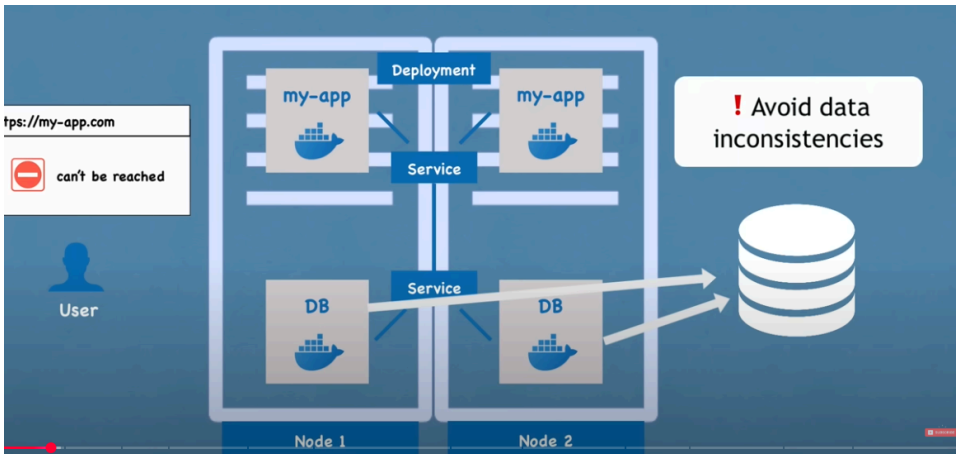
How Database Replication Works:
Kubernetes doesn't handle replication for you, but it can **manage** the **Pods** and **storage**. You need to set up **database replication** inside your database (using tools like MySQL or MongoDB replication).

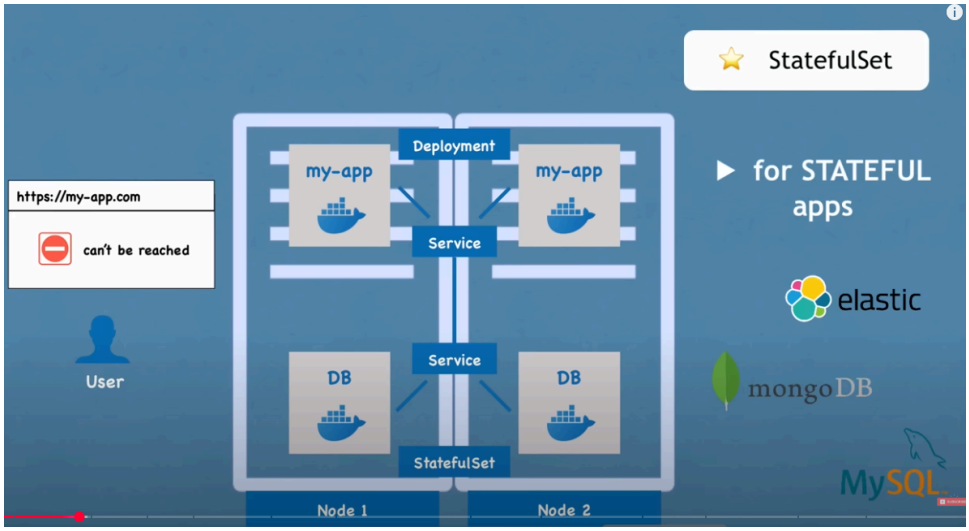
What Kubernetes Does:

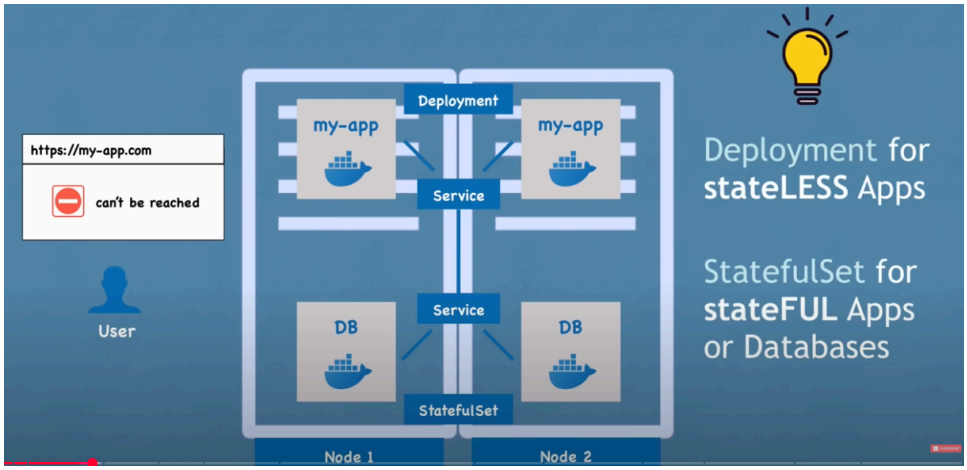
- Kubernetes **manages** the **Pods** running your database, using **StatefulSets** instead of Deployments.
- **StatefulSets** ensure each database replica has its own **persistent storage** and stable network identity.

But the actual **replication** (syncing data between database instances) needs to be set up **inside the database** (using MySQL, PostgreSQL, MongoDB replication features, etc.).


! DB can't be replicated via Deployment!
https://my-app.com
can't be reached
User
Deployment
my-app
my-app
Service
Service
DB
DB
Node 1
Node 2


tps://my-app.com
can't be reached
User
Deployment
my-app
my-app
Service
Service
DB
DB
! Avoid data inconsistencies
Node 1
Node 2


StatefulSet
https://my-app.com
can't be reached
User
Deployment
my-app
my-app
Service
Service
DB
DB
StatefulSet
Node 1
Node 2
for STATEFUL apps
elastic
mongoDB
MySQL



- The **primary** database handles **write** operations.
- The **replicas** handle **read** operations.
- **Kubernetes** ensures the databases are running, but **database replication** and **read/write routing** are managed by the **database itself** and your **application** or a **proxy**.