BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Kubernetes

Aditya Goel

Agenda

- **Kubernetes**
  - ✓Introduction
  - ✓Architecture
  - ✓Components
  - ✓Hands-On Demo

Kubernetes

Official definition of Kubernetes:

- Open source container orchestration tool
- Developed by Google
- Helps manage containerized applications

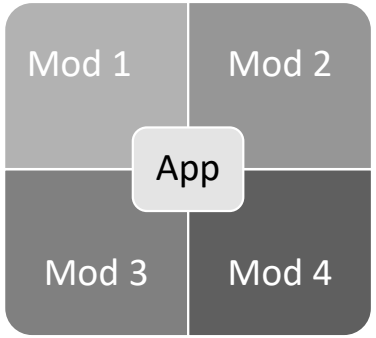
Why Kubernetes?

- What Problems does Kubernetes solve?
- What are the tasks of an orchestration tool?

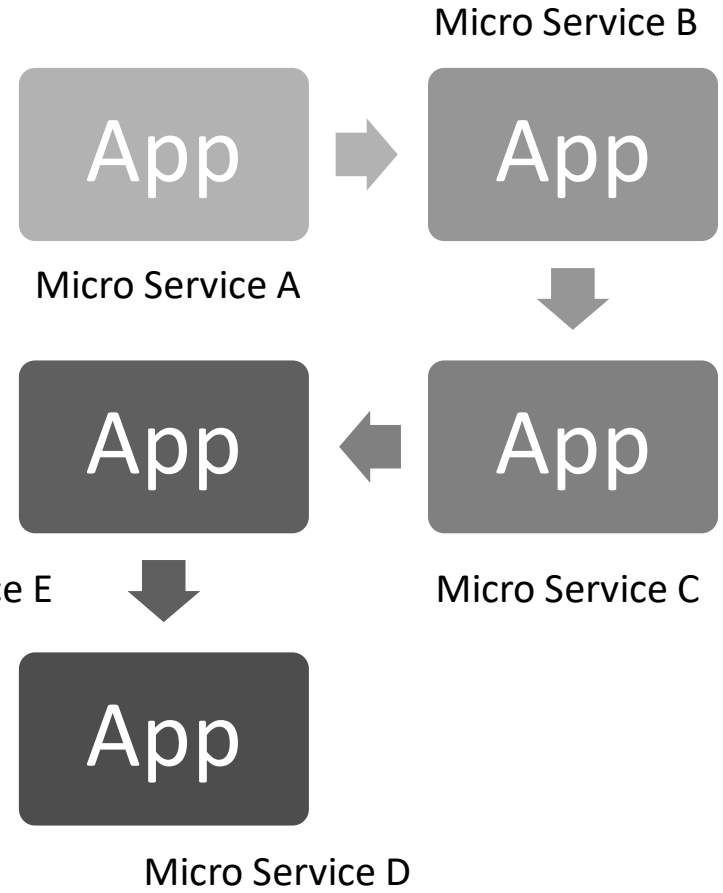
Why Kubernetes?

Need for container orchestration tool

- Trend from Monolithic arch to Micro-Services arch



Monolithic



Micro Service B

Micro Service A

Micro Service C

Micro Service E

Micro Service D

Kubernetes - Features

What features do orchestration tools offer?

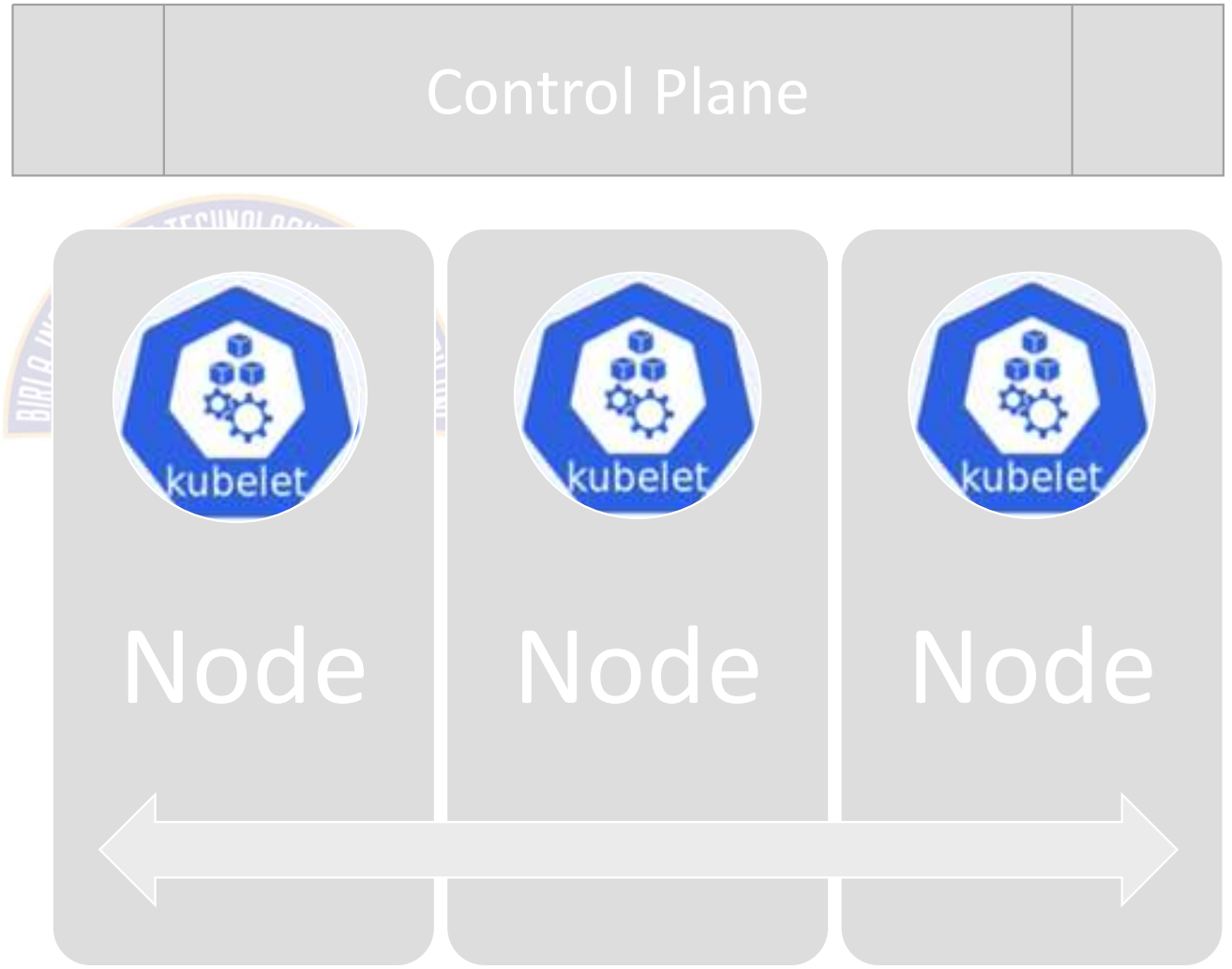
- High Availability or no downtime

- Scalability or high performance

- Disaster recovery – backup and restore

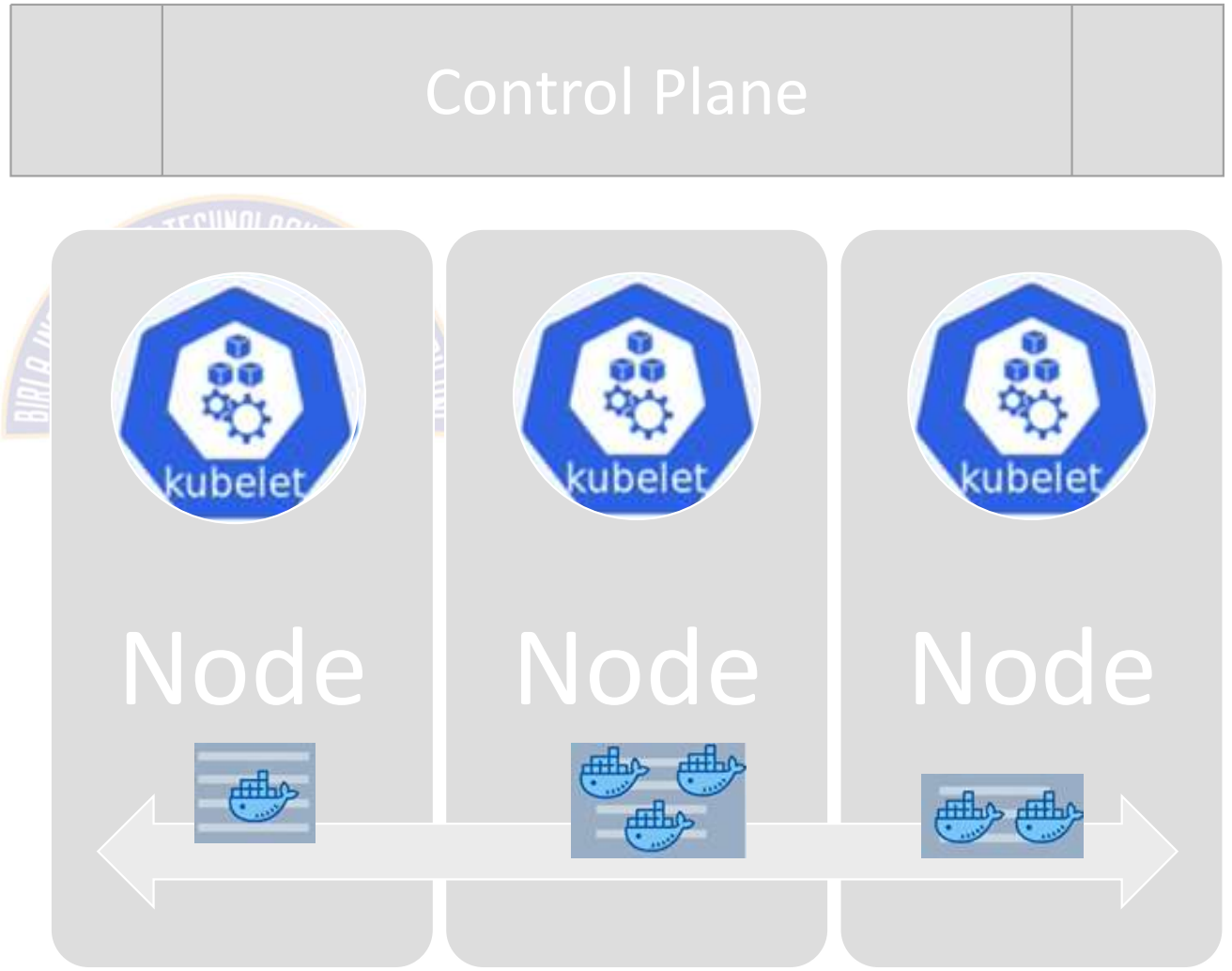
Kubernetes - Architecture

Primary “node” agent



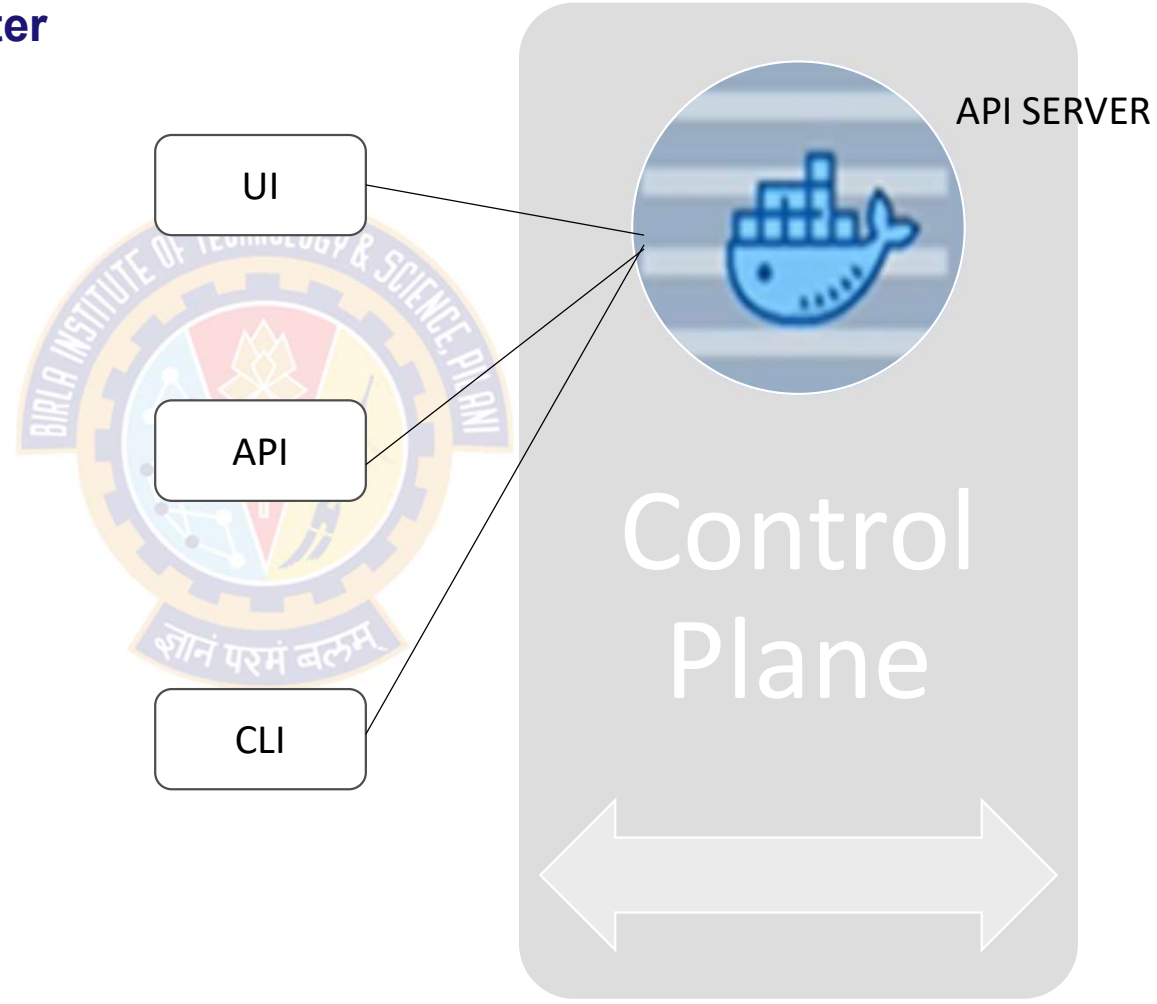
Kubernetes - Architecture

Primary “node” agent



Kubernetes - Processes

API Server: Entry-point to K8s cluster

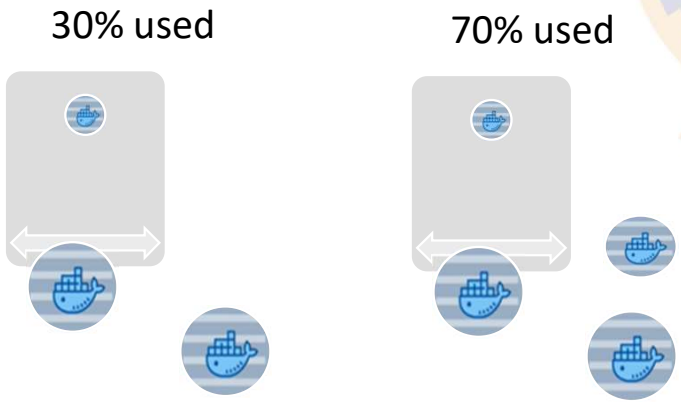


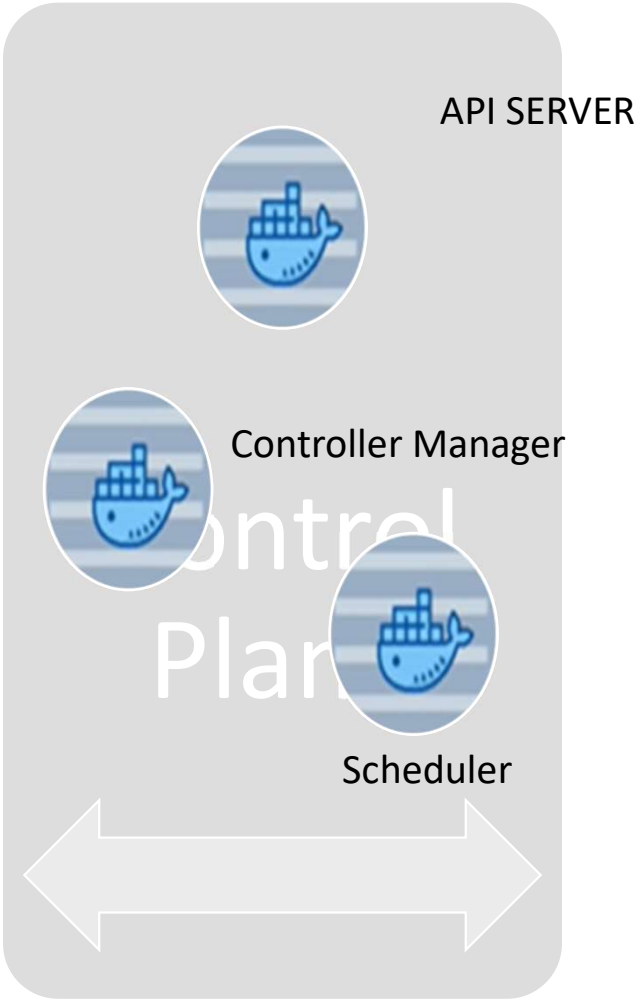
Kubernetes - Processes

API Server: Entry-point to K8s cluster

Controller manager: keeps track of whats happening in the cluster

Scheduler: ensure pods placement





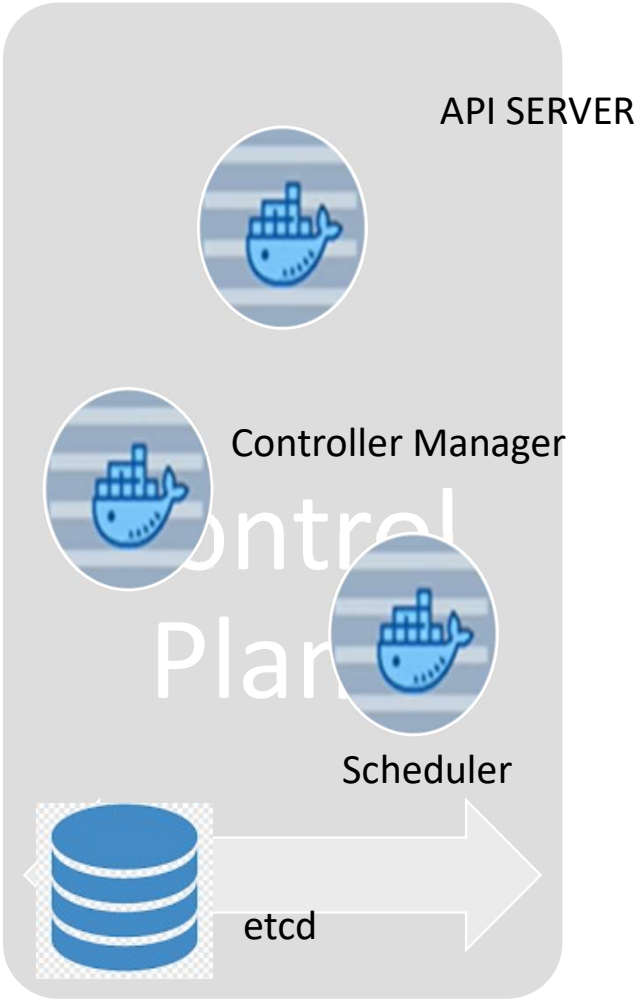
Kubernetes - Processes

API Server: Entry-point to K8s cluster

Controller manager: keeps track of whats happening in the cluster

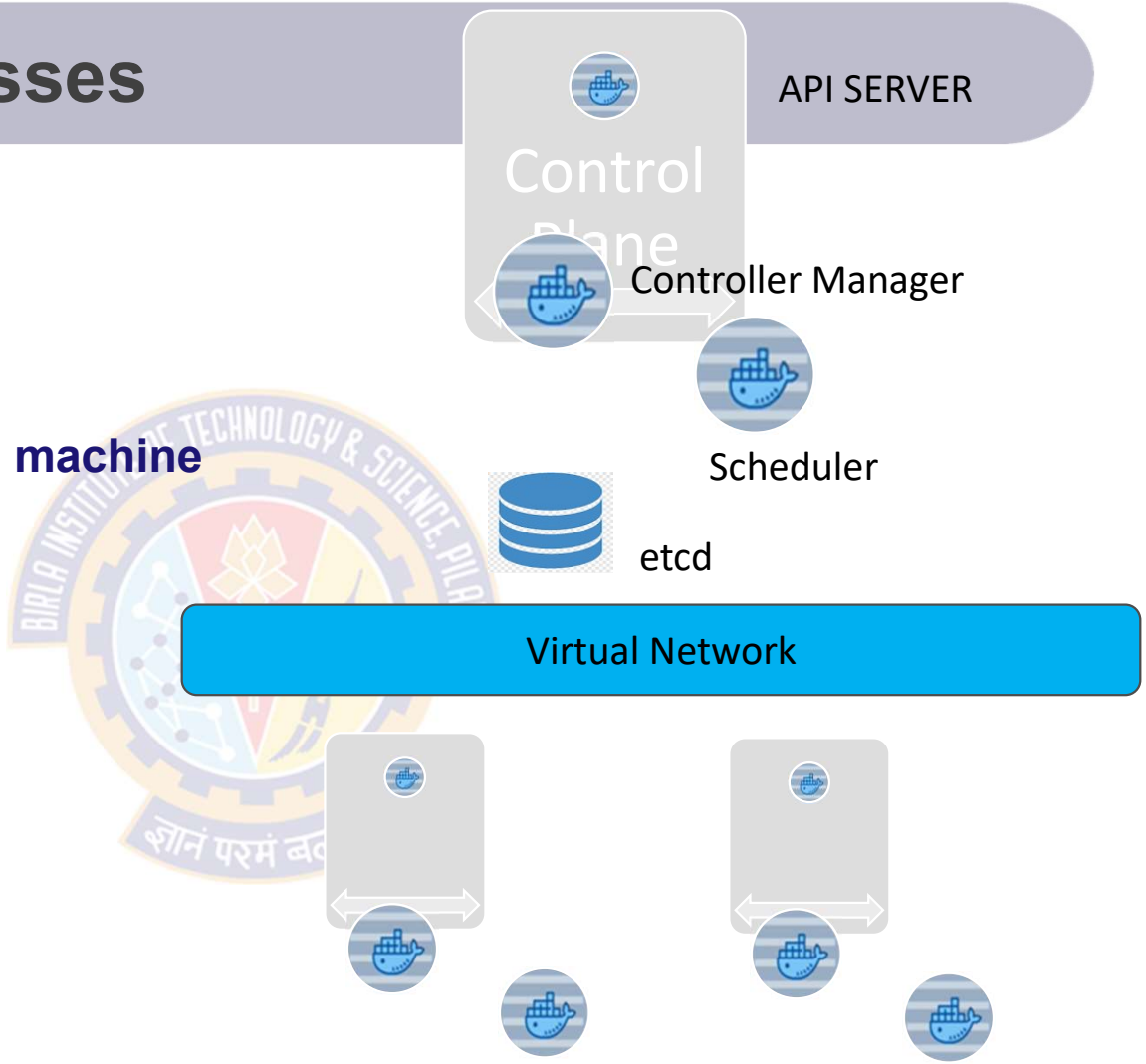
Scheduler: ensure pods placement

etcd: Kubernetes backing store

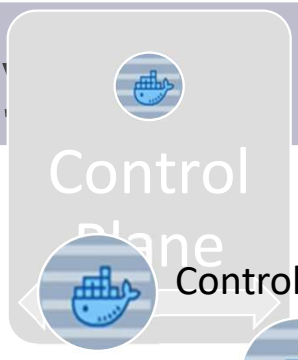


Kubernetes - Processes

Virtual Network: creates one unified machine



Kubernetes – Master Redundancy



API SERVER

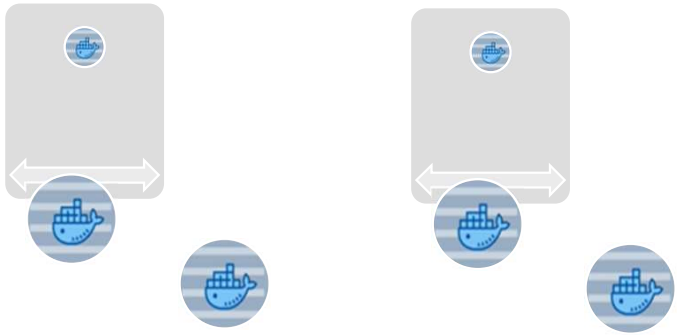
Control Plane Nodes

Handful of master processes

Controller Manager

Scheduler

etcd



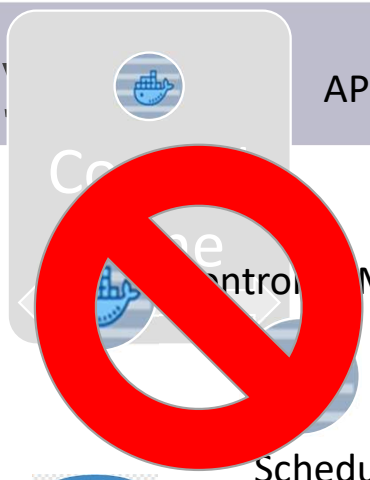
Worker Nodes

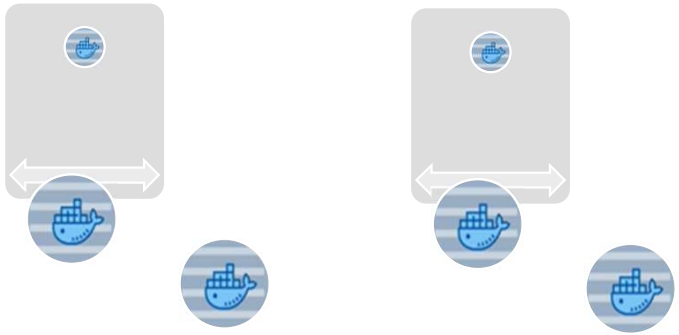
Higher workload

Much bigger and more resources

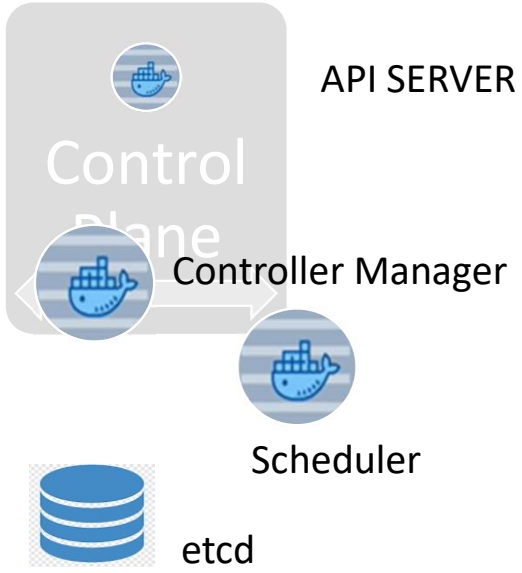
Kubernetes – Master Redundancy

Control Plane Nodes
Handful of master processes
Much more important





Worker Nodes
Higher workload
Much bigger and more resources

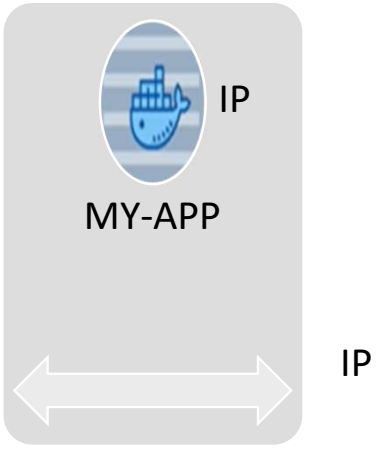


Kubernetes Components

- Pod
- Service
- Ingress
- ConfirgMap
- Secret
- Deployment
- StatefulSet

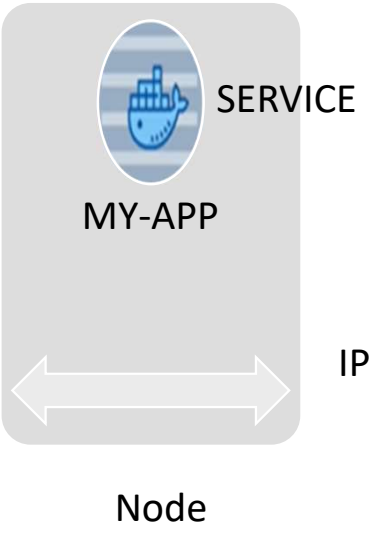
Pod

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



Service

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

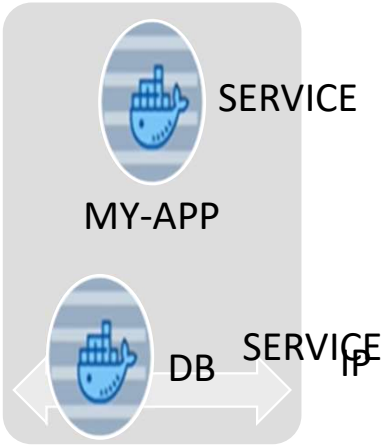


Service

App should be accessible through browser

http://node-ip:port

EXTERNAL SERVICE

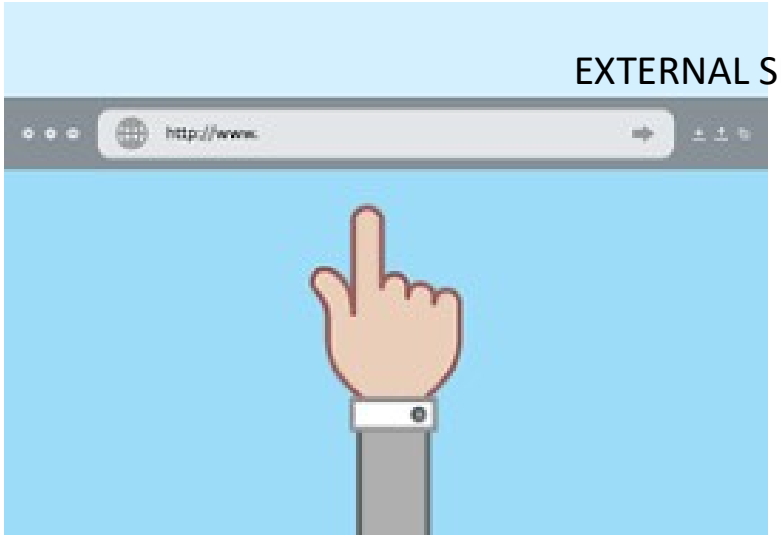


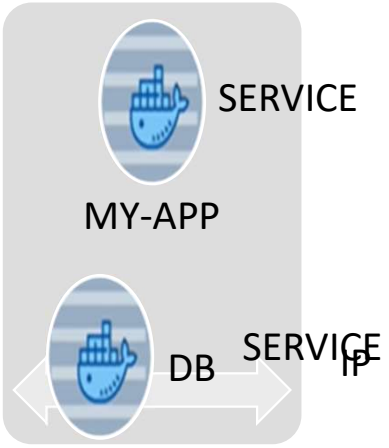
Node

Service

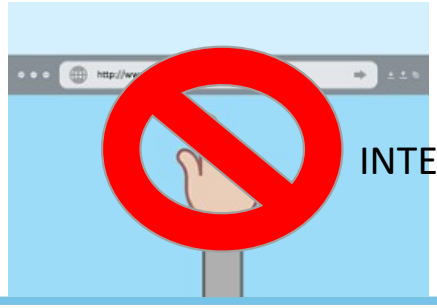
App should be accessible through browser

http://node-ip:port





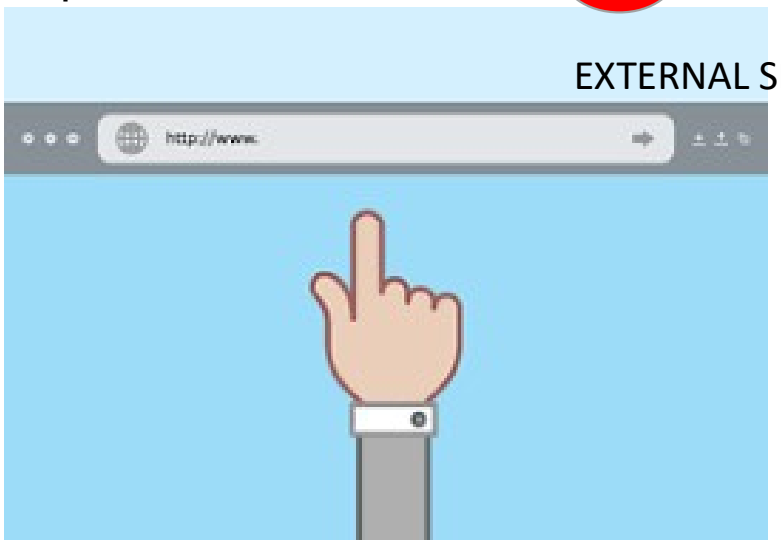
http://db-service-ip:port



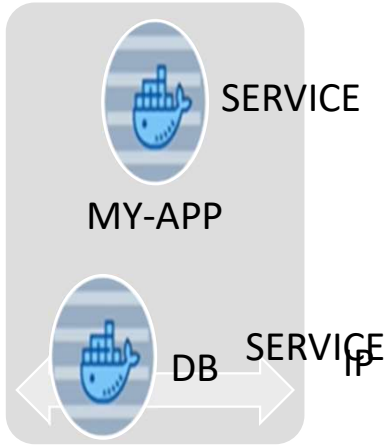
Service

App should be accessible through browser

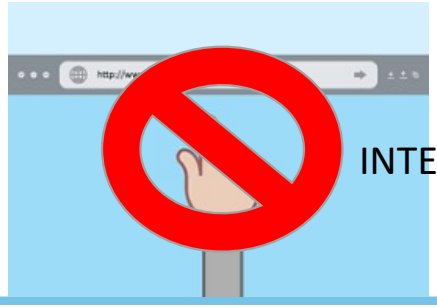
http://124.89.101.2:8080



EXTERNAL SERVICE



http://db-service-ip:port

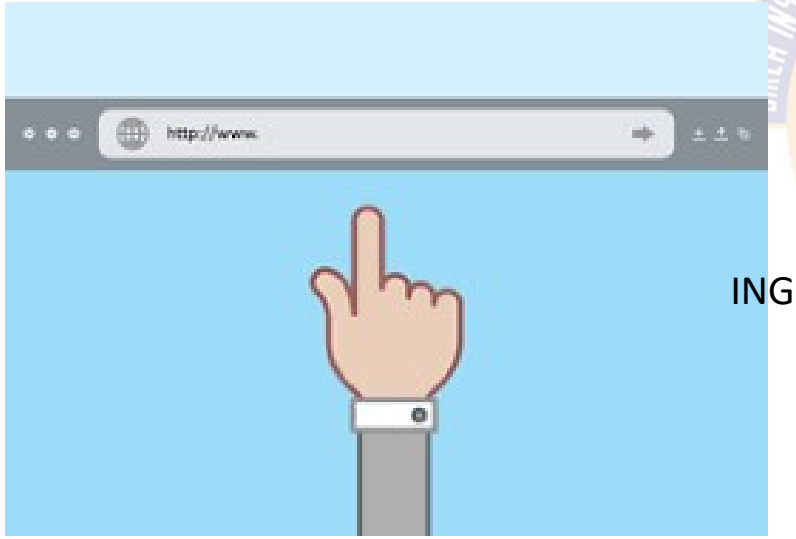


INTERNAL SERVICE

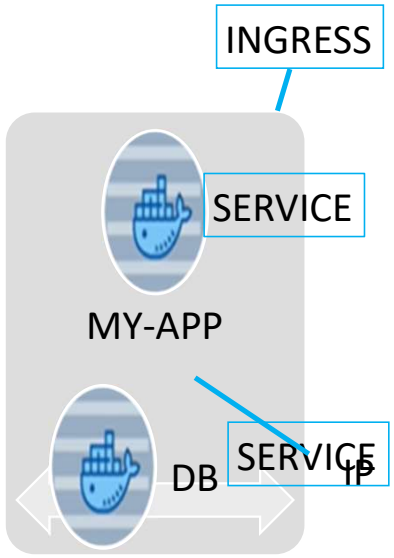
Ingress

App should be accessible through browser

http://my-app.com



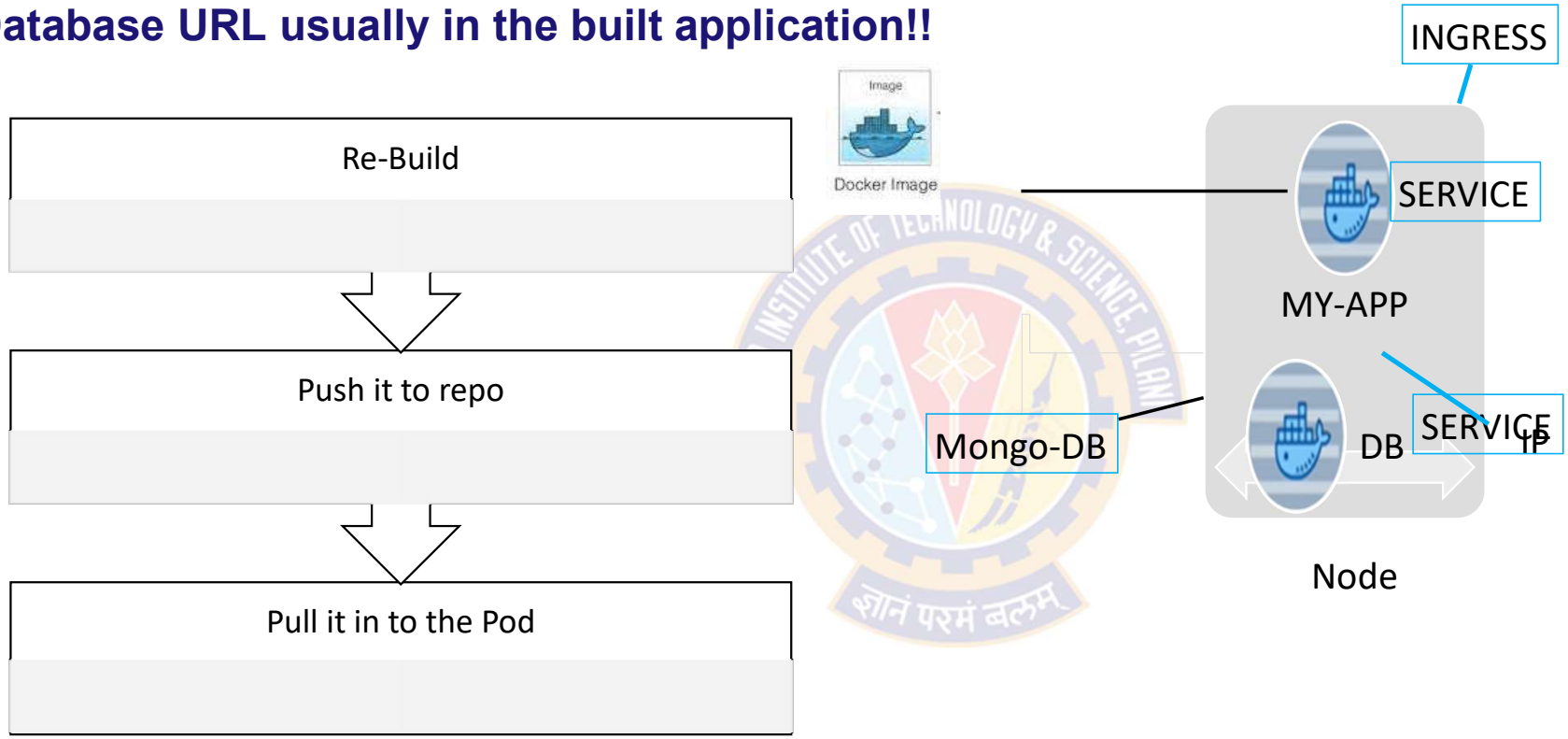
INGRESS



Node

Component?

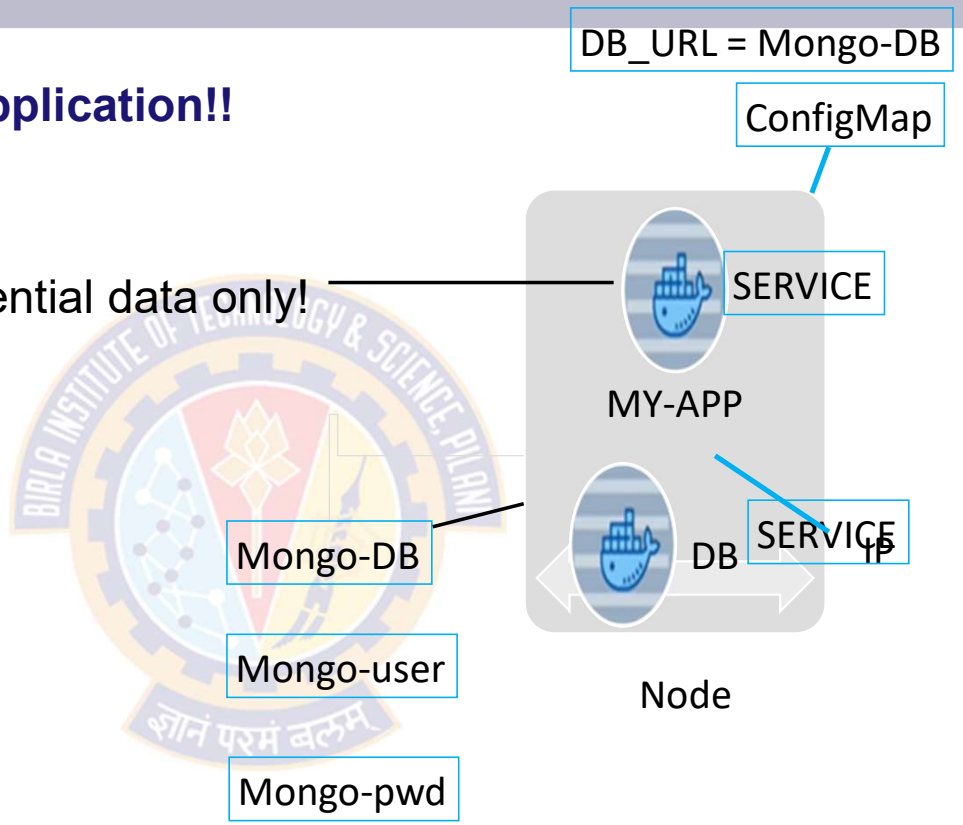
Database URL usually in the built application!!



ConfigMap

Database URL usually in the built application!!

ConfigMap is for non-confidential data only!



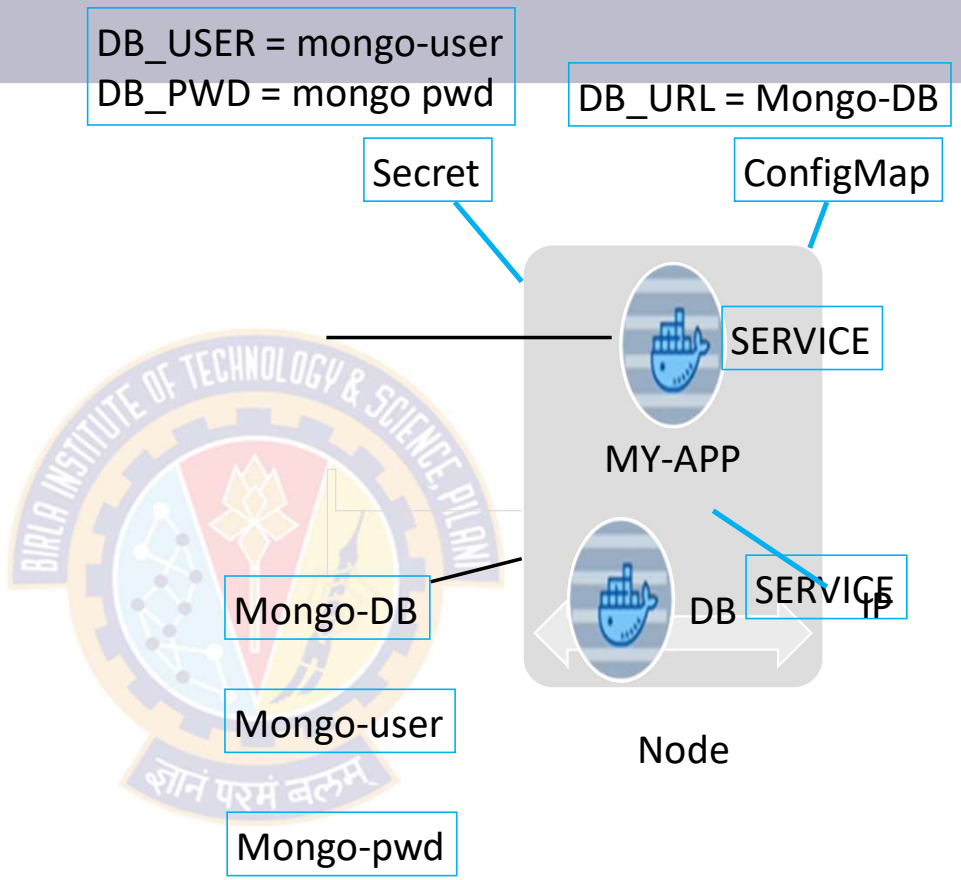

```
Map map = new HashMap<>();

map.put("DB_URL", "http://mongoDB.jswsteel.vasind.bits.edu");
map.put("WS_EndPoint", "http://weatherforecastingdata.gov.in");
map.put("NumberOfThreadsAllowed", 10);
```

Secret

Used to store secret data

Reference Secret in Deployment



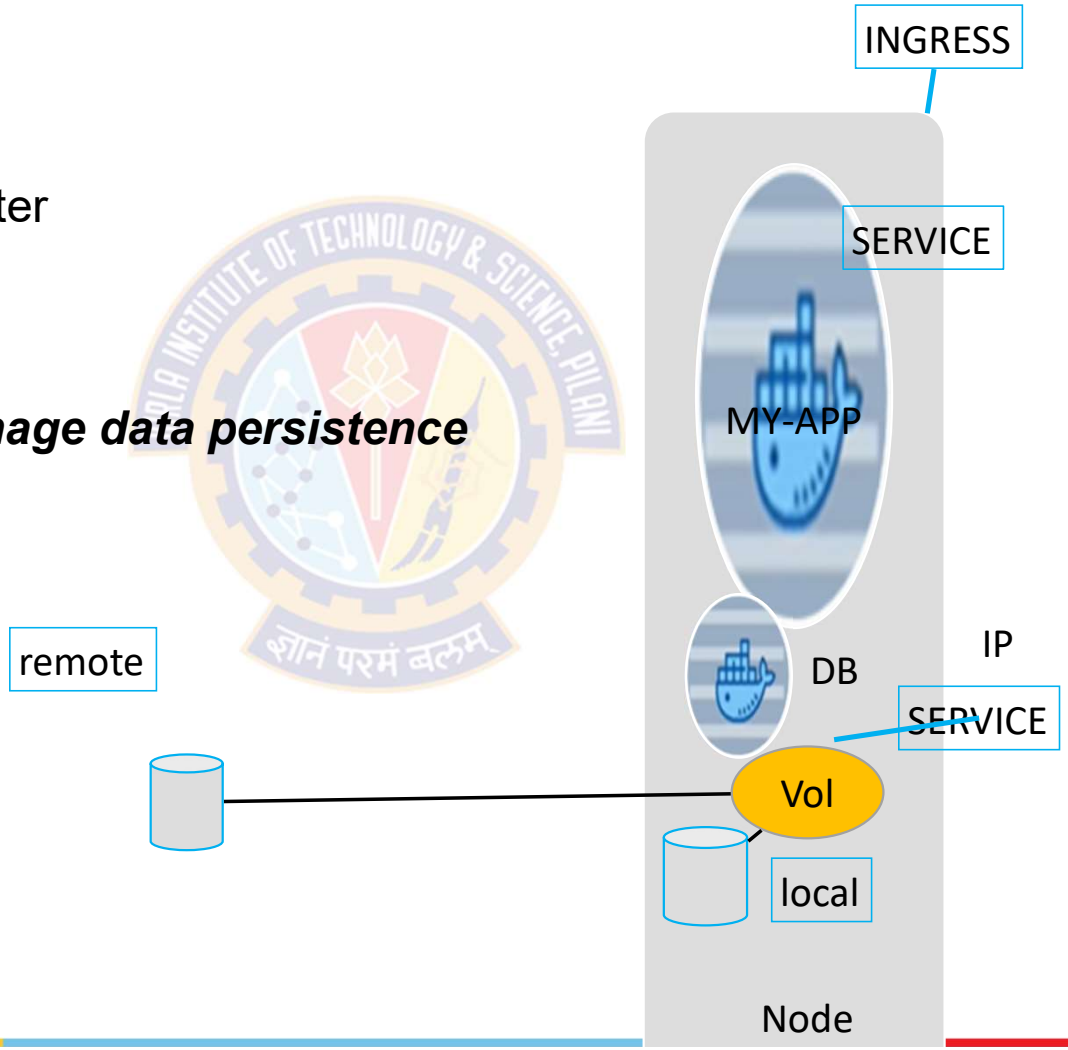
Vol

Storage

Storage on local machine

Or remote, outside the K8s cluster

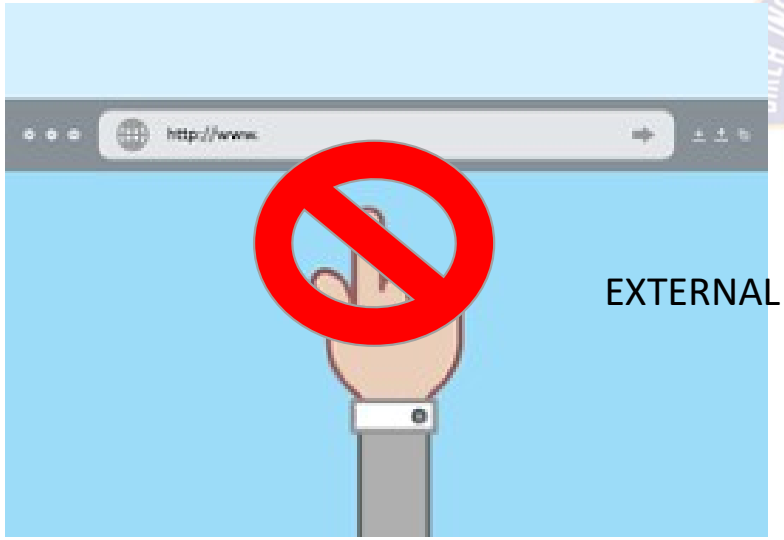
PS: Kubernetes does not manage data persistence



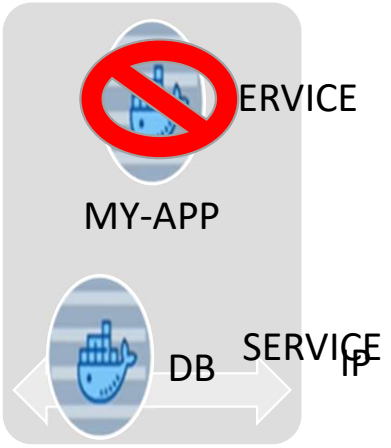
Deployment??

App should be accessible through browser

http://node-ip:port



EXTERNAL SERVICE

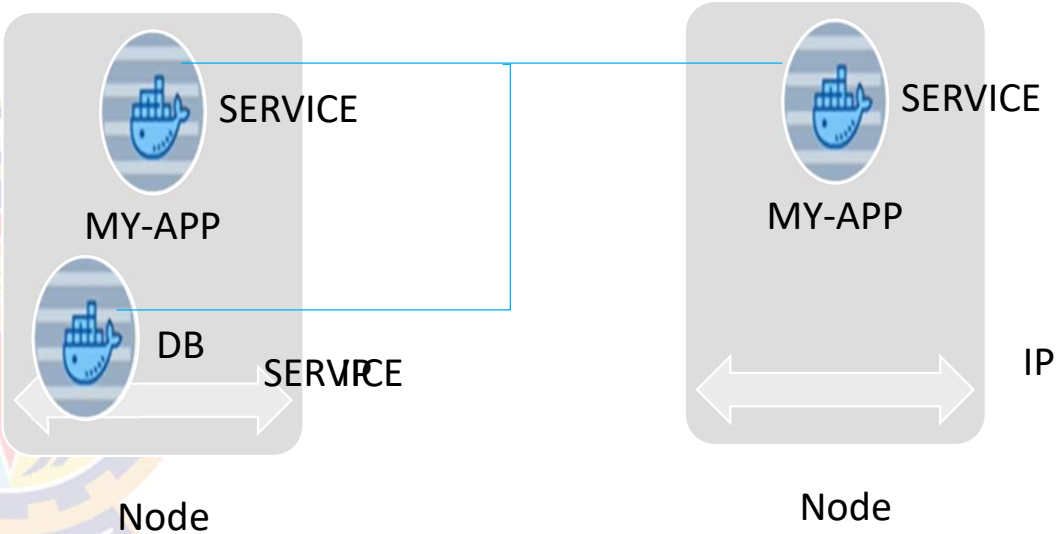


Node

Deployment??

Permanent IP

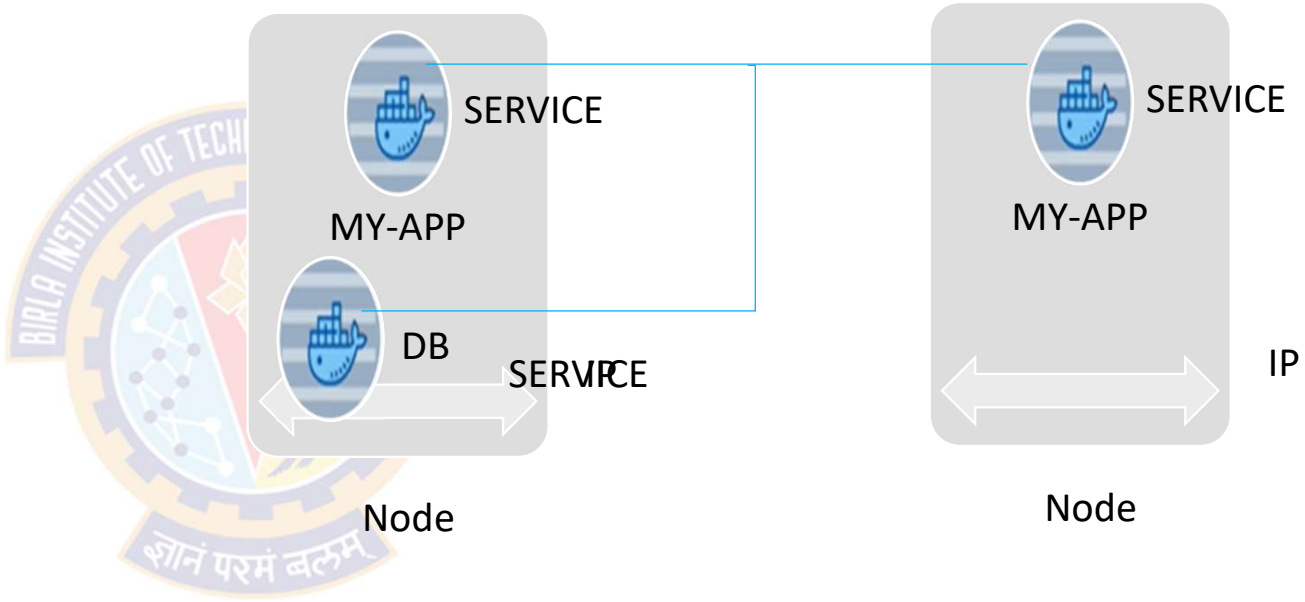
Load balancer



Deployment

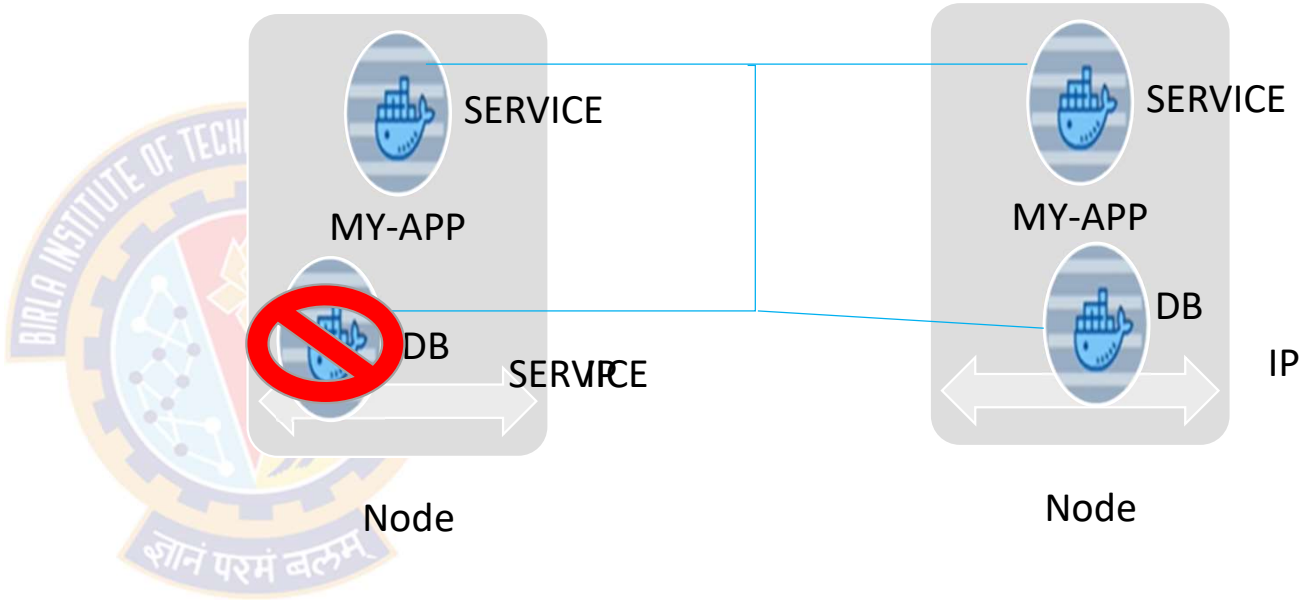
Blueprint for “MY-APP” Pods

Create “Deployments”



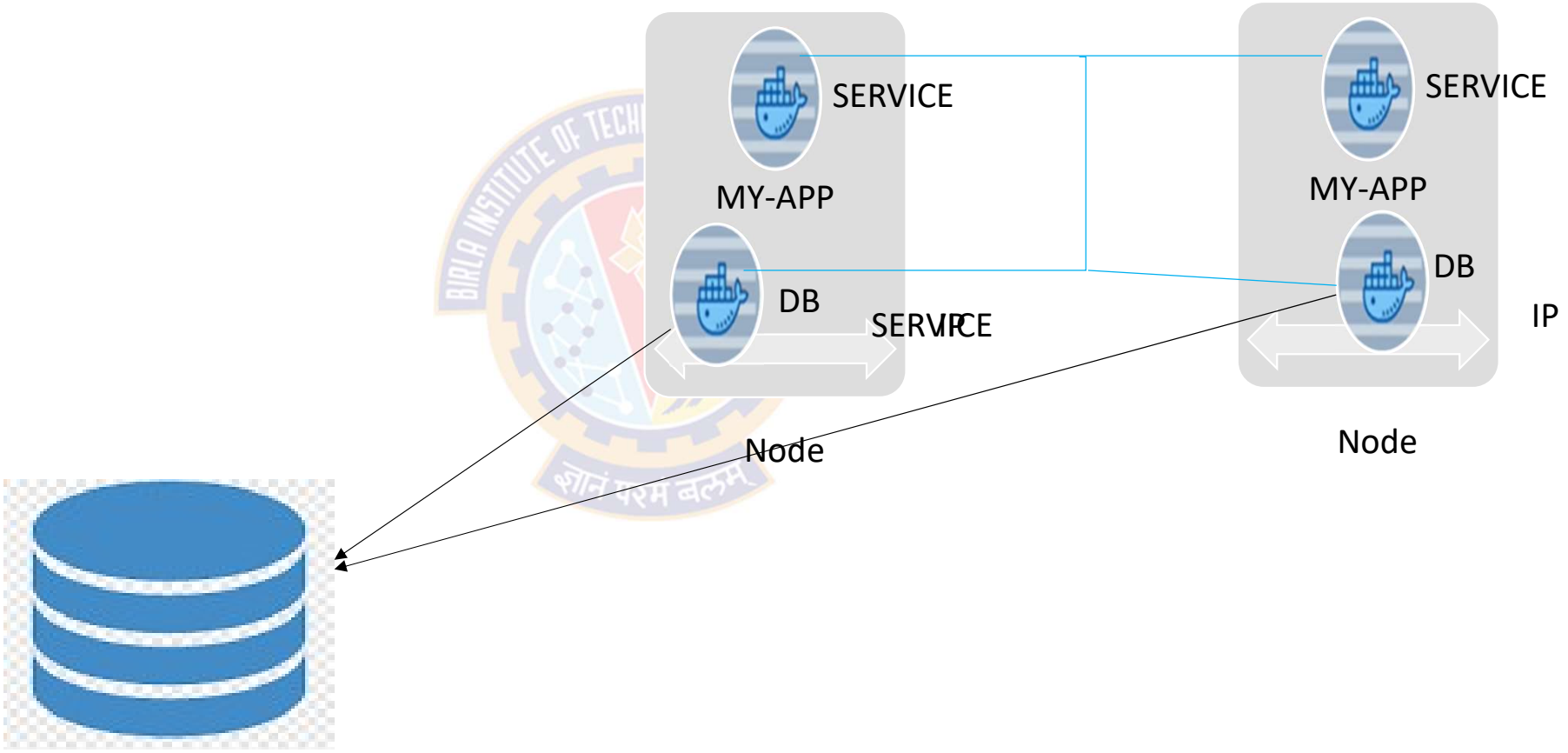
Deployment

DB can not be replicated via Deployment



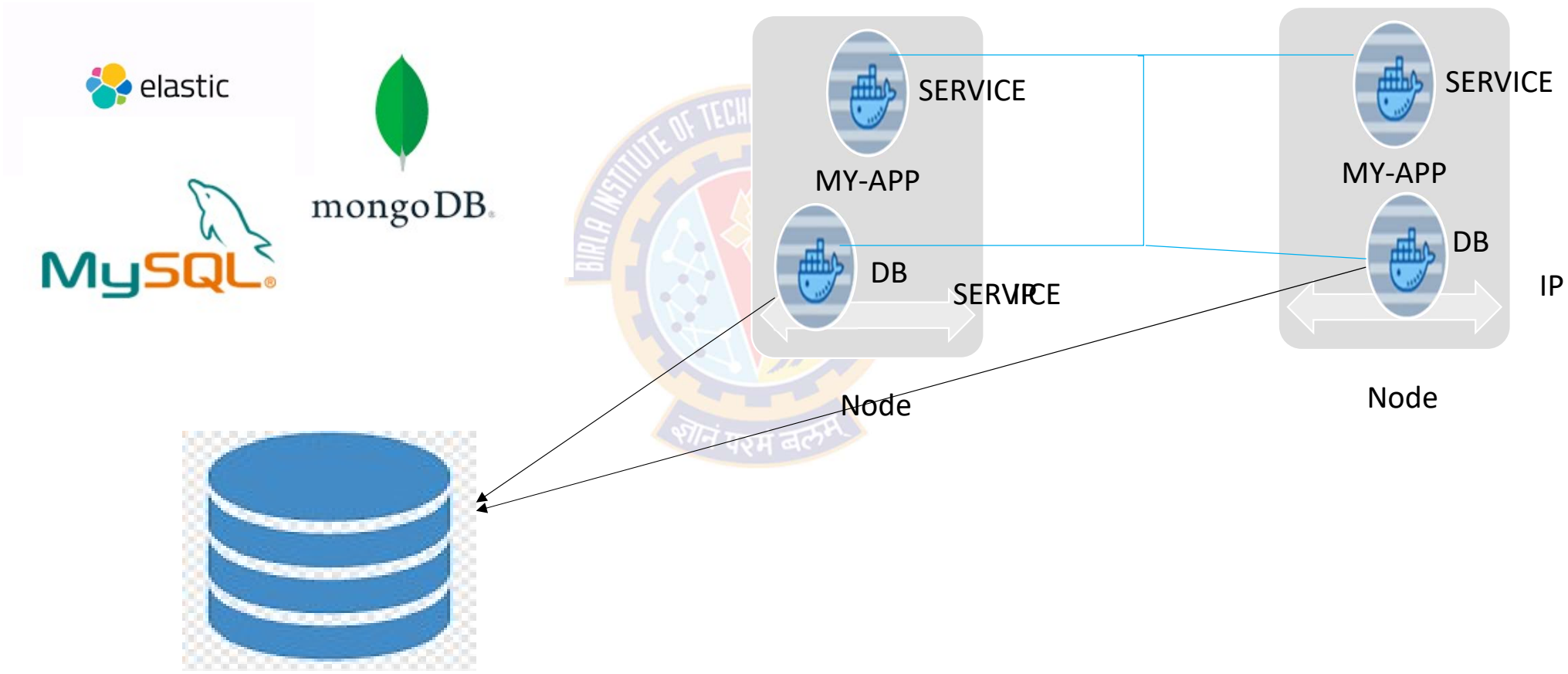
Deployment

DB can not be replicated via Deployment



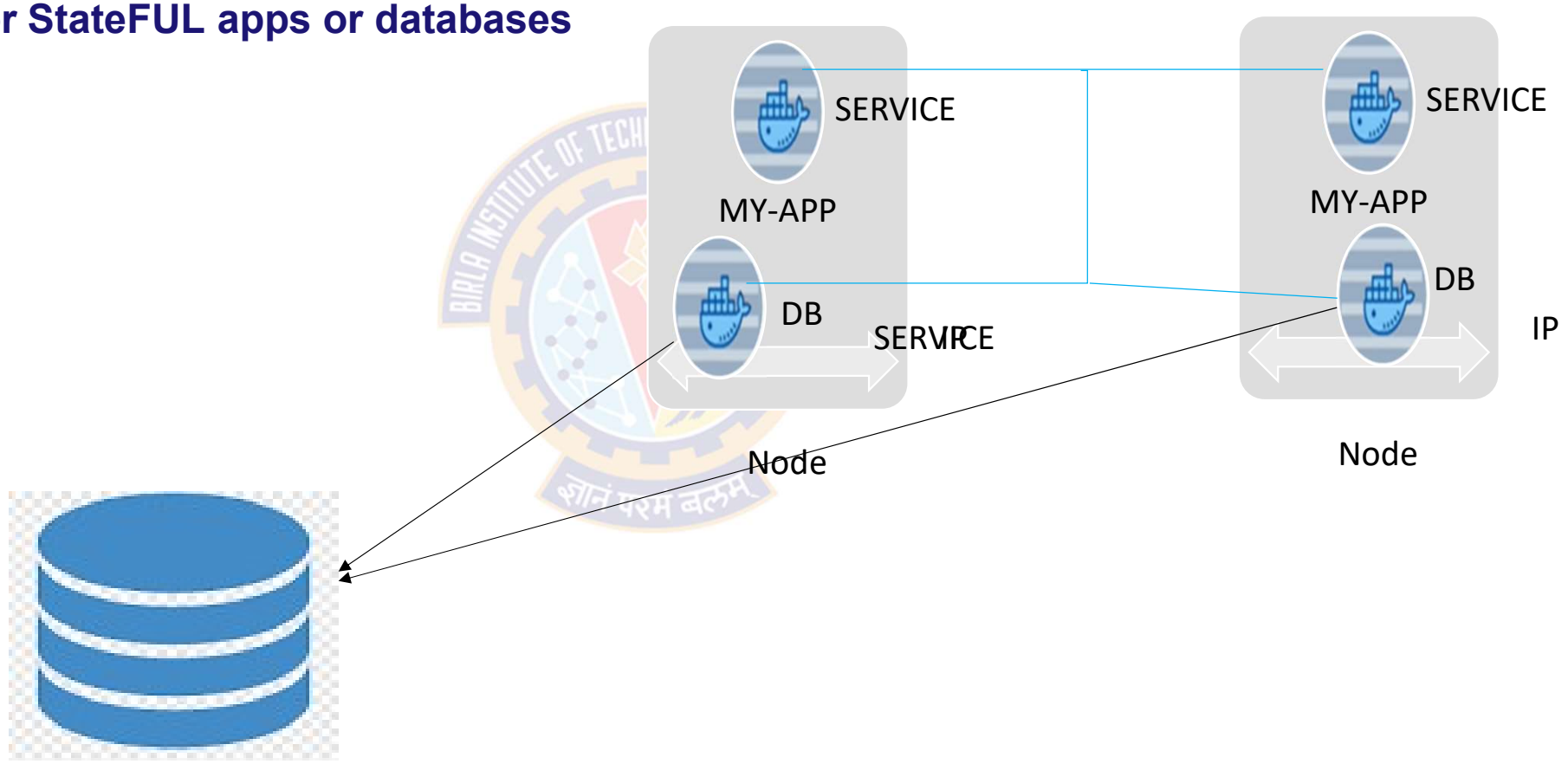
StatefulSet

For Stateful apps



Deployment for Stateless apps

StatefulSet for StateFUL apps or databases

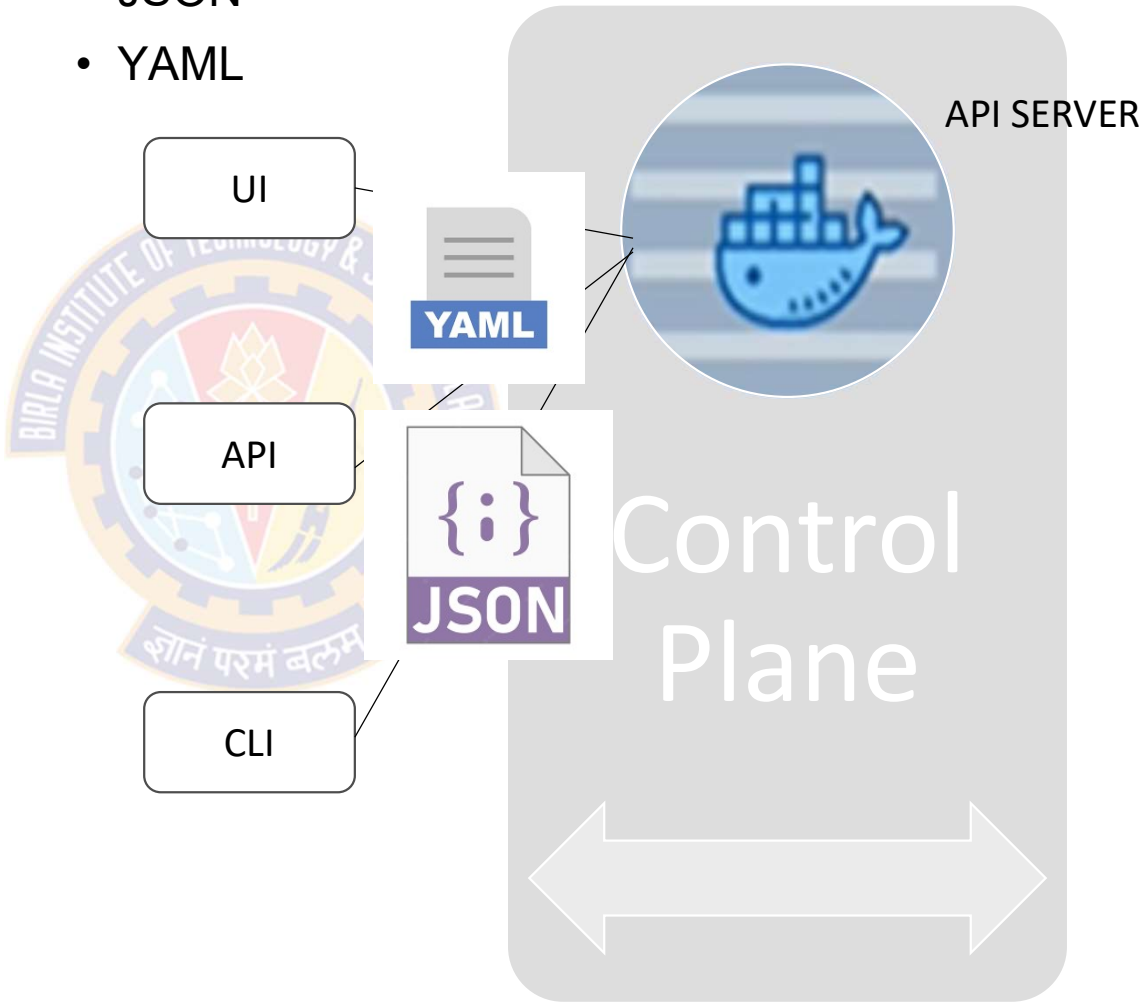


Kubernetes - Configuration

API Server: Entry-point to K8s cluster

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
  labels:
    app: my-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-app
          image: my-image
          env:
            - name: SOME_ENV
              value: $SOME_ENV
          ports:
            - containerPort: 8080
```

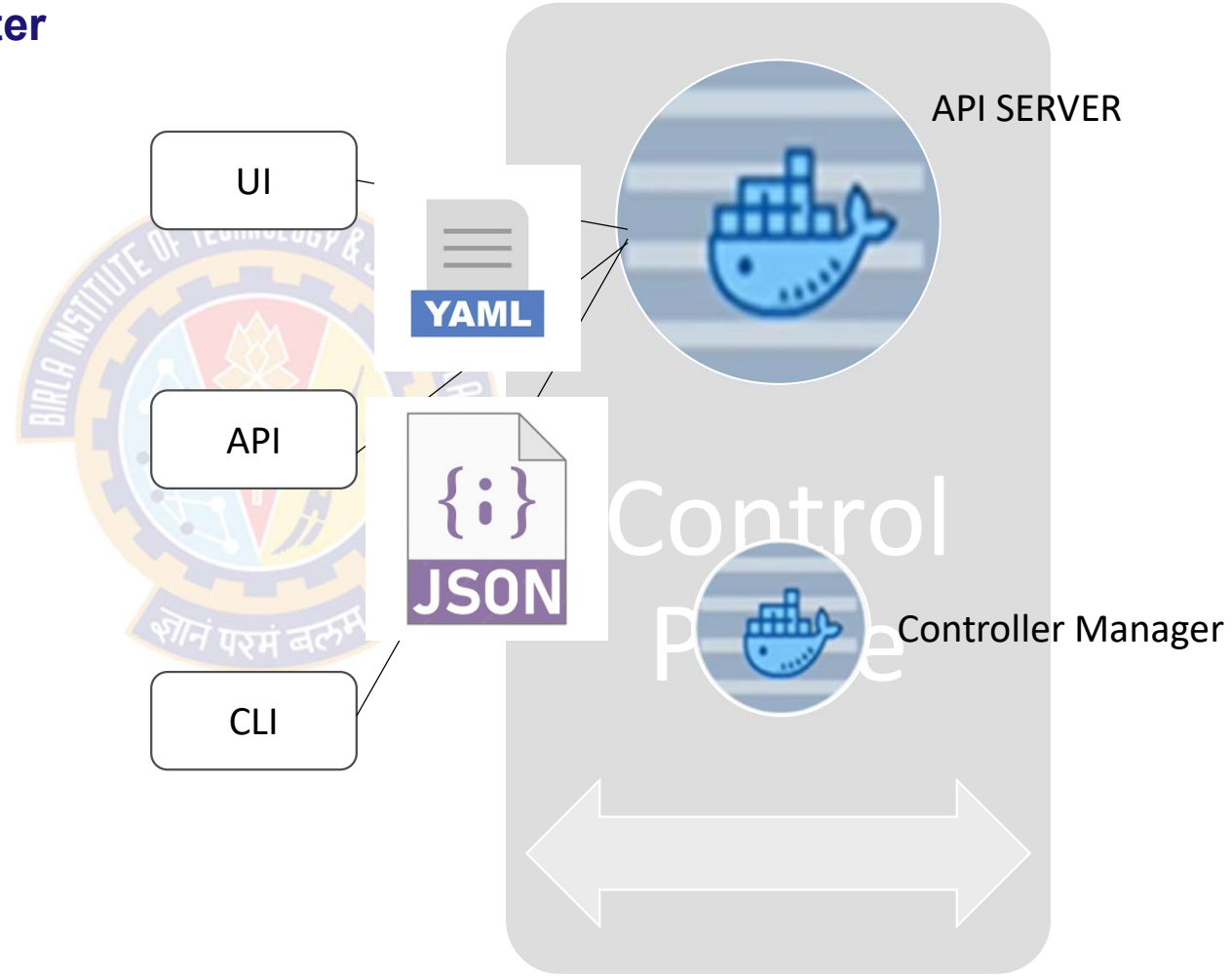
- JSON
- YAML



Kubernetes - Configuration

API Server: Entry-point to K8s cluster

- Declarative
- Is == Should



Kubernetes Configuration

- Each Configuration file has 3 parts:
  - **Metadata**
  - Specification
  - Status

Deployment

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
```

Service

nginx-service.yaml

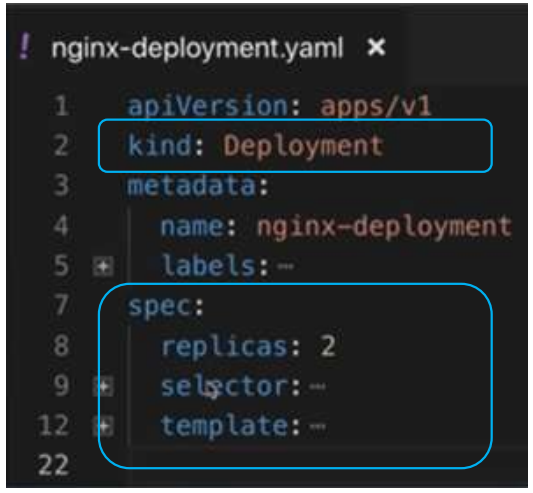
```
1   apiVersion: v1
2   kind: Service
3   metadata:
4     name: nginx-service
5   spec:
6     selector:…
8     ports:…
12
```

Kubernetes Configuration

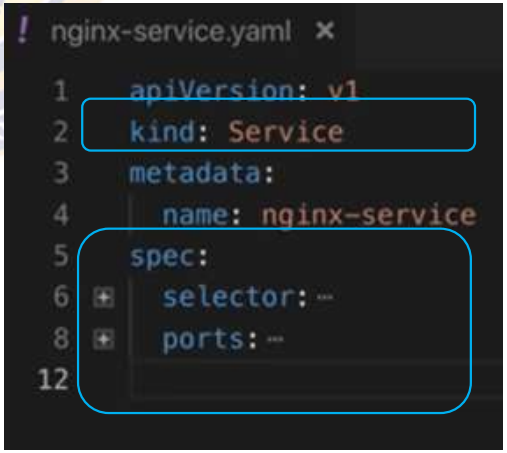
- Each Configuration file has 3 parts:
  - Metadata
  - **Specification**
  - Status

Attributes of "**Spec**" are specific to the "**kind**"

Deployment



Service



Kubernetes Configuration

- Each Configuration file has 3 parts:
  - Metadata
  - Specification
  - **Status : K8s updates the state continously**

Deployment Desired? Vs Actual?

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?

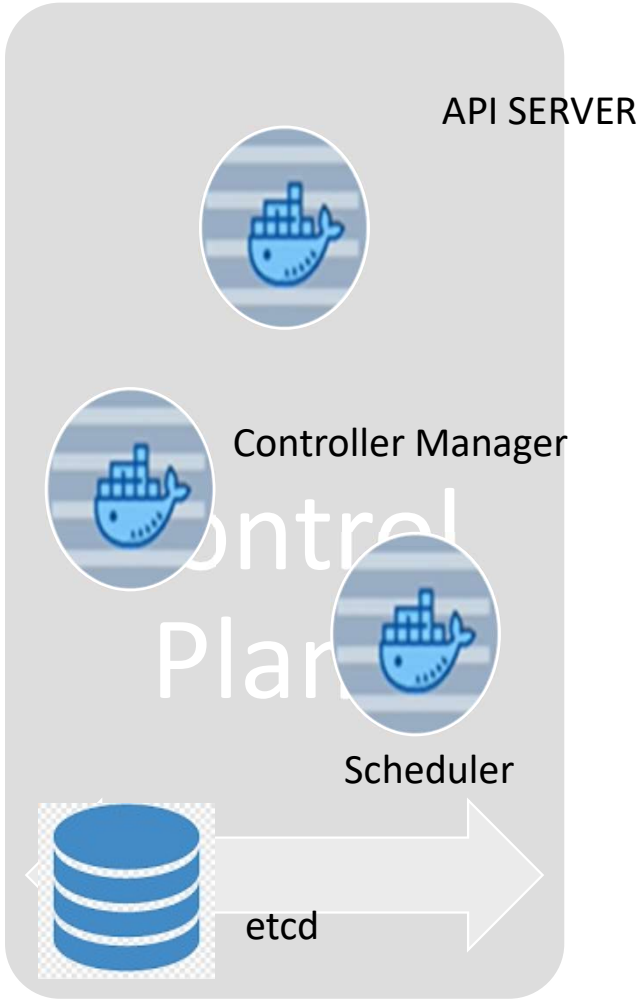
API Server: Entry-point to K8s cluster

Controller manager: keeps track of whats happening in the cluster

Scheduler: ensure pods placement

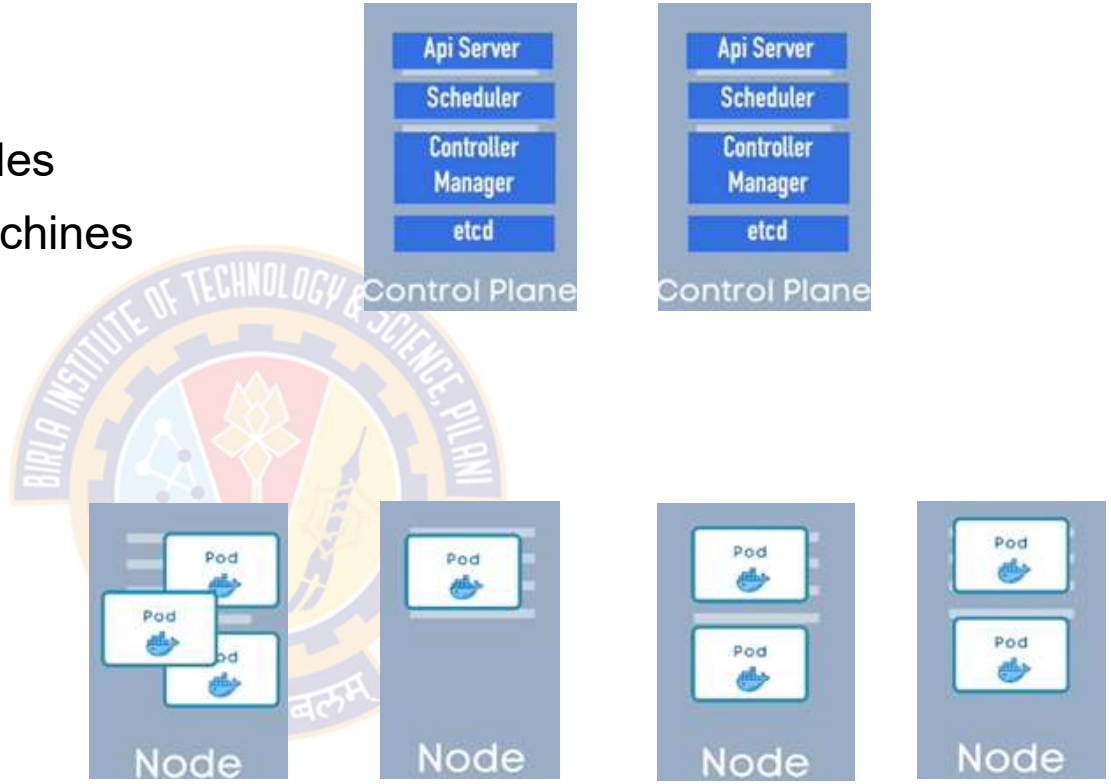
etcd: Kubernetes backing store

Provides the Status information



Production Cluster

- Multiple Master and Worker nodes
- Separate Virtual or Physical machines



Test/Local Cluster Setup

- Master and Node processes run on One machine
  - Minikube
  - Micro K8s

mongo-configmap.yaml and mongo-secret.yaml

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

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

Mongo.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo-deployment
  labels:
    app: mongo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo
  template:
    metadata:
      labels:
        app: mongo
    spec:
      containers:
      - name: mongodb
        image: mongo:5.0
        ports:
        - containerPort: 27017
```

```yaml
env:
      - name: MONGO_INITDB_ROOT_USERNAME
        valueFrom:
          secretKeyRef:
            name: mongo-secret
            key: mongo-user
      - name: MONGO_INITDB_ROOT_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mongo-secret
            key: mongo-password
---
apiVersion: v1
kind: Service
metadata:
  name: mongo-service
spec:
  selector:
    app: mongo
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
```

Webapp.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp-deployment
  labels:
    app: webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: webapp
        image: nanajanashia/k8s-demo-app:v1.0
        ports:
        - containerPort: 3000
```

```
env:
        - name: USER_NAME
          valueFrom:
            secretKeyRef:
              name: mongo-secret
              key: mongo-user
        - name: USER_PWD
          valueFrom:
            secretKeyRef:
              name: mongo-secret
              key: mongo-password
        - name: DB_URL
          valueFrom:
            configMapKeyRef:
              name: mongo-config
              key: mongo-url
---
apiVersion: v1
kind: Service
metadata:
  name: webapp-service
spec:
  type: NodePort
  selector:
    app: webapp
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
      nodePort: 30100
```

Use private IP of any of the nodes to access the app!!

- Kubectl get node
- Kubectl apply –f mongo-configmap.yaml
- Like wise apply – mongo-secret.yaml, mongo.yaml, webapp.yaml
- Kubectl get deployments
- Kubectl get pods
- Kubectl get services
- Kubectl get all
- Kubectl describe service webapp-service

- Kubectl delete service <service-name>
- Kubectl delete deployment <deployment-name>
- Kubectl delete configmap <configmap name>
- Kubectl delete secret <secret-name>