Cloud and Machine Learning

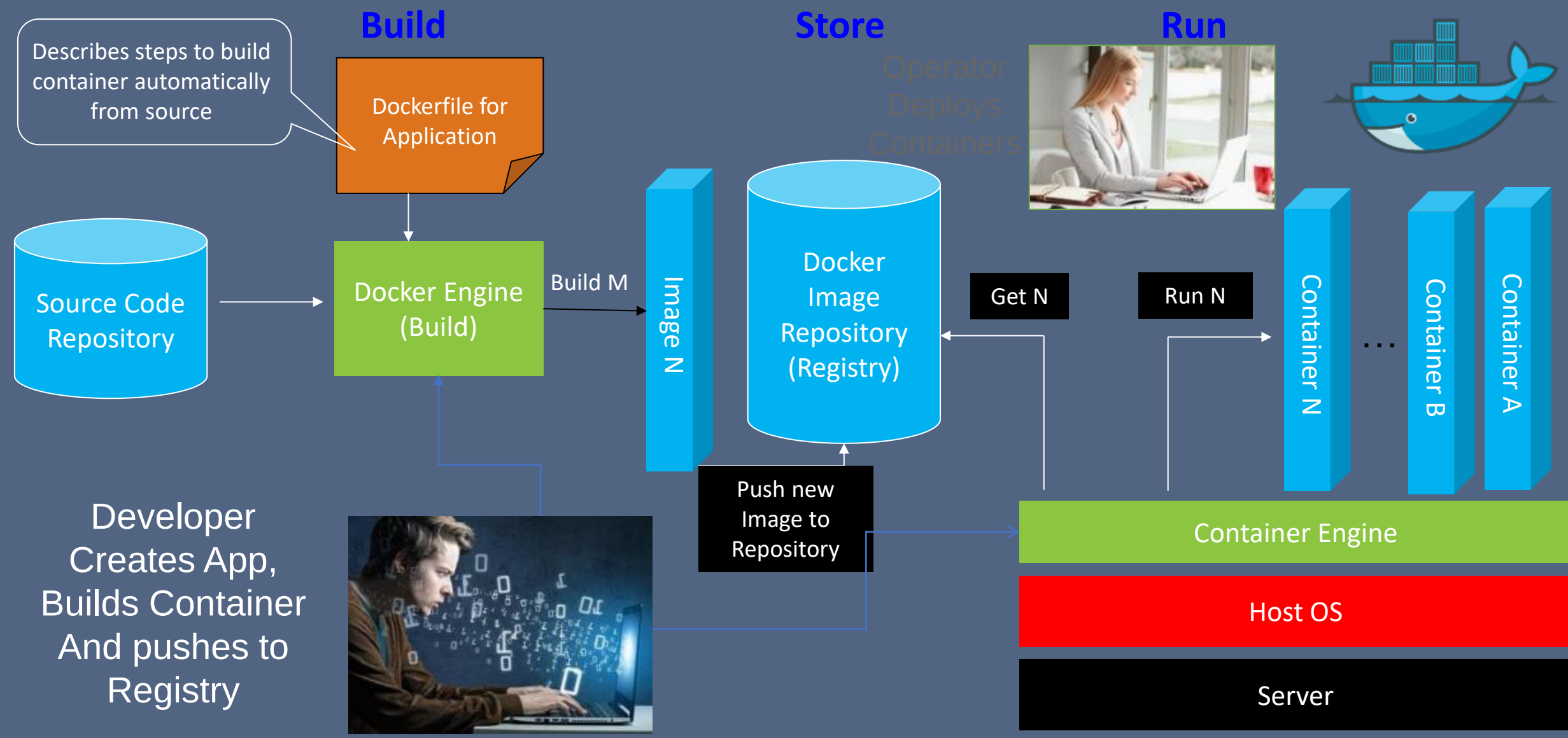
CSCI-GA.3033-085 Fall 2024

Prof. Hao Yu

Prof. I-Hsin Chung

Lecture 8: Kubernetes

What are the Basic Functions of Containers



Docker MNIST HW – graded

- **Work:** Run MNIST inside a Docker container
  - MNIST can be based on PyTorch or TenserFlow.
  - We provided an example code set.
- **Submission:**
  - Source code, files containing your own work or being modified by you (35 points)
    - Vagrantfile (If need to use Vagrant for your experiment)
    - Dockerfile (must include one or a few changes to the example, i.e. mnist/main.py command line options)
    - On screen output (partial capture) of the 'docker run'.
    - Other downloaded files with you modifications
  - Report (55 points)
    - Document your steps so we can reproduce them. These should include a workflow, observations with attention to details, map to learnt concepts, code comment or explanation, and brief comparative discuss.
    - The purpose of the report is show that you can not only do the operation, but have conceptual build-up of container technologies.
  - **Singularity** (10 points):
    - Run MNIST inside a Singularity container (inside a Vagrant VM if need)
  - Compare Docker and Singularity in your report.

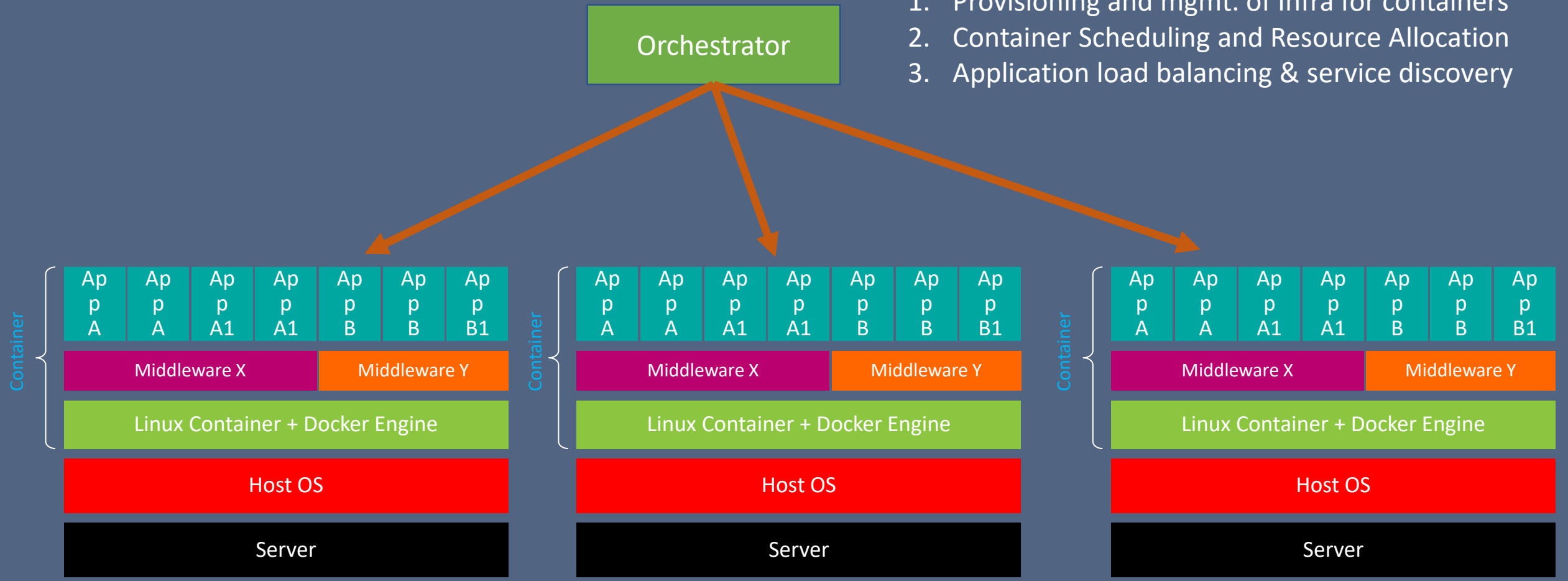
Docker MNIST HW – Examplary

- A possible work flow:
  - `git clone https://github.com/yuhaohaoyu/ibmcloud-fall-2023.git`
  - `git clone https://github.com/pytorch/examples.git`
  - `cd ibmcloud-fall-2023; cp -r ../examples/mnist .`
  - `vi Vagrantfile Dockerfile`
  - `vagrant up; vagrant ssh; cd /vagrant;`
  - `docker build -t mnist . ;`
  - `docker run -it mnist 2>&1 | tee docker-run.out`
- Submission:
  - Vagrantfile (only if need), Dockerfile, docker-run.out, mnist/main.py file if there is change.

- References:
  - Docker hands-on slides.
  - Vagrant and Docker hands-on code: https://github.com/pytorch/examples.git
  - Install docker on laptop:
    - Install docker with administrator privilege
      - https://www.geeksforgeeks.org/how-to-install-docker-on-windows/
    - Install docker in a virtual machine with root privilege: e.g. Vagrant-VM
      - https://docs.docker.com/engine/install/ubuntu/

What is container orchestration?

1. Provisioning and mgmt. of Infra for containers
2. Container Scheduling and Resource Allocation
3. Application load balancing & service discovery

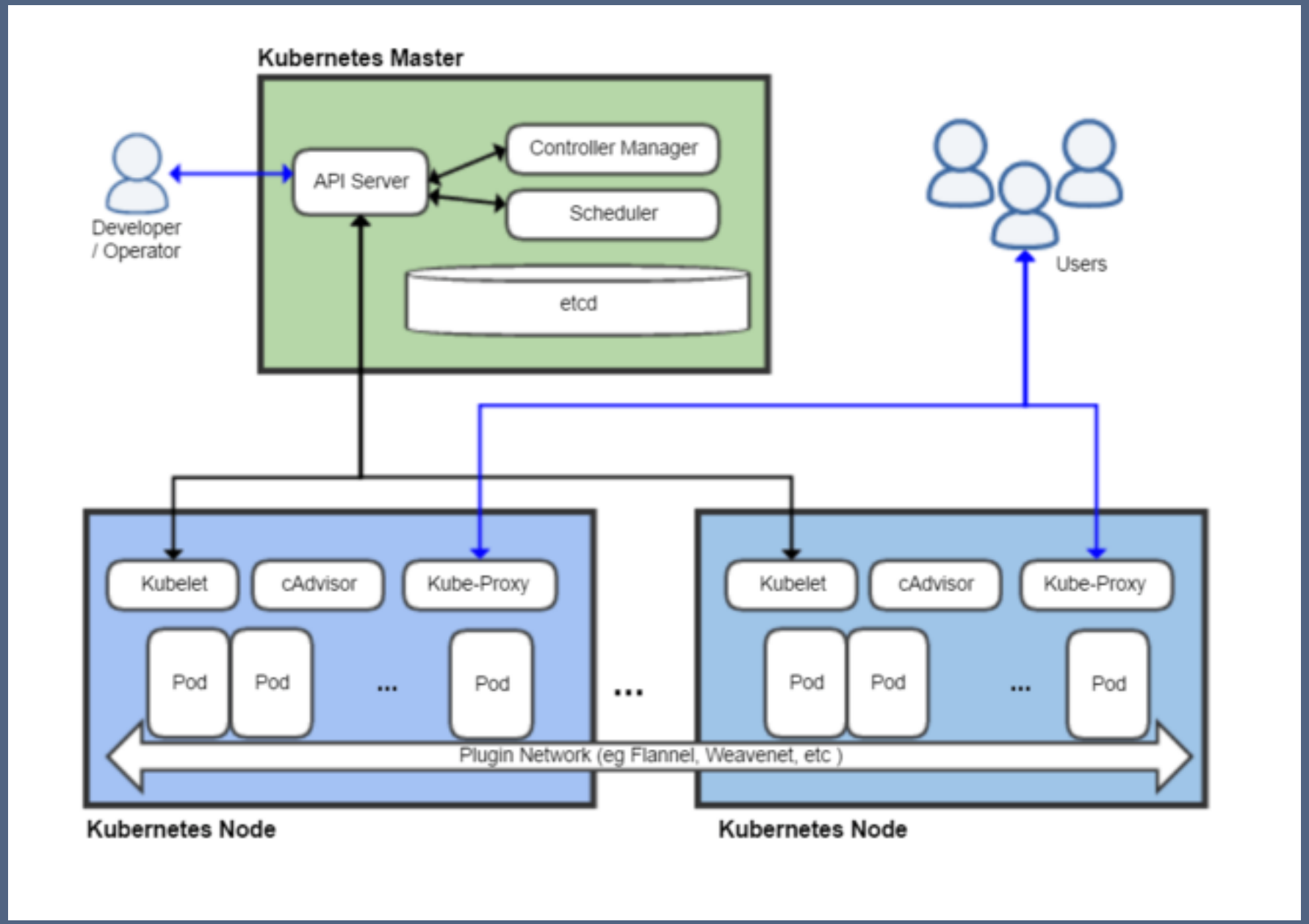


Container orchestrators

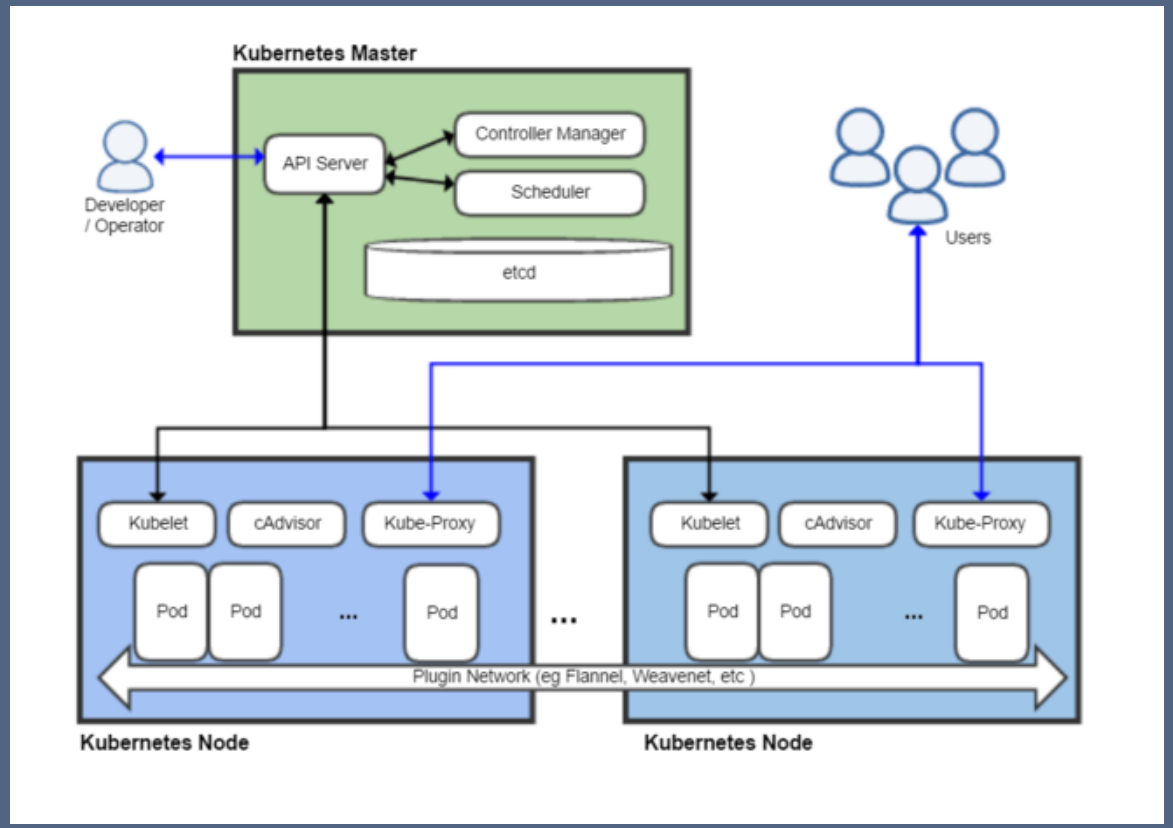
- Apache Mesos & DCOS
- Mesos Marathon
- Docker Swarm
- Docker Compose
- CoreOS Fleet
- Tupperware (twine)
- OpenShift
- Kubernetes

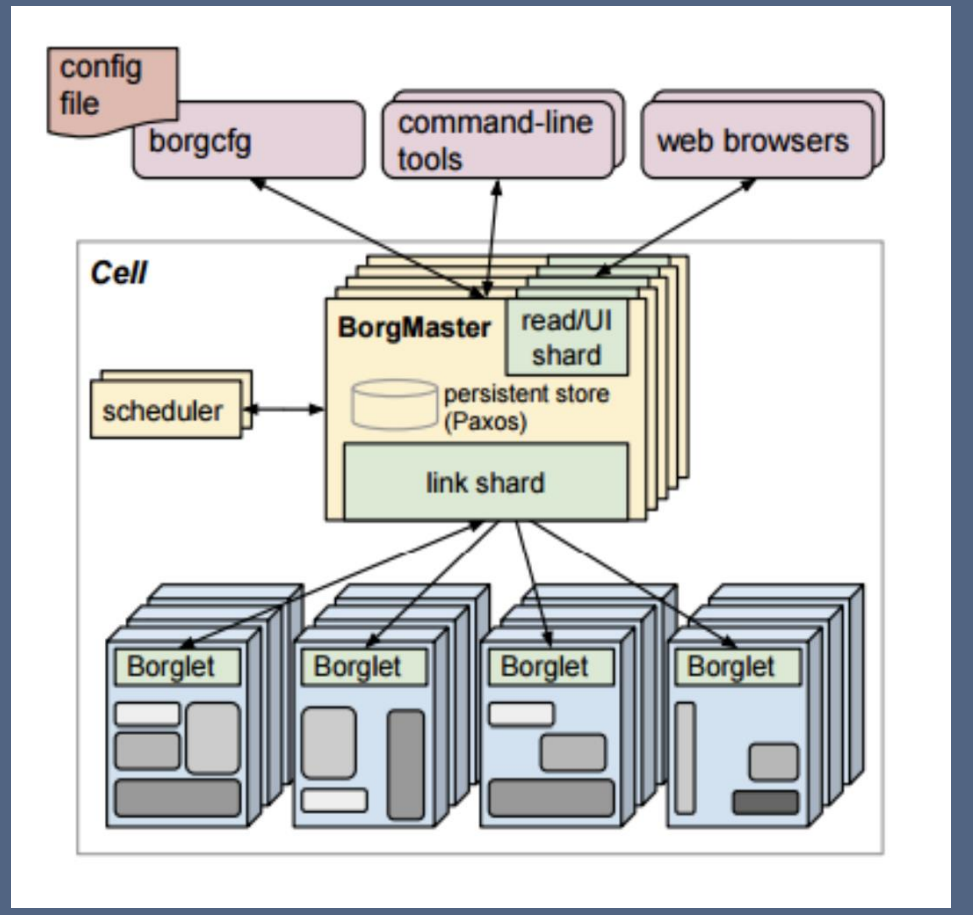
What is Kubernetes?

- “Kubernetes is a portable, extensible open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.”
  - From: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/


Kubernetes Master
API Server
Controller Manager
Scheduler
etcd
Developer
/ Operator
Users
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
...
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
Plugin Network (eg Flannel, Weavenet, etc)
Kubernetes Node
Kubernetes Node

Kubernetes vs Borg

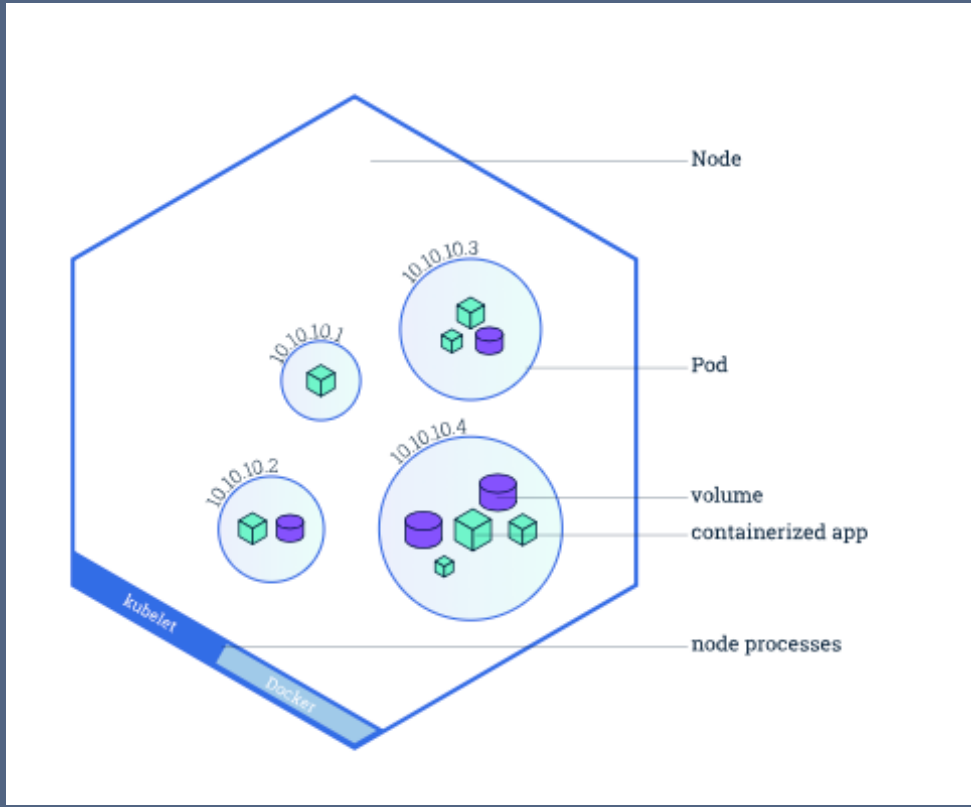




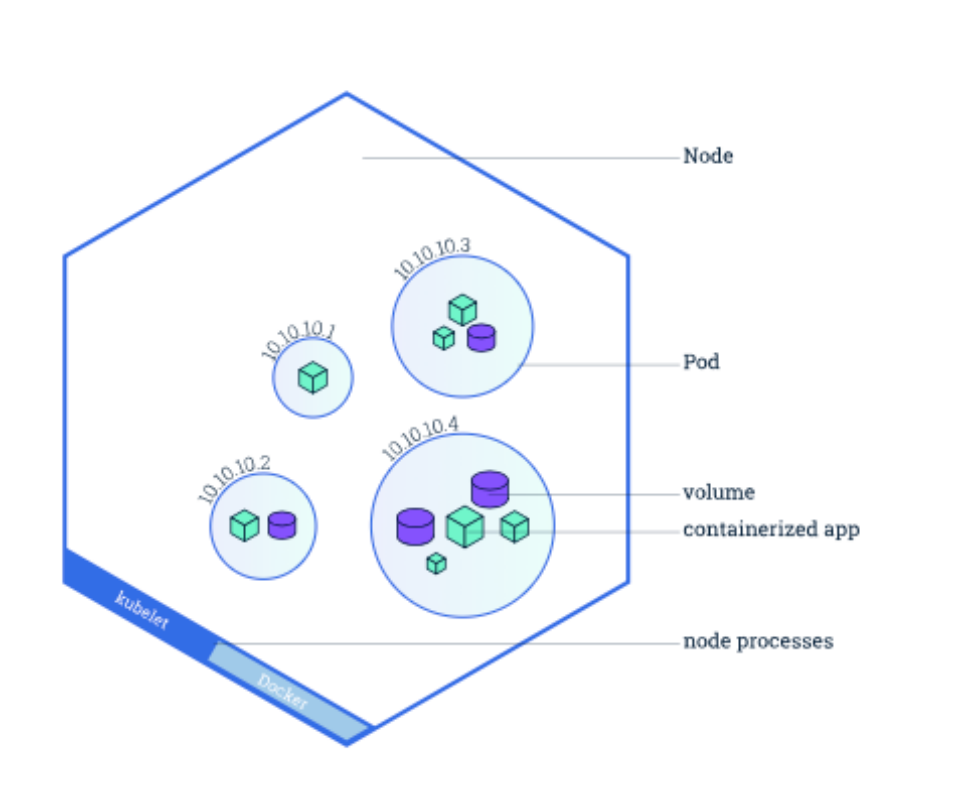
Kubernetes basic objects and controllers

- Basic Kubernetes objects
  - **Pod**
  - **Service**
  - **Volume**
  - Namespace
- Kubernetes Machinery
  - **Kube API Server**
  - **Kube Scheduler**
  - **Controller-manager**
  - **Kubelet**
  - **Kube-proxy**
- Controllers
  - **ReplicaSet**
  - **Deployment**
  - StatefulSet
  - DaemonSet
  - Job

Nodes & Pods


Node
10.10.10.3
10.10.10.1
Pod
10.10.10.4
10.10.10.2
volume
containerized app
kubelet
Docker
node processes

Nodes & Pods

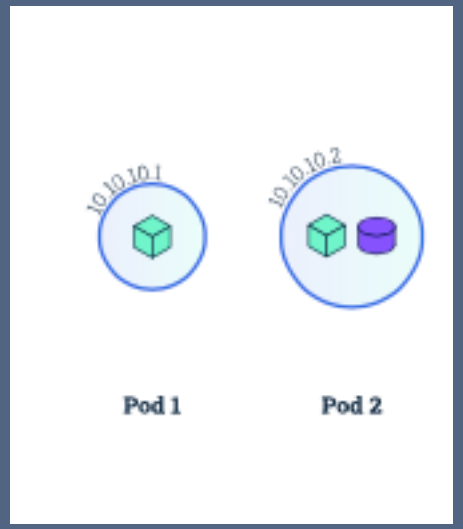


Node:

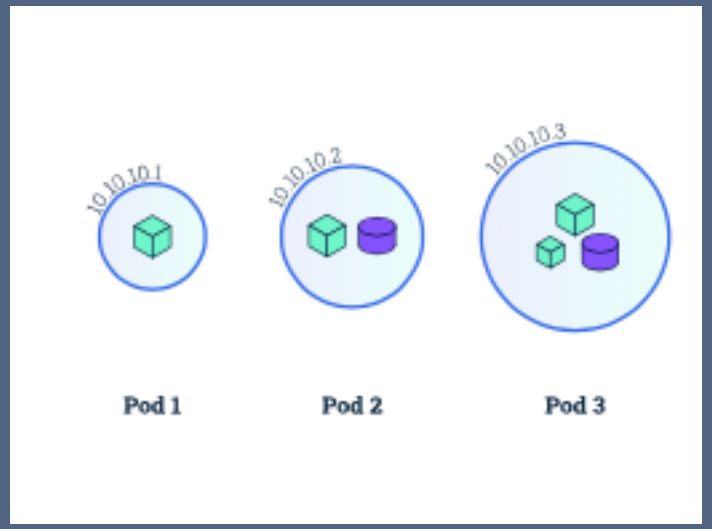
1. Kubelet (communication between master → Node)
2. Container runtime (docker, cri-o, podman)

Nodes & Pods

Nodes & Pods

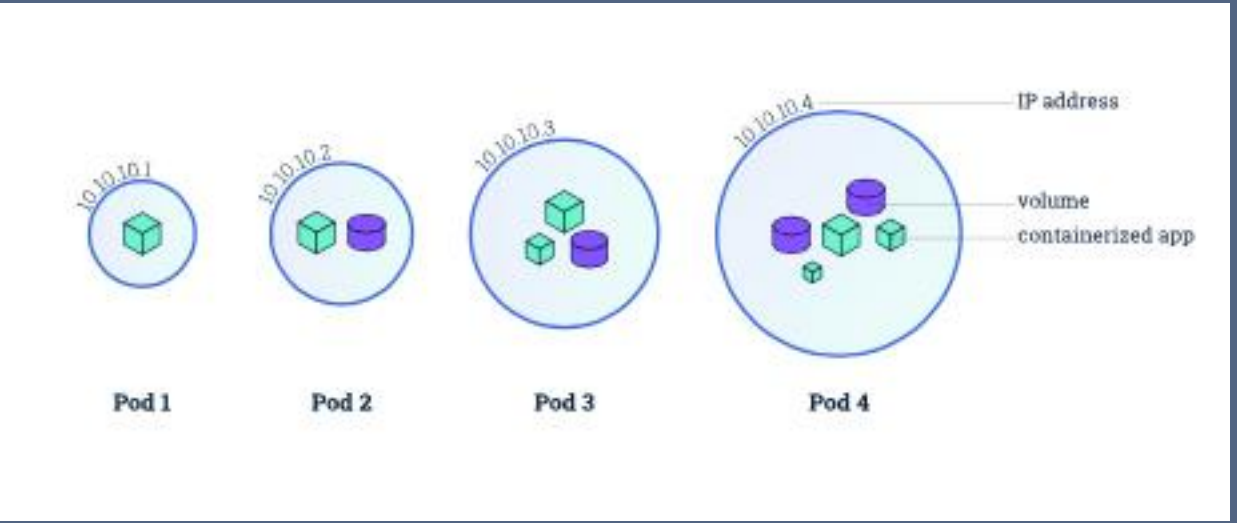


Nodes & Pods

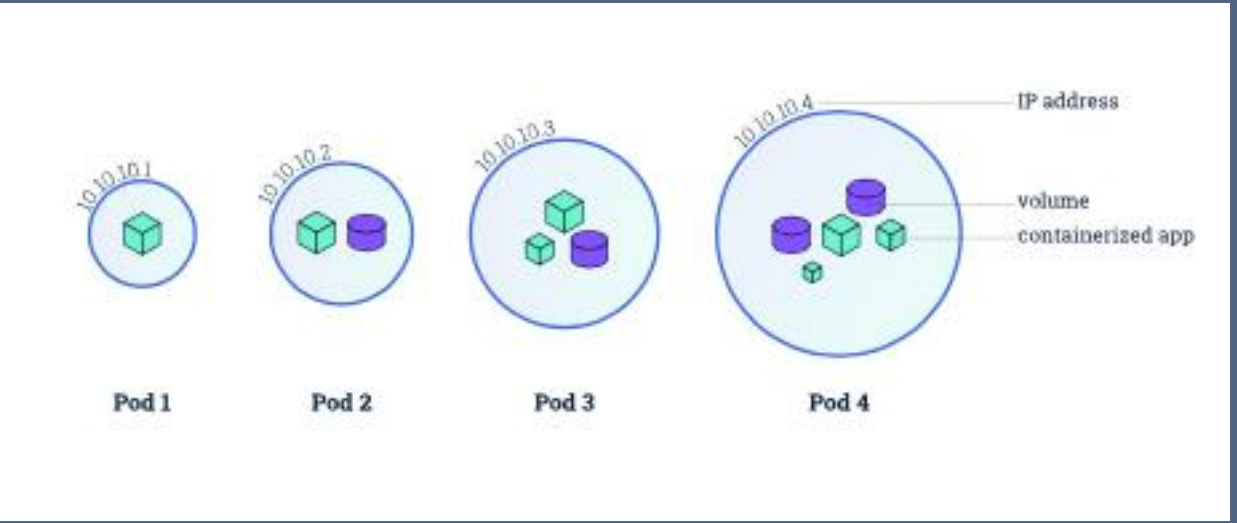


Nodes & Pods

Nodes & Pods


10.10.10.1
10.10.10.2
10.10.10.3
10.10.10.4
IP address
volume
containerized app
Pod 1
Pod 2
Pod 3
Pod 4

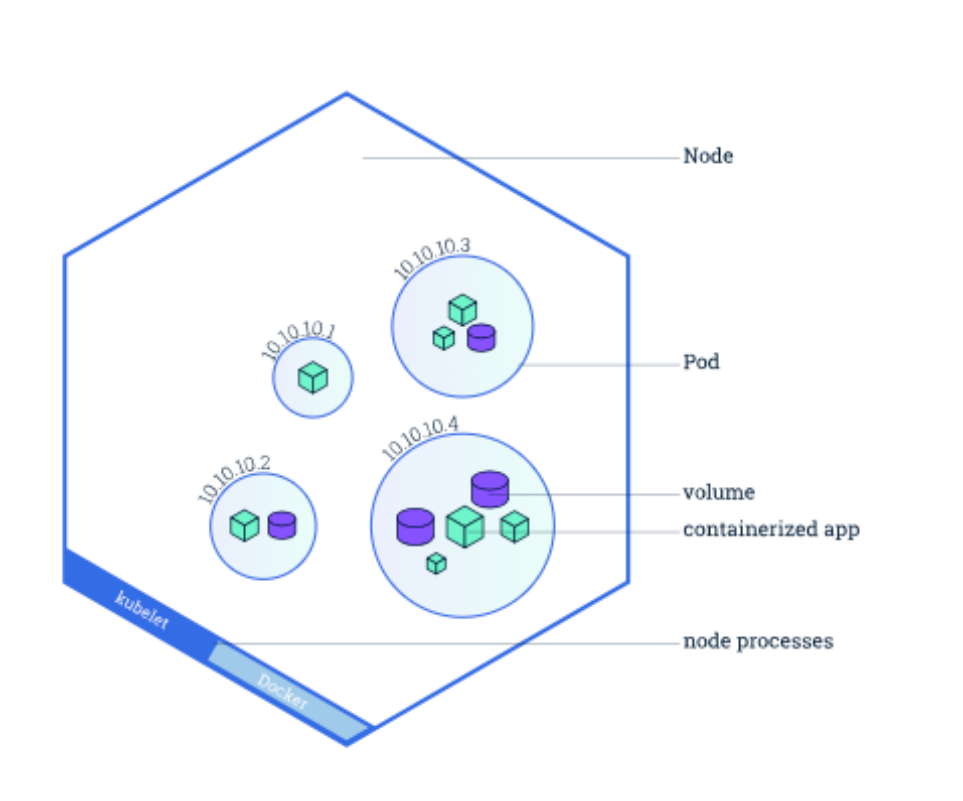
Nodes & Pods

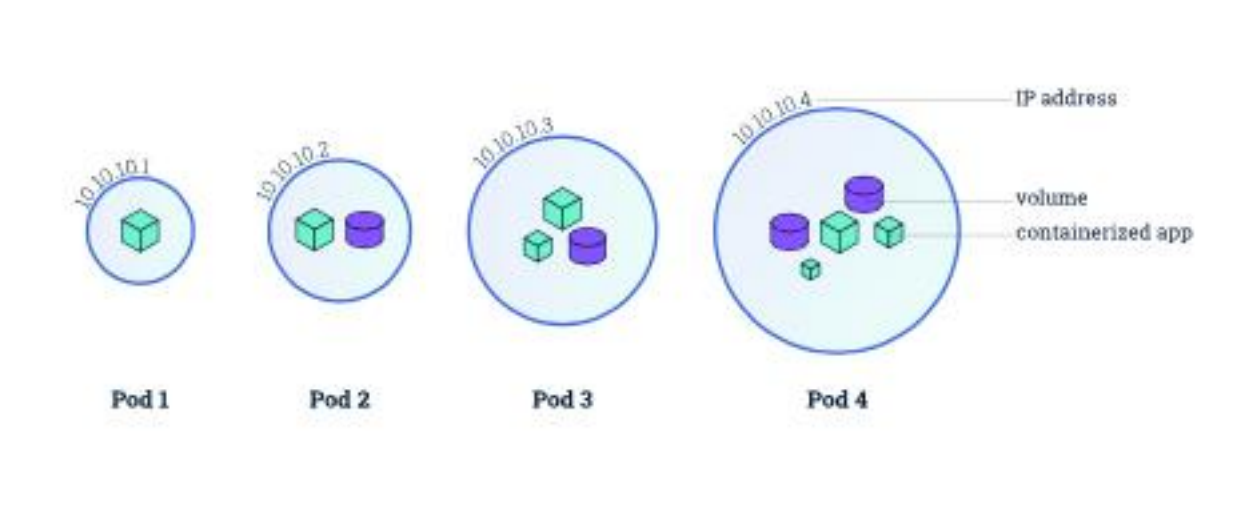


Pod:

1. Containers run in Pods
2. Shared storage
3. Shared network
4. Container behavior (ports, health, resources)
5. All containers of a pod run on the same machine

Nodes & Pods





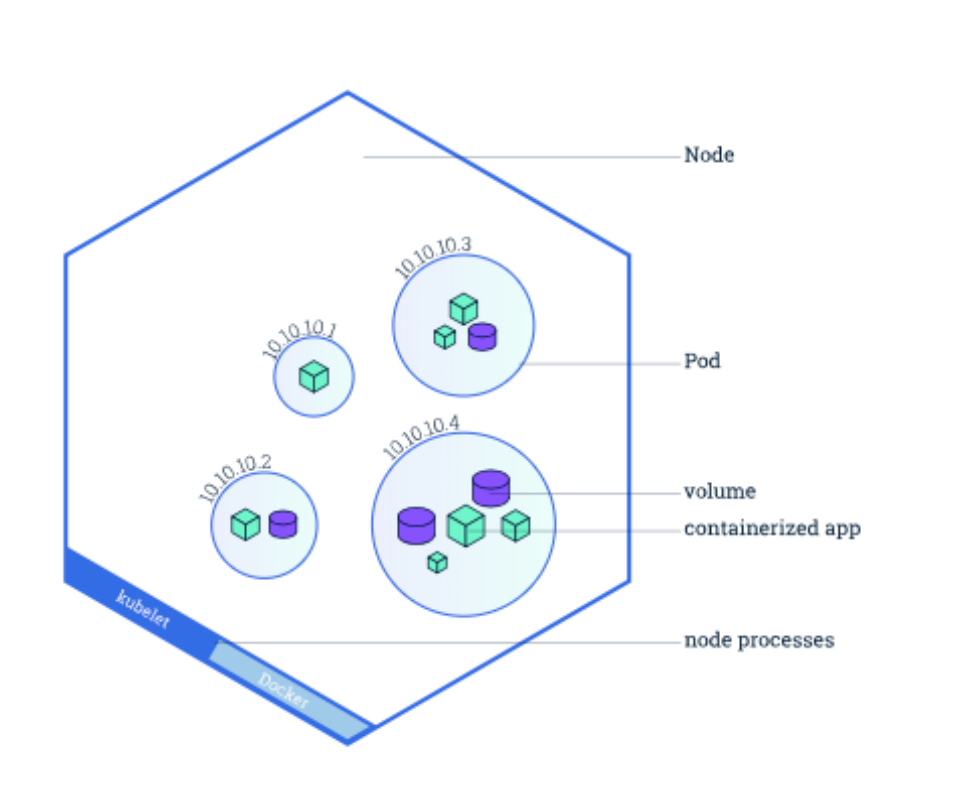
Node:

1. Kubelet (communication between master → Node)
2. Container runtime (docker, rkt)

Pod:

1. Containers run in Pods
2. Shared storage
3. Shared network
4. Container behavior (ports, health, resources)
5. All containers of a pod run on the same machine

K8S Pod vs real PODS



“Hello world” pod spec

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

```
-bash-4.2$ kubectl create -f hello.yaml
pod/myapp-pod created
-bash-4.2$ kubectl get pods
NAME                                          READY     STATUS              RESTARTS   AGE
ibm-cert-manager-cert-manager-768b66977-qp6db 1/1       Running             0          12d
myapp-pod                                     0/1       ContainerCreating   0          5s
test-pd                                       1/1       Running             0          2d10h
-bash-4.2$
```

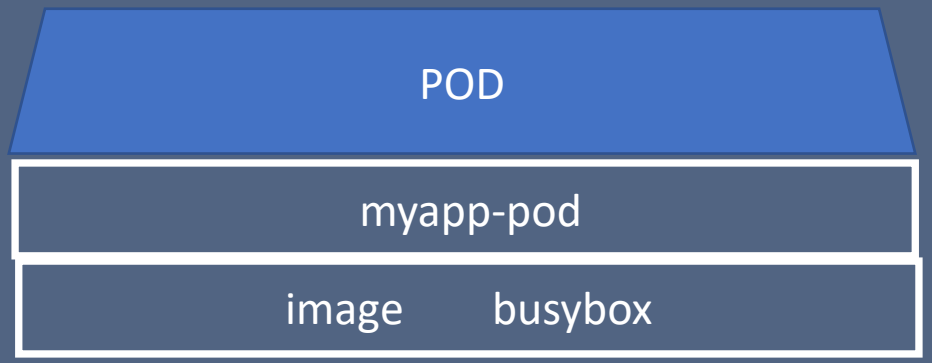
Differences between Docker and K8S

```
docker run busybox /bin/sh -c 'echo hello && sleep 10'
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

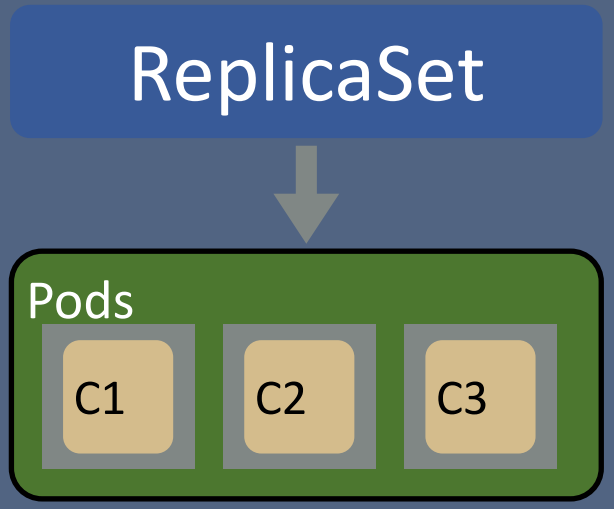
Pod spec to Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```



Kubernetes ReplicaSets

- ReplicaSets allow multiple copies of pods to be deployed
- One or more containers in a pod
- If a container dies, the ReplicaSet will spawn a new one



ReplicaSet

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-rs
  labels:
    app: myapp-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      tier: myapp-rs
  template:
    metadata:
      labels:
        tier: myapp-rs
    spec:
        containers:
        - name: myapp-container
          image: busybox
          command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

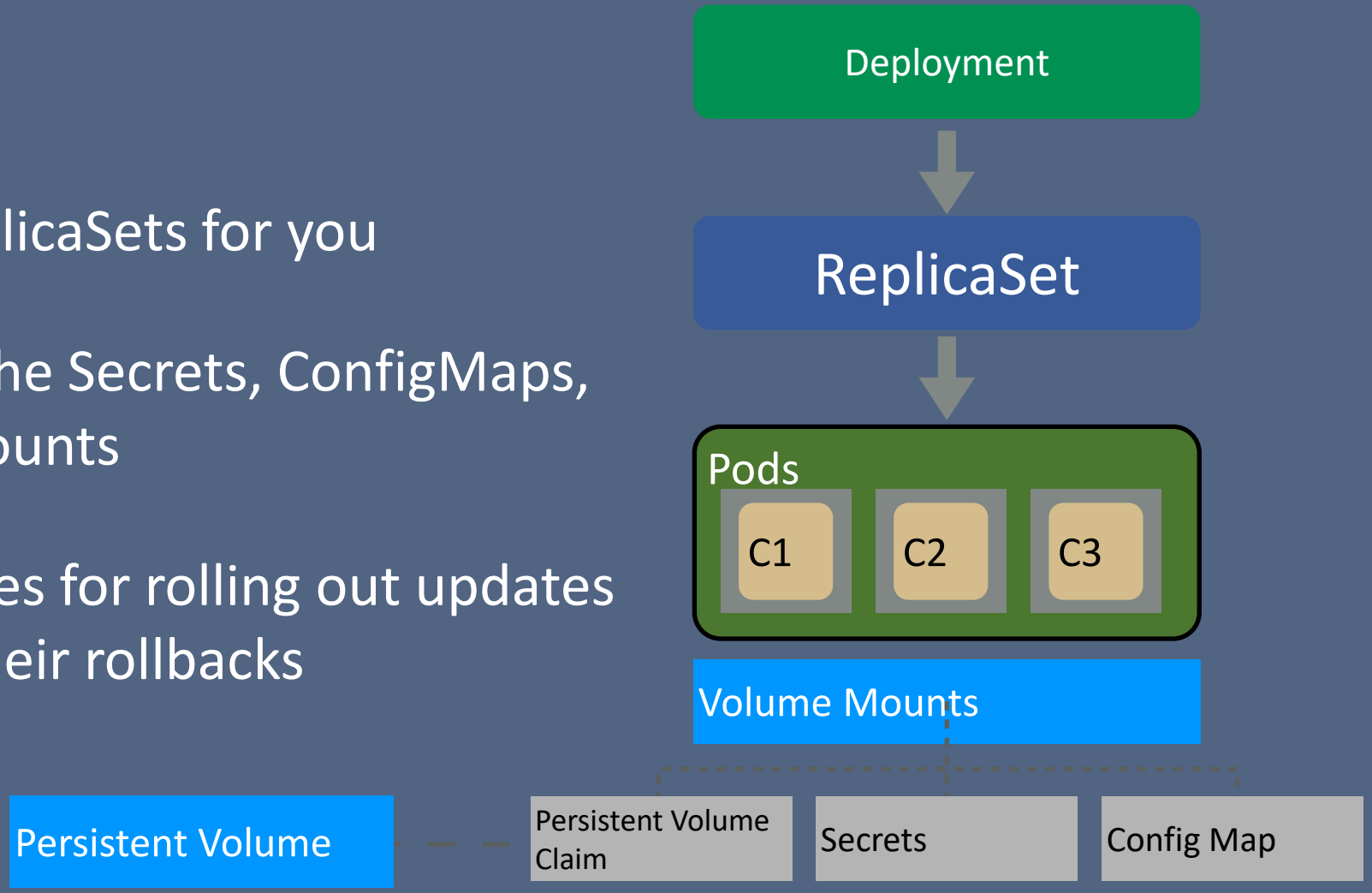
ReplicaSet vs Pod

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-rs
  labels:
    app: myapp-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      tier: myapp-rs
  template:
    metadata:
      labels:
        tier: myapp-rs
    spec:
        containers:
        - name: myapp-container
          image: busybox
          command: ['sh', '-c', 'echo Hello Kubernetes
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 36']
```

Kubernetes Deployments

- Deployment
  - Sets up the ReplicaSets for you
  - Also specified the Secrets, ConfigMaps, and Volume Mounts
  - Provides features for rolling out updates and handling their rollbacks

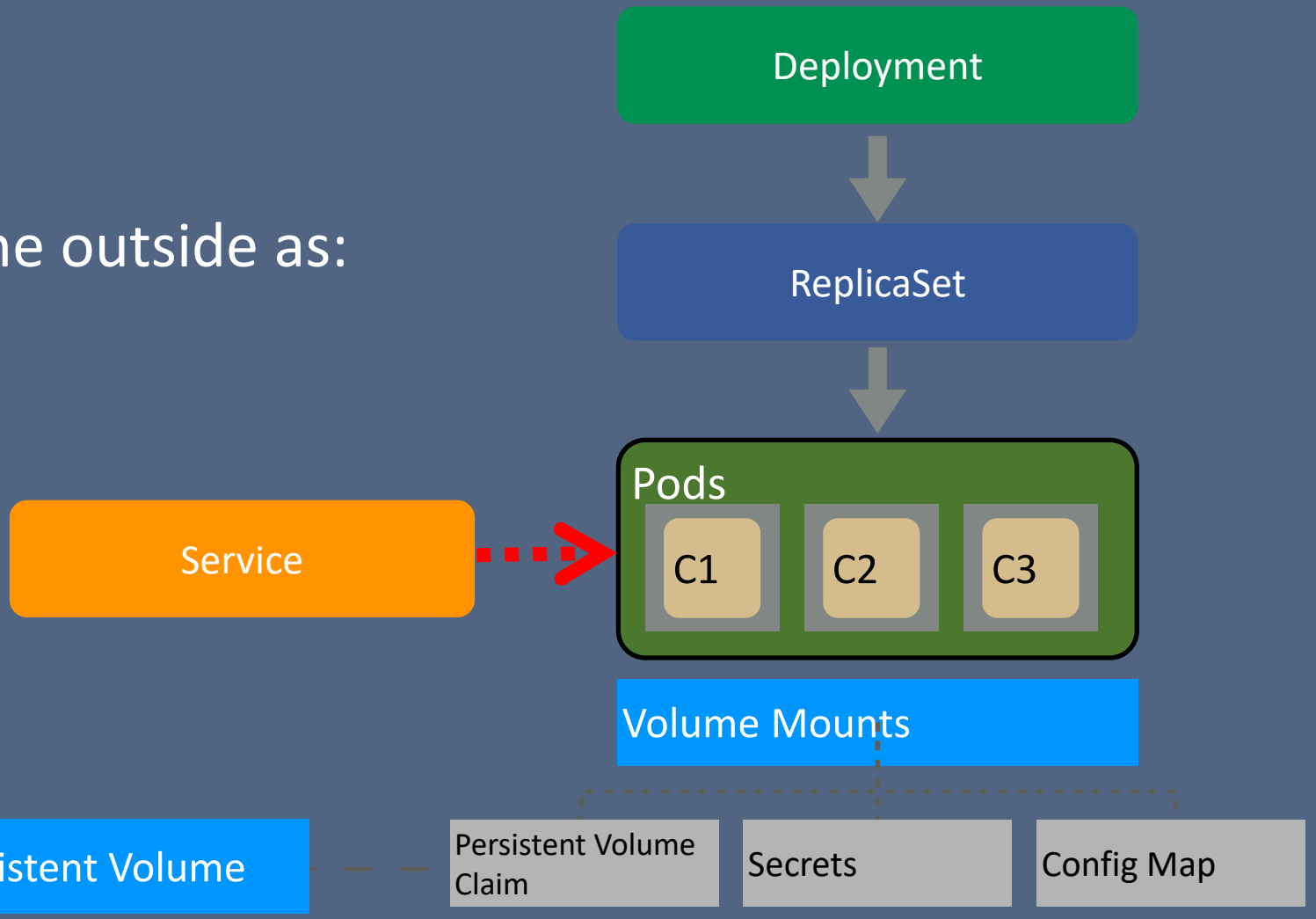


Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-deployment
  labels:
    app: hello-d
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-d
  template:
    metadata:
      labels:
        app: hello-d
    spec:
    spec:
        containers:
        - name: myapp-container
          image: busybox
          command: ['sh', '-c', 'echo Hello Kubernetes! I am updated!! && sleep 3600']
```

Kubernetes Service

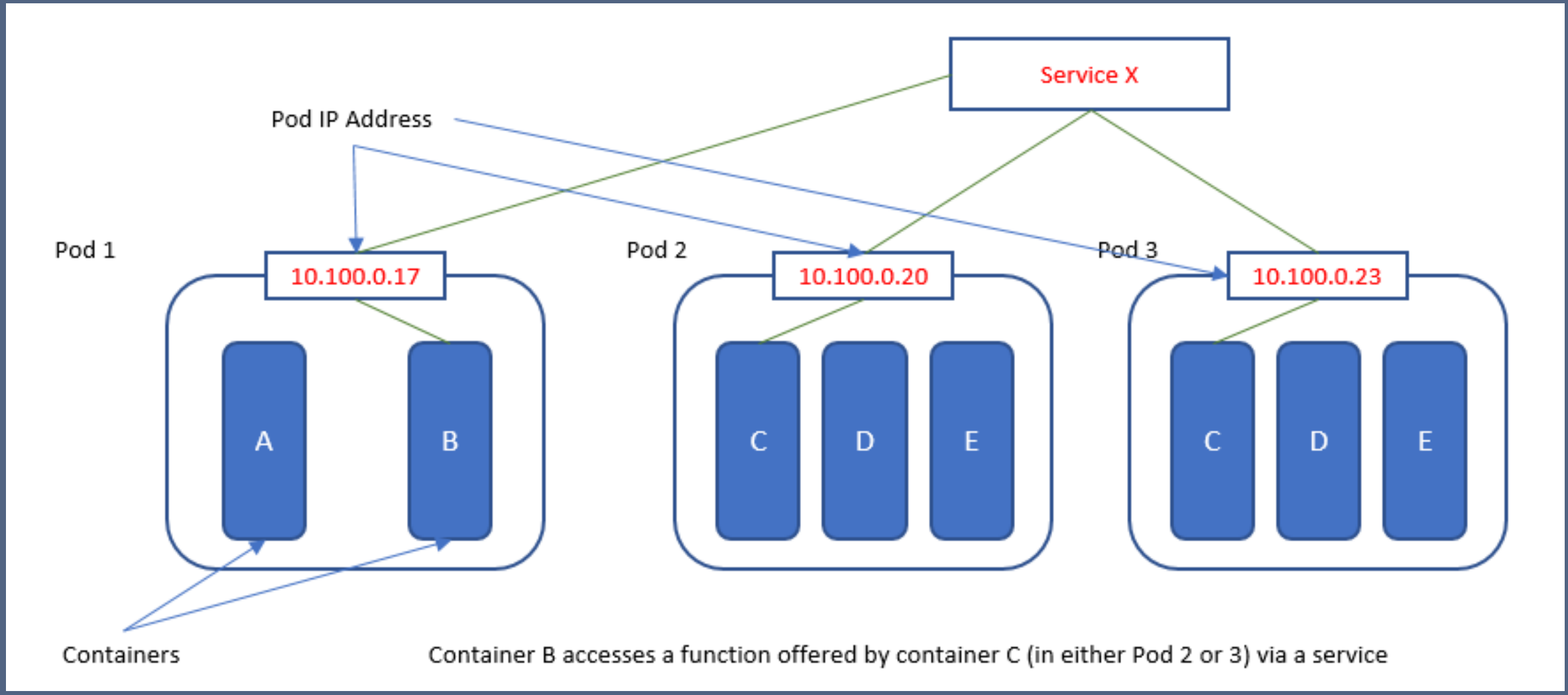
- Service
  - Exposes Pods to the outside as:
    - ClusterIP
    - NodePort
    - Load Balancer



Service Example

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: hello-d
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

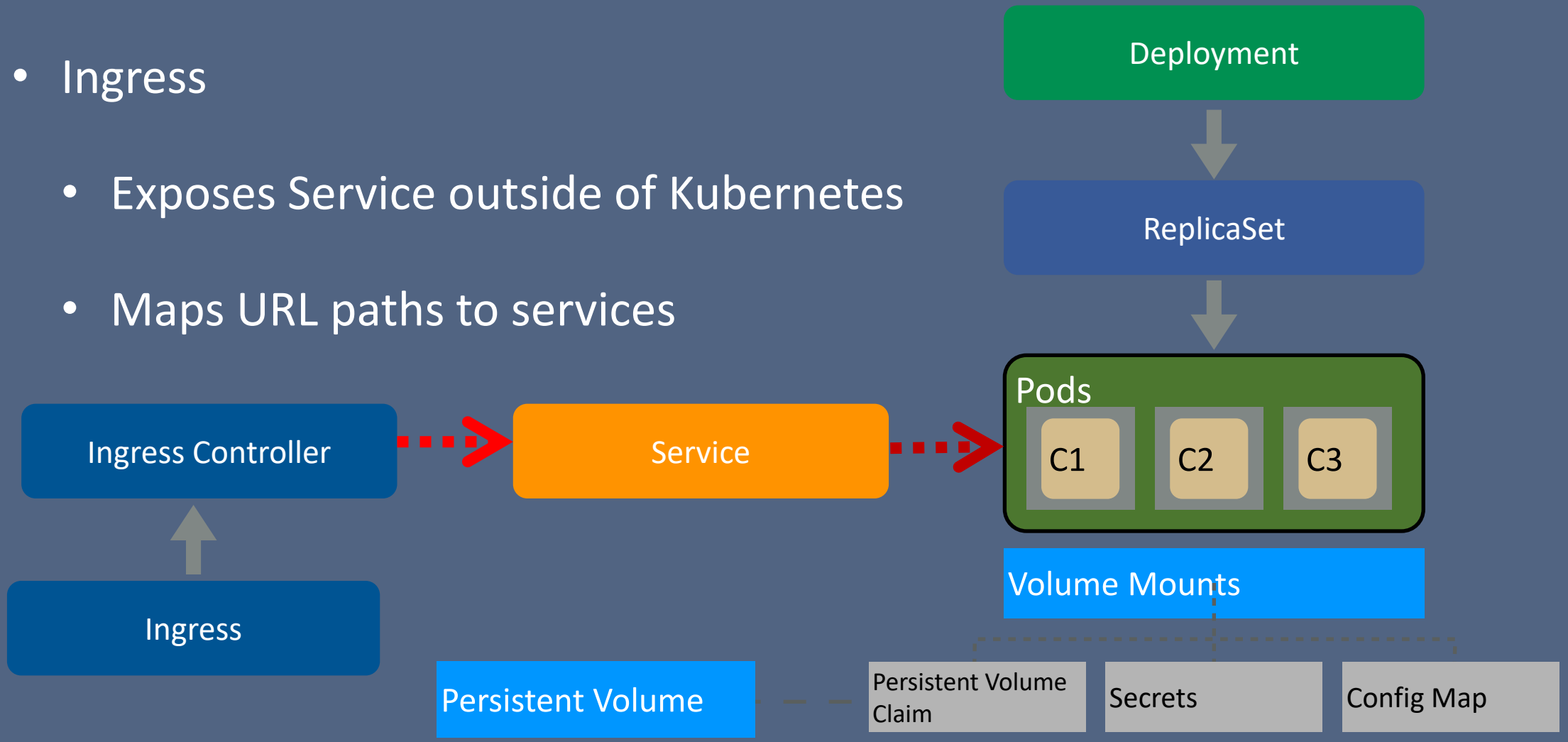
Kubernetes Service



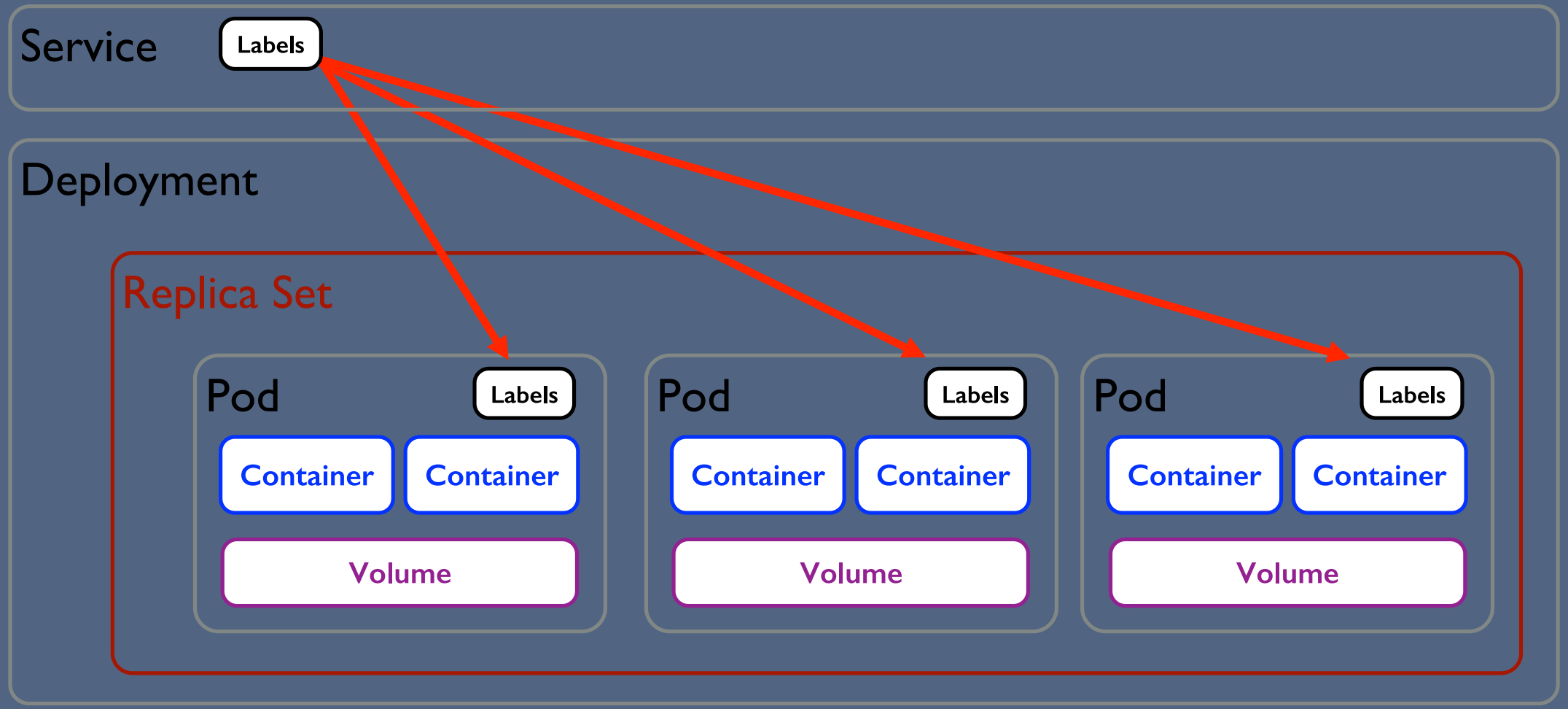
https://en.wikipedia.org/wiki/Kubernetes

Kubernetes Deployment Model

- Ingress
  - Exposes Service outside of Kubernetes
  - Maps URL paths to services

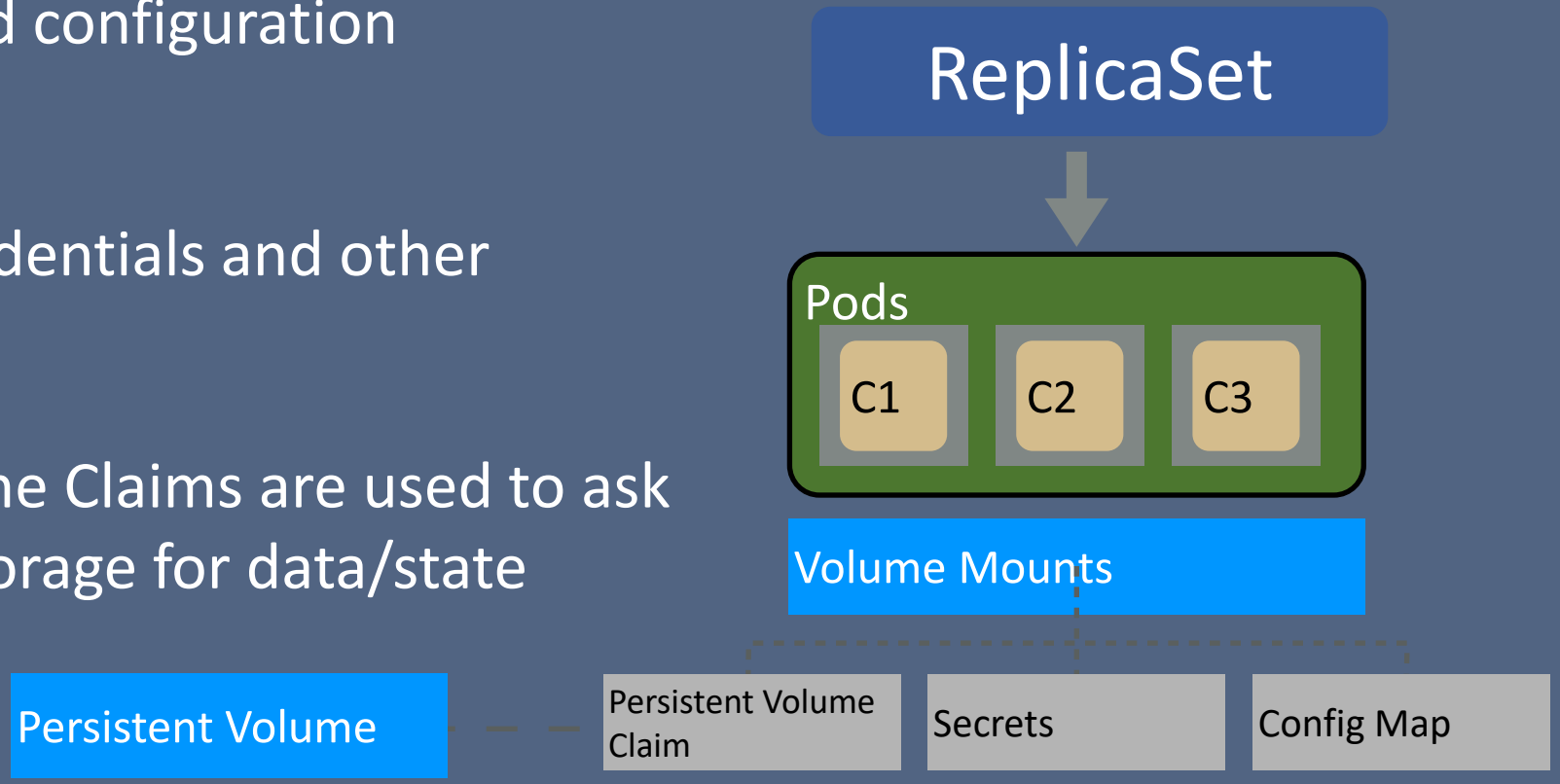


Kuberntes Logical View: Putting all together



Kubernetes Volume Mounts

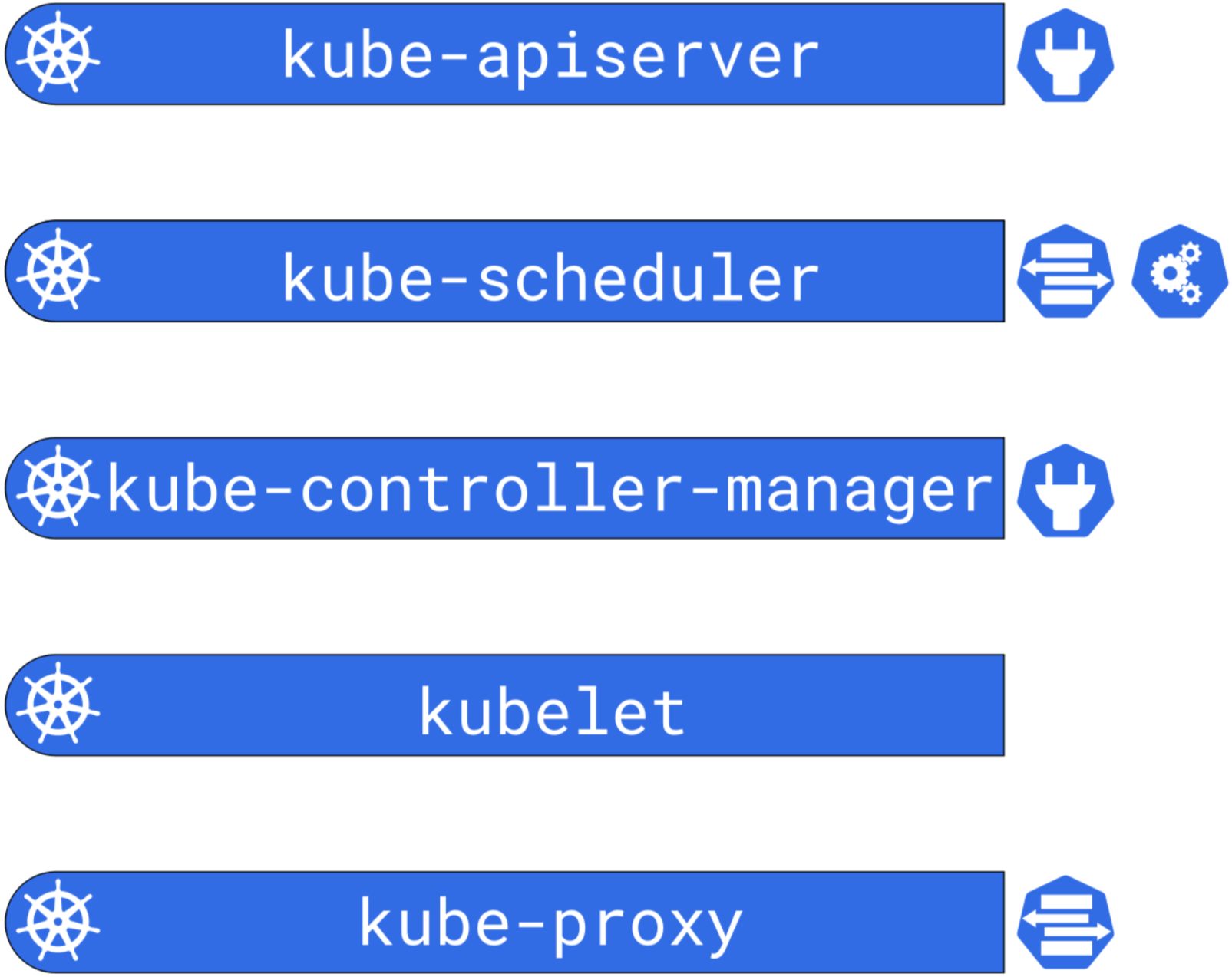
- Volumes
  - ConfigMaps hold configuration parameters
  - Secrets hold credentials and other secrets
  - Persistent Volume Claims are used to ask for persistent storage for data/state

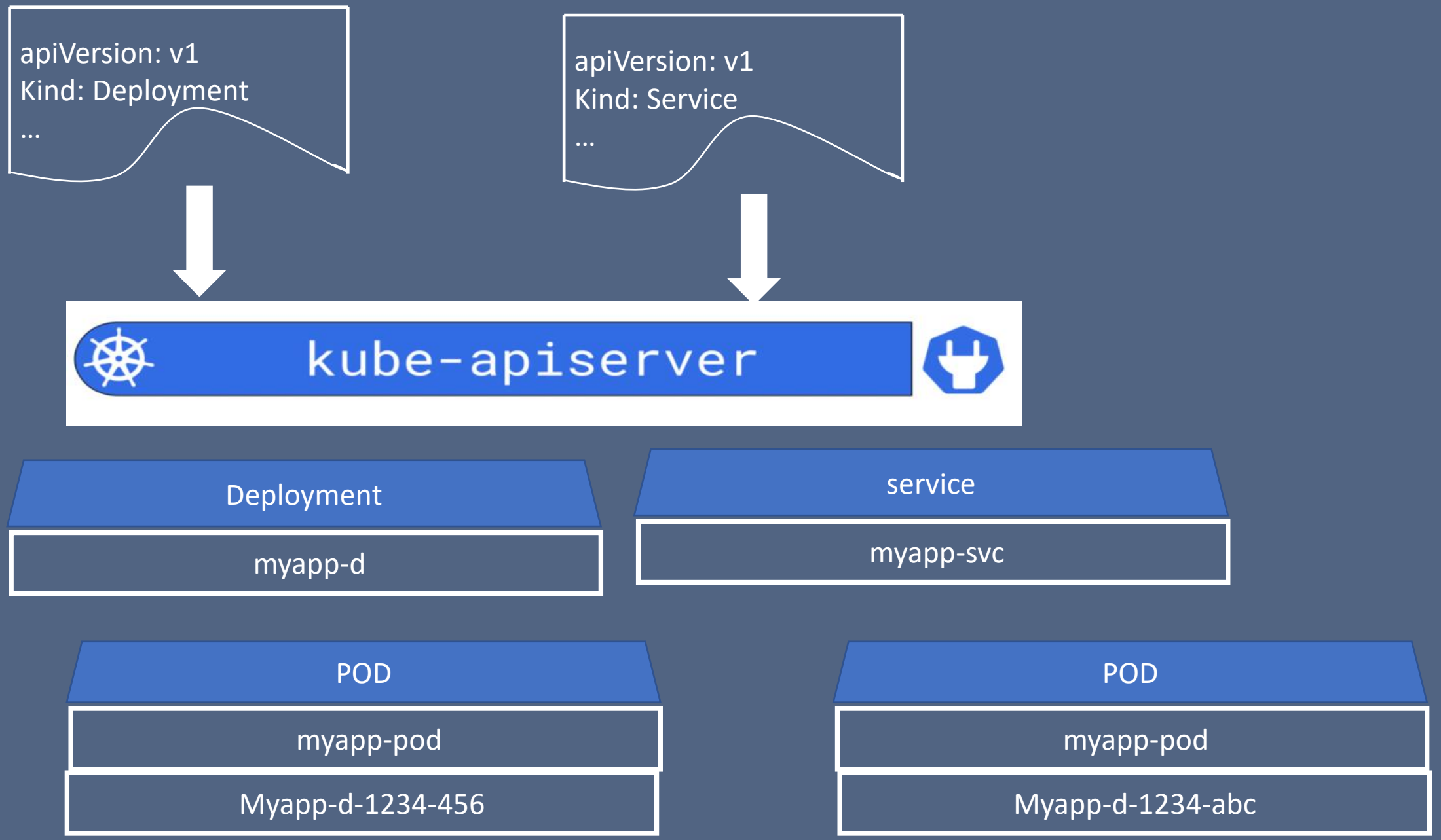


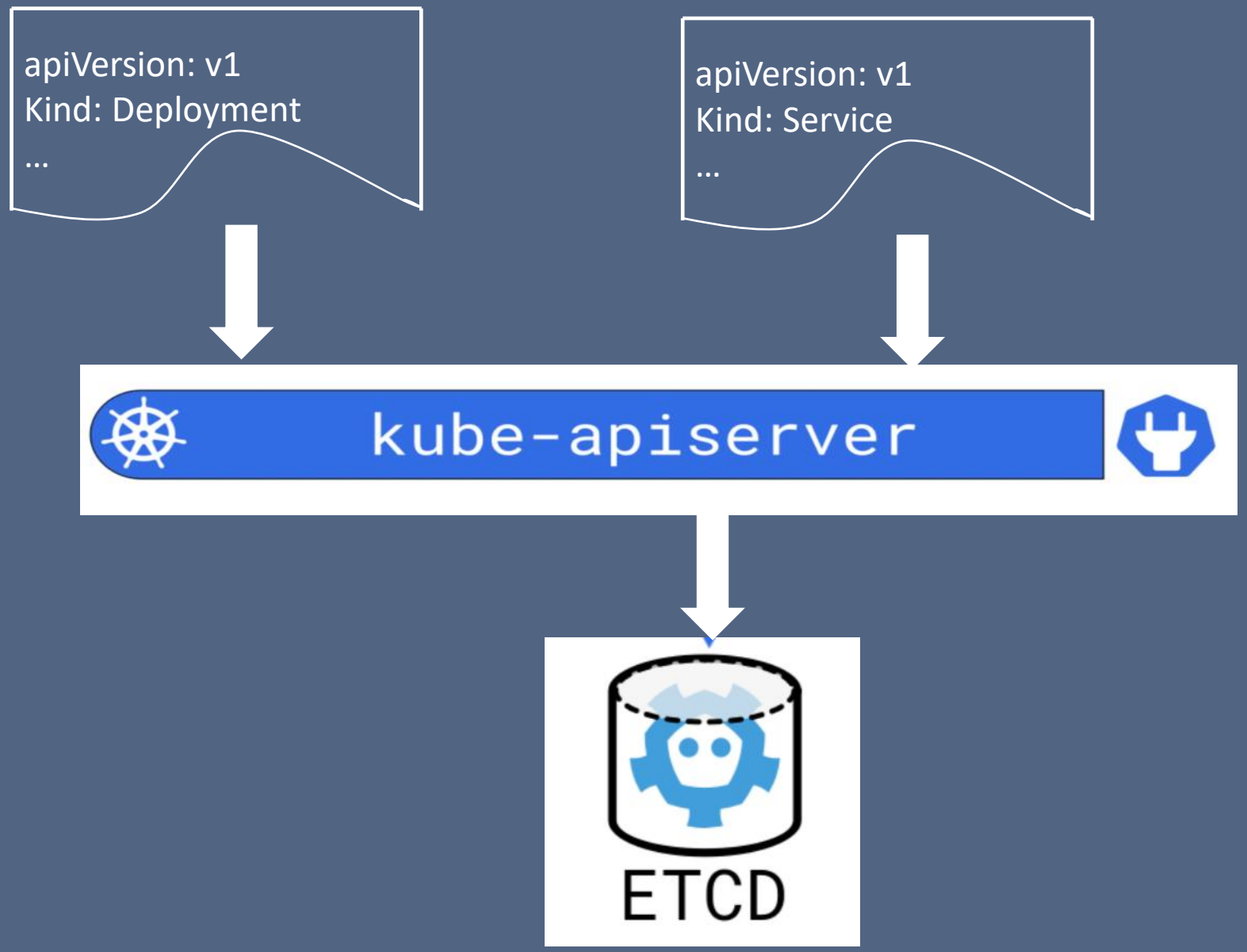
Kubernetes machinery

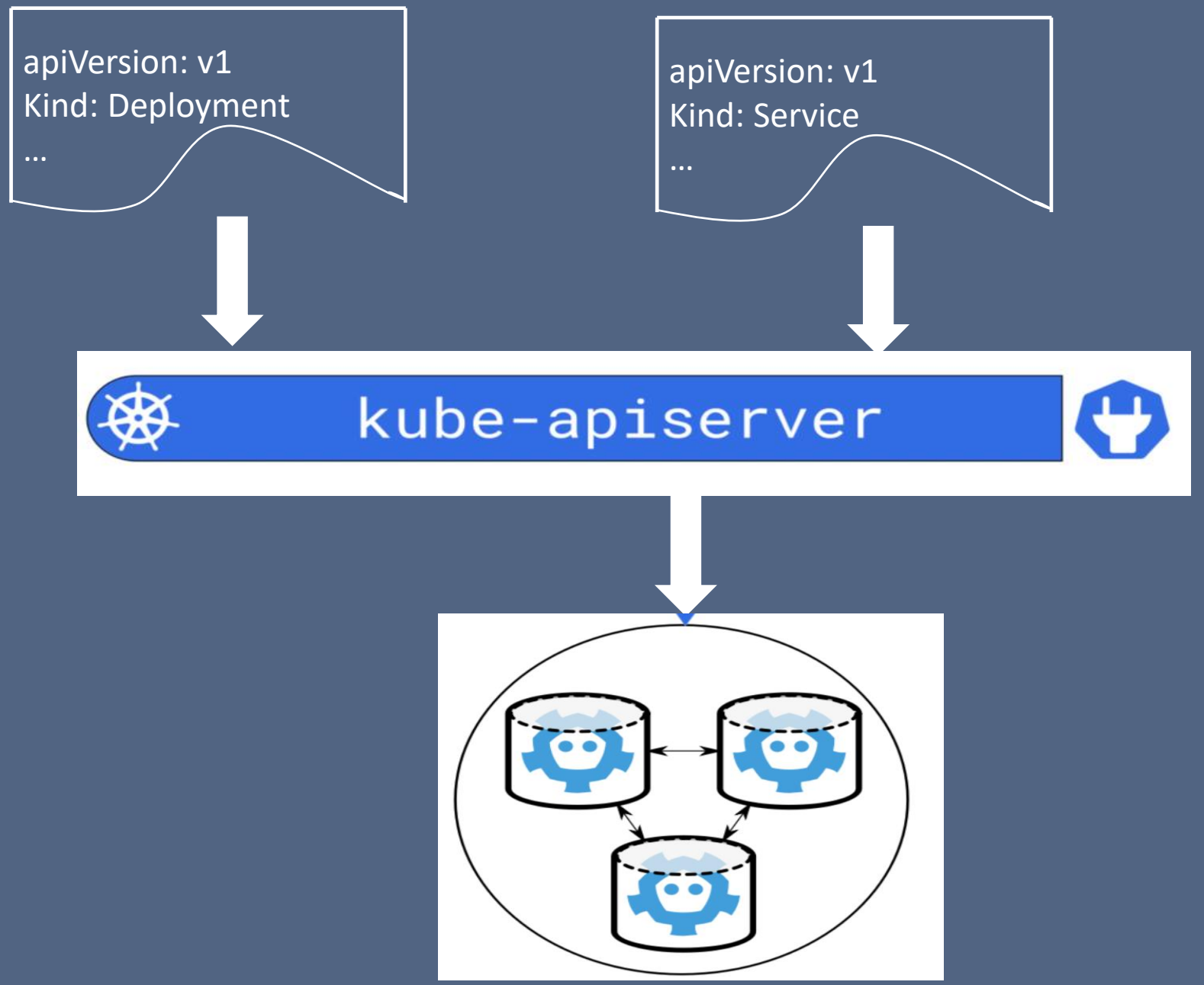
- **Kube API Server**
- **Kube Scheduler**
- **Controller-manager**
- Kubelet
- Kube-proxy


Replacement
Plugin
Config

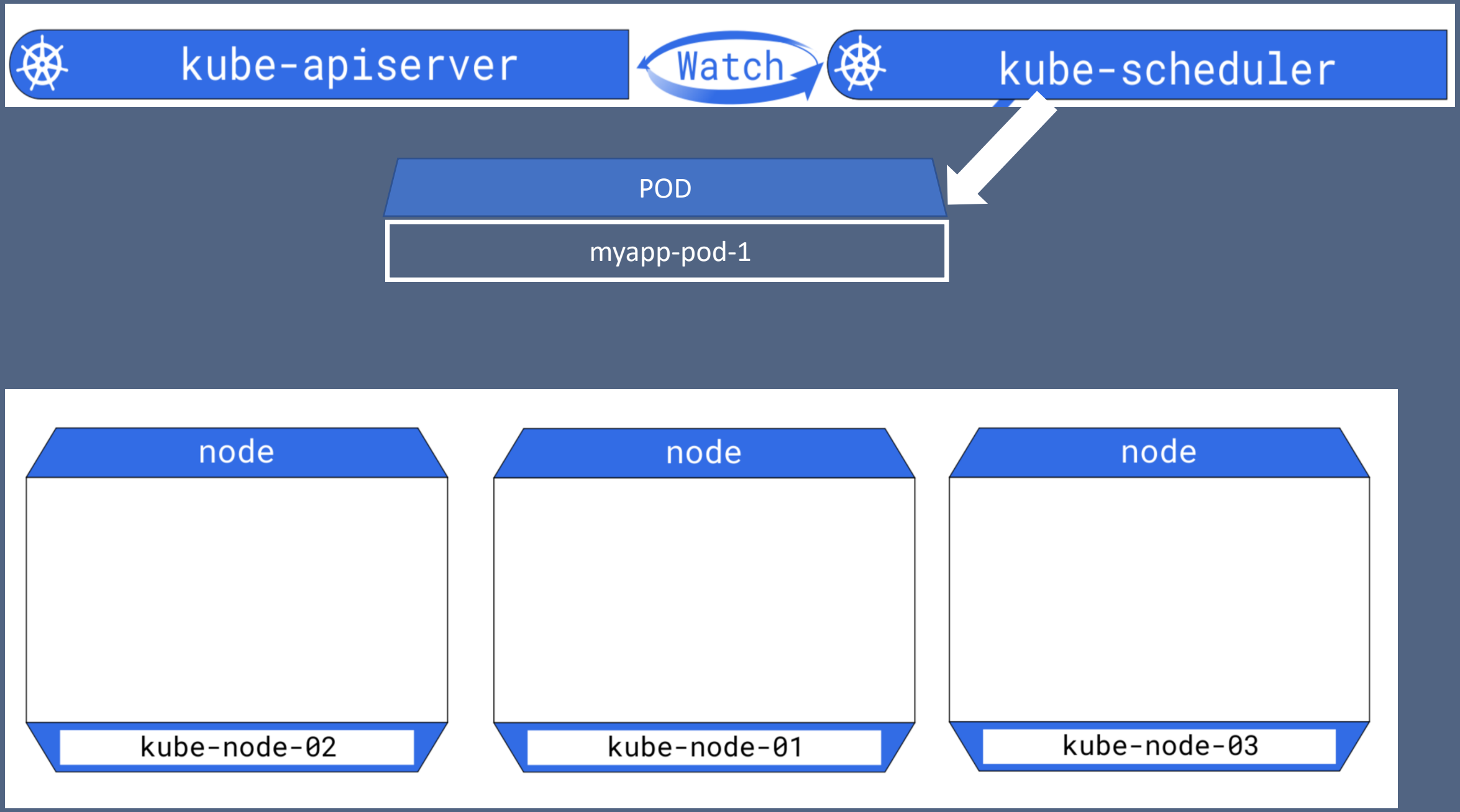

kube-apiserver
kube-scheduler
kube-controller-manager
kubelet
kube-proxy

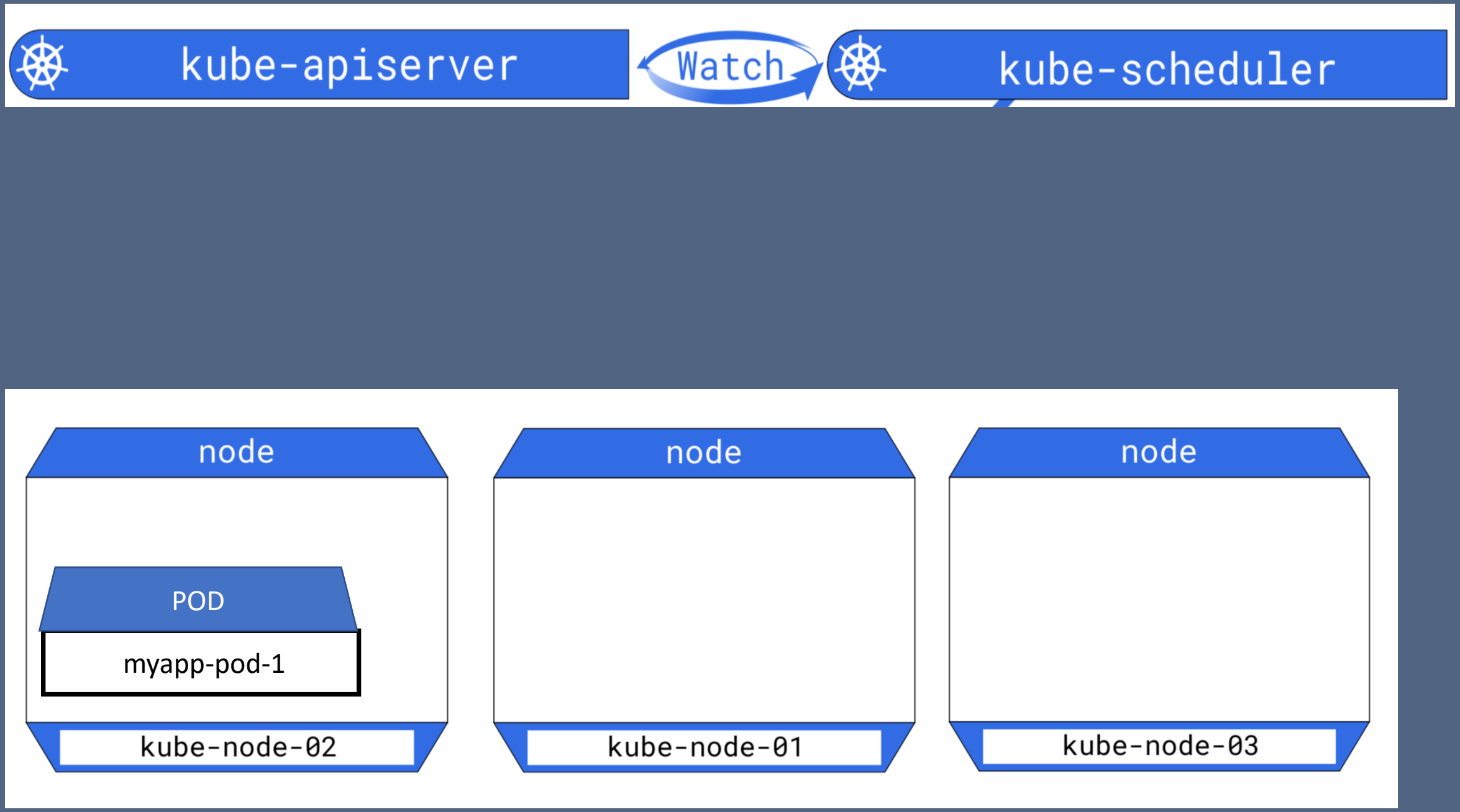

apiVersion: v1
Kind: Deployment
…
apiVersion: v1
Kind: Service
…
kube-apiserver
Deployment
myapp-d
service
myapp-svc
POD
myapp-pod
Myapp-d-1234-456
POD
myapp-pod
Myapp-d-1234-abc

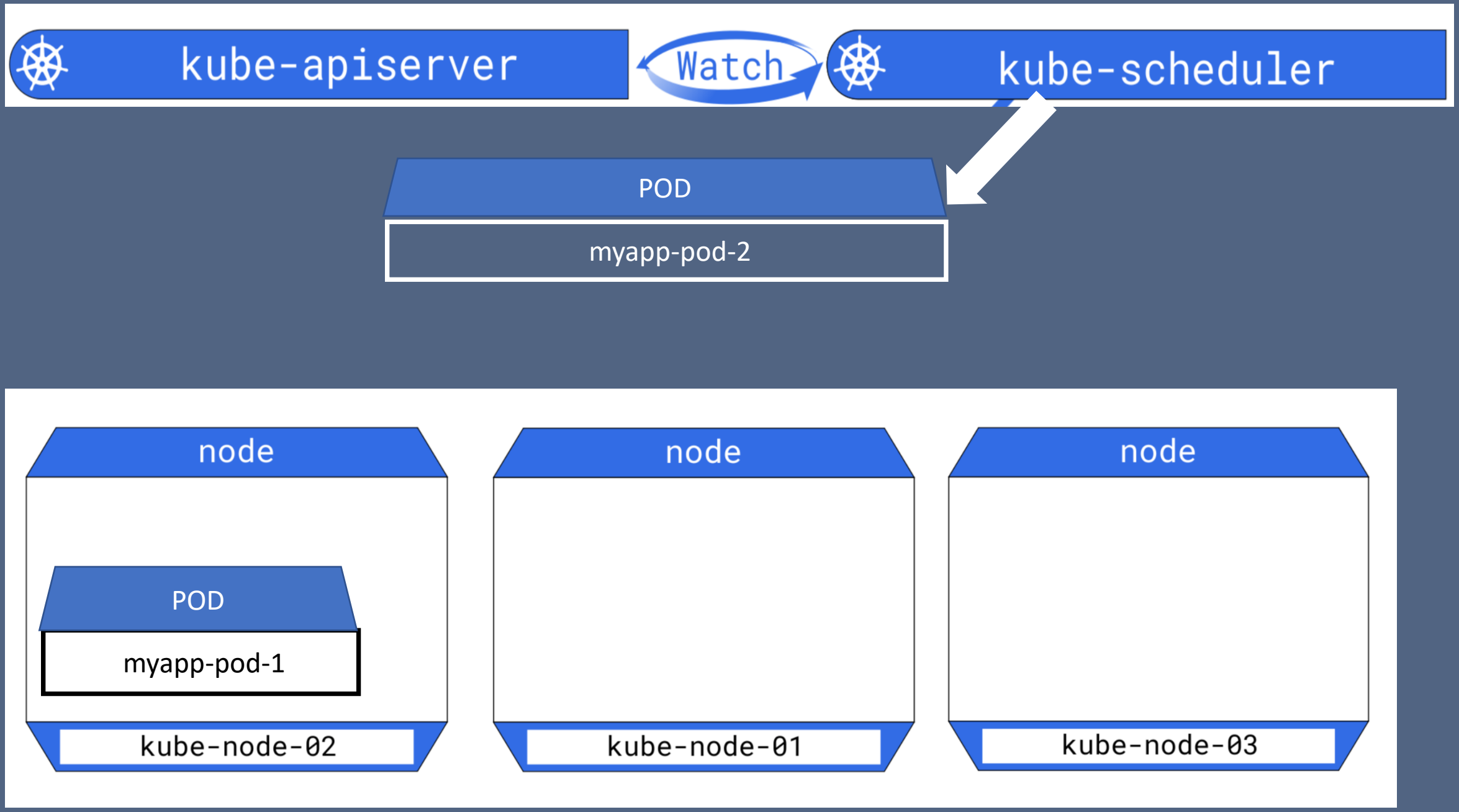

apiVersion: v1
Kind: Deployment
…
apiVersion: v1
Kind: Service
…
kube-apiserver
ETCD

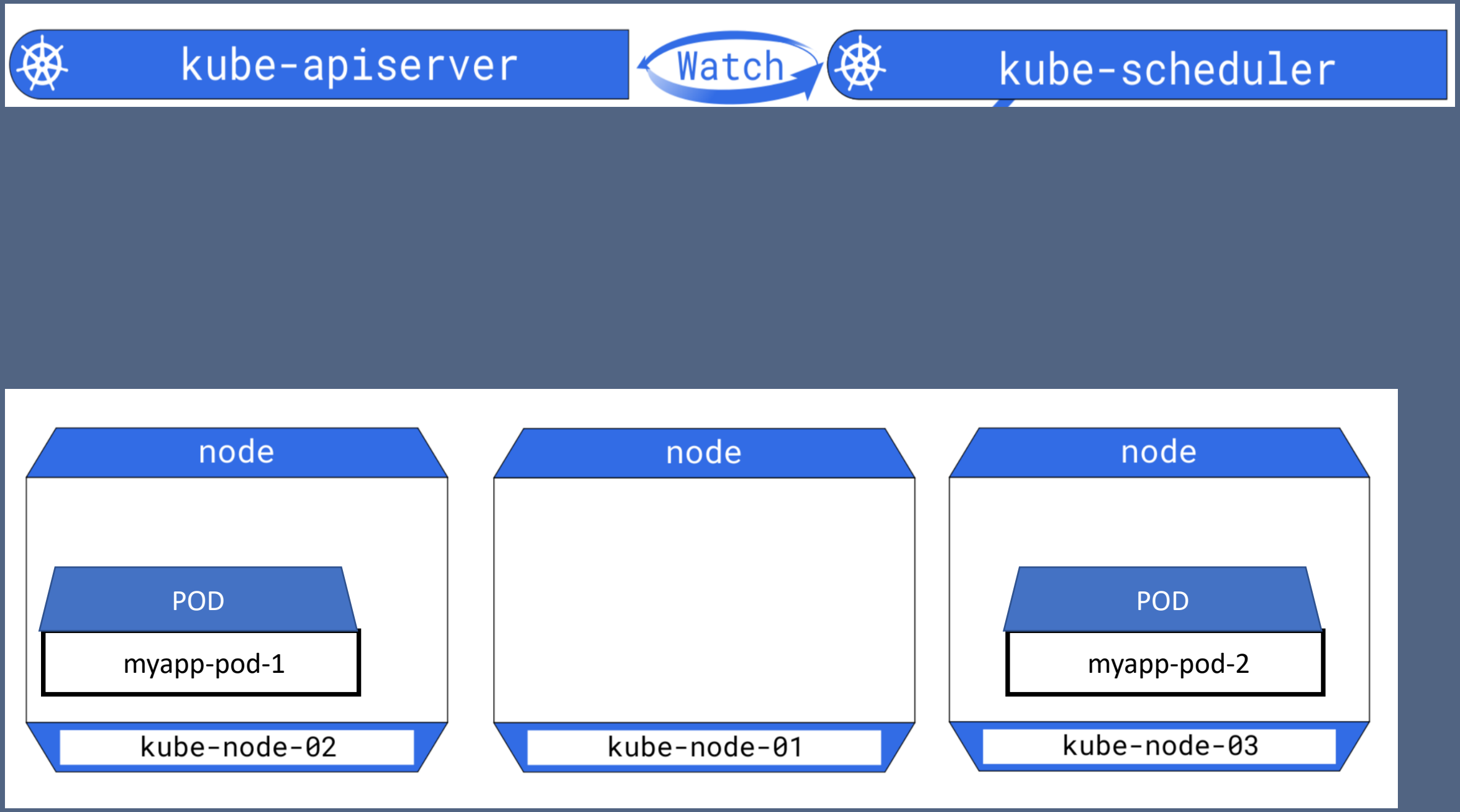

apiVersion: v1
Kind: Deployment
…
apiVersion: v1
Kind: Service
…
kube-apiserver


kube-scheduler

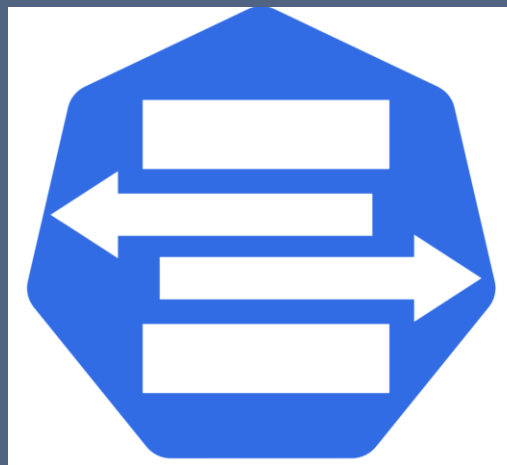

kube-apiserver
Watch
kube-scheduler
POD
myapp-pod-1
node
node
node
kube-node-02
kube-node-01
kube-node-03


kube-apiserver
Watch
kube-scheduler
node
POD
myapp-pod-1
kube-node-02
node
kube-node-01
node
kube-node-03


kube-apiserver
Watch
kube-scheduler
POD
myapp-pod-2
node
POD
myapp-pod-1
kube-node-02
node
kube-node-01
node
kube-node-03

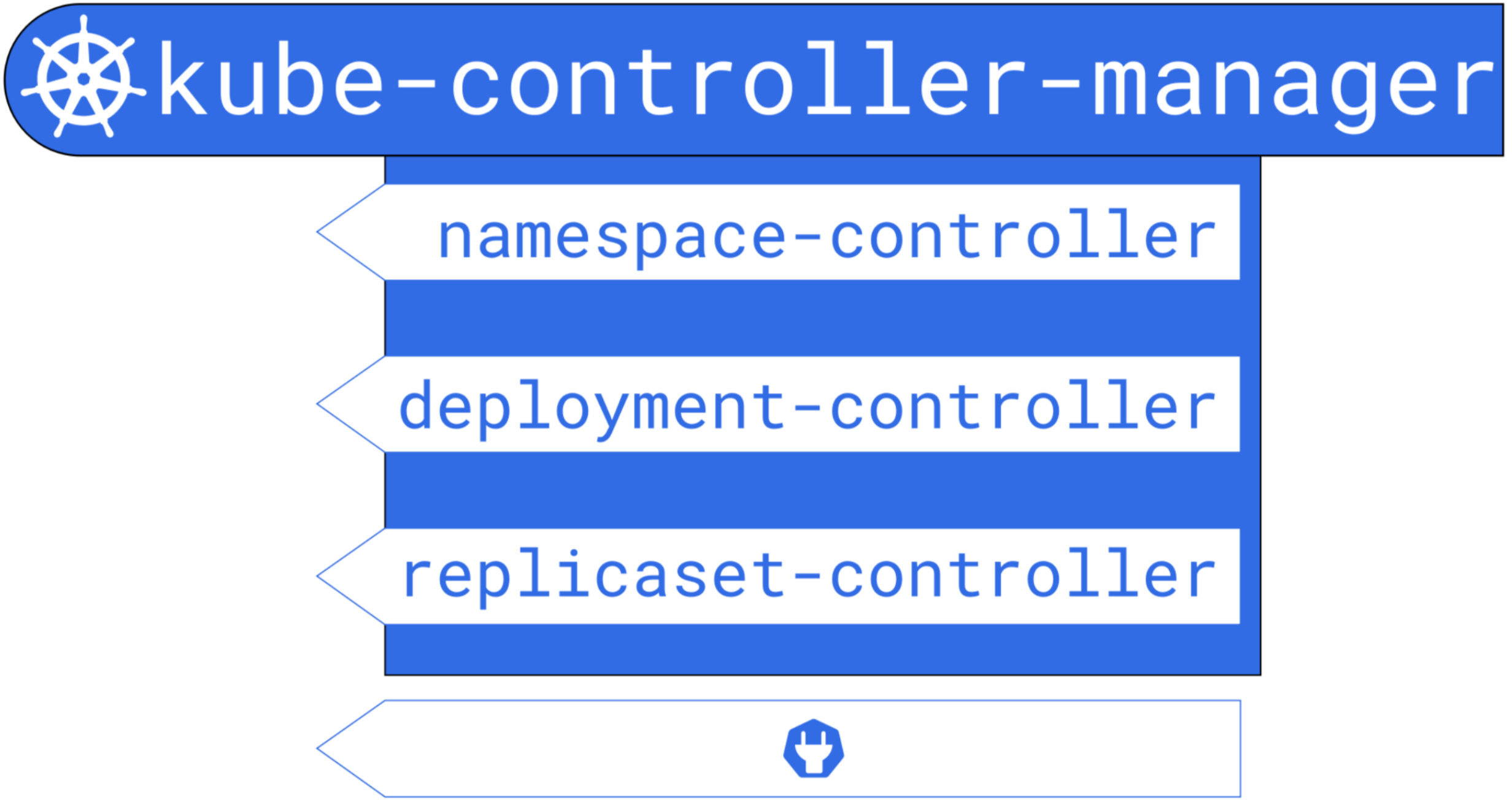

kube-apiserver
Watch
kube-scheduler
node
POD
myapp-pod-1
kube-node-02
node
kube-node-01
node
POD
myapp-pod-2
kube-node-03

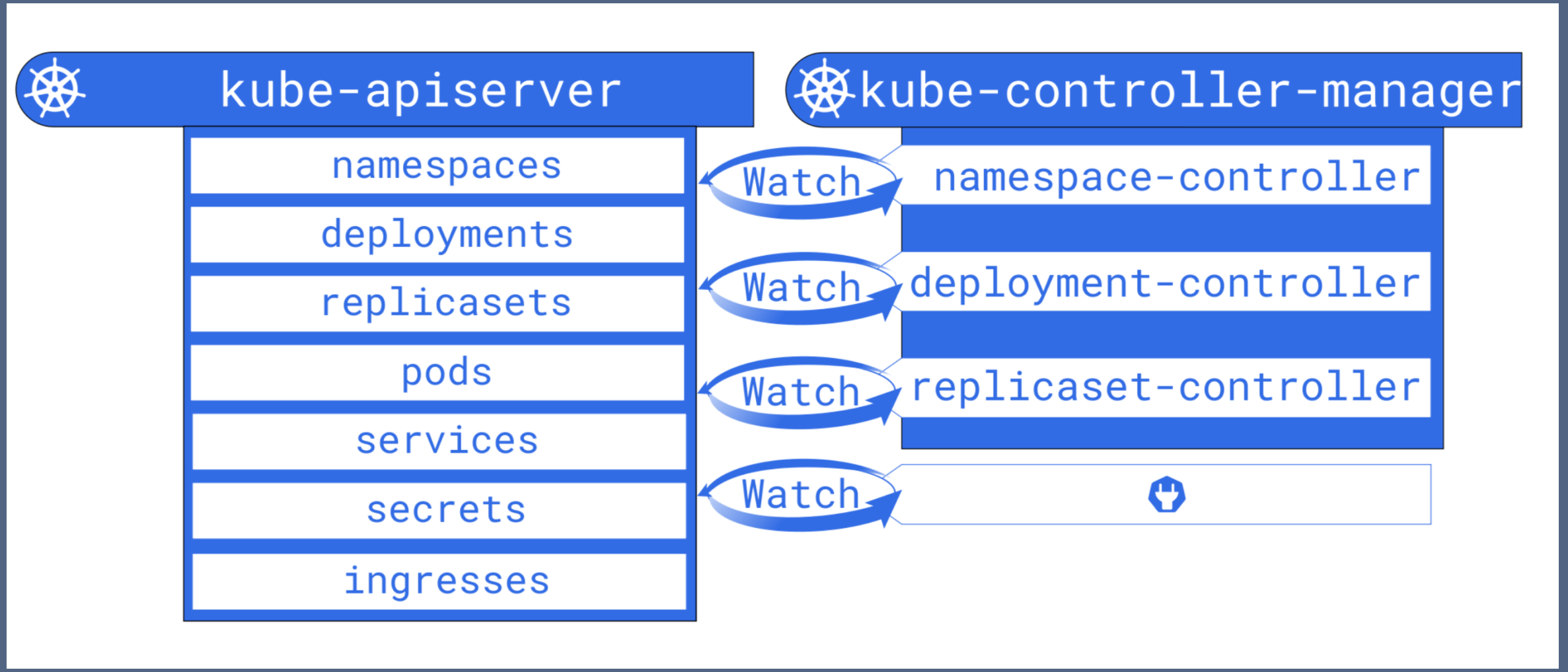
affinity/anti-affinity
nodeSelector
taints/tolerations
reservations/limits



custom
schedulers


kube-controller-manager

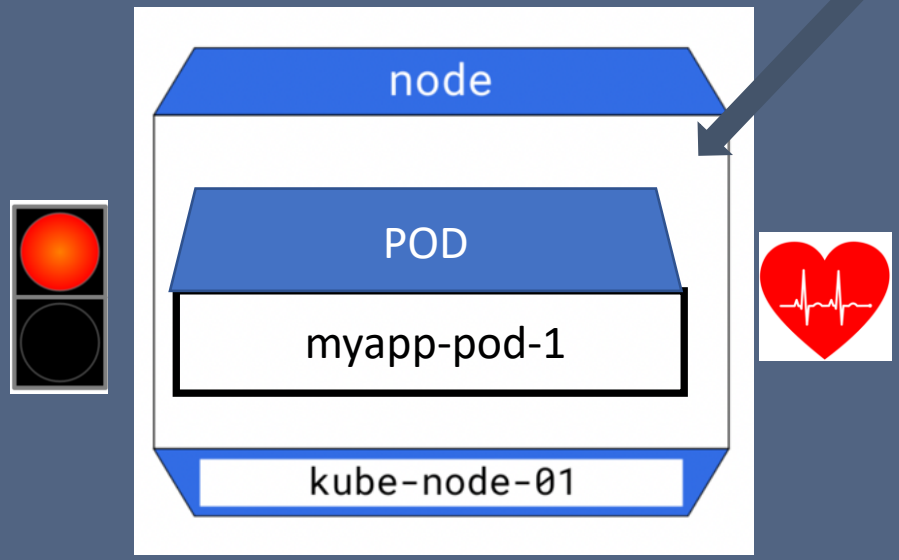

kube-controller-manager
namespace-controller
deployment-controller
replicaset-controller

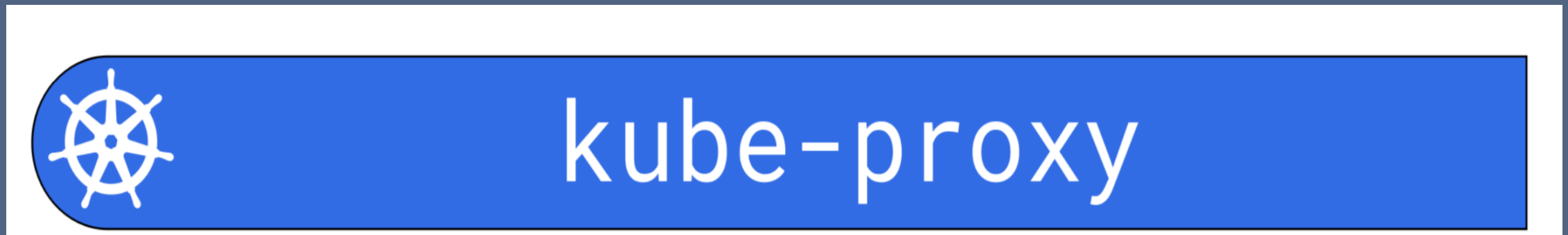

kube-apiserver
namespaces
deployments
replicasets
pods
services
secrets
ingresses
kube-controller-manager
Watch
namespace-controller
Watch
deployment-controller
Watch
replicaset-controller
Watch

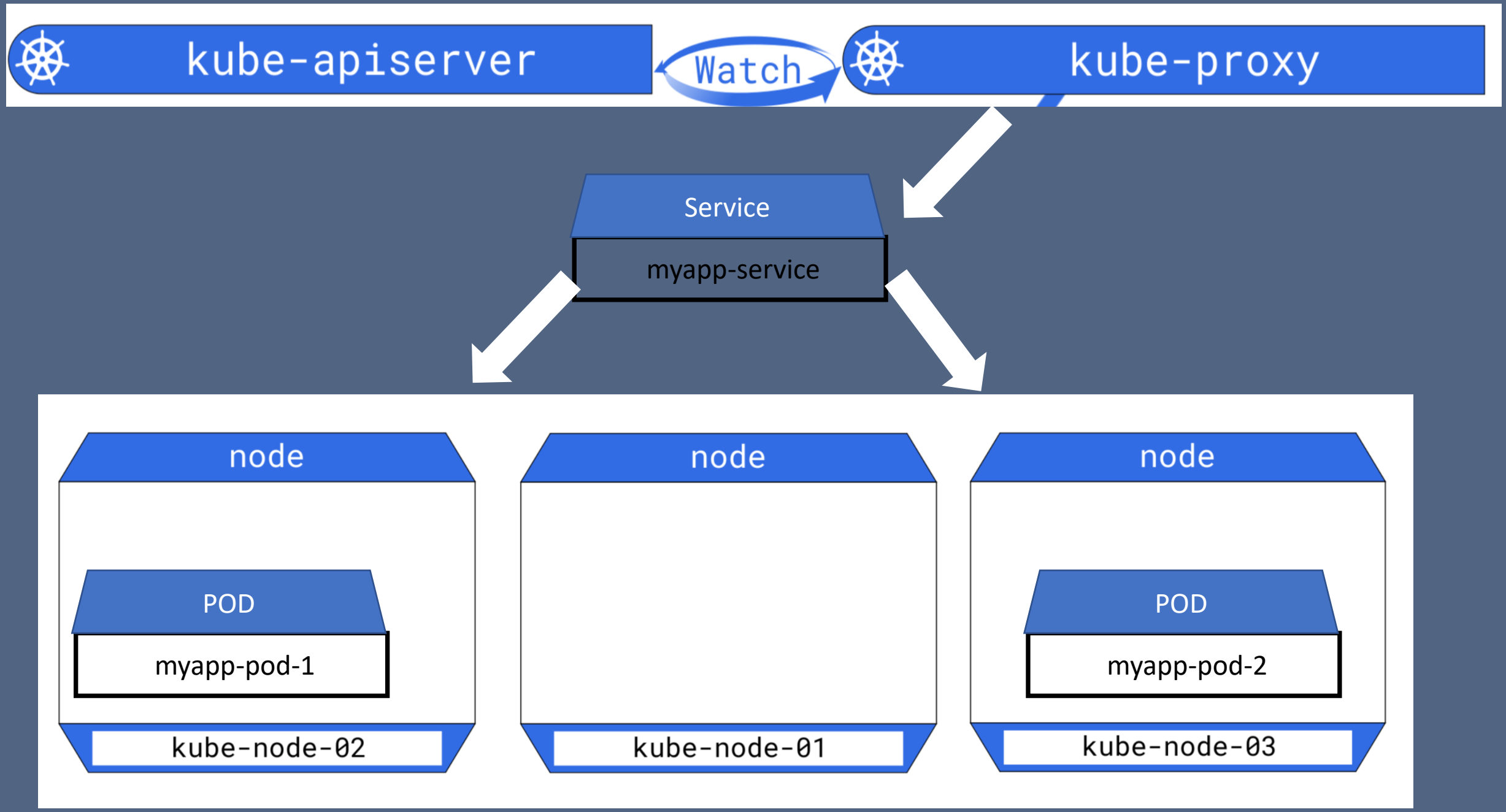

kubelet

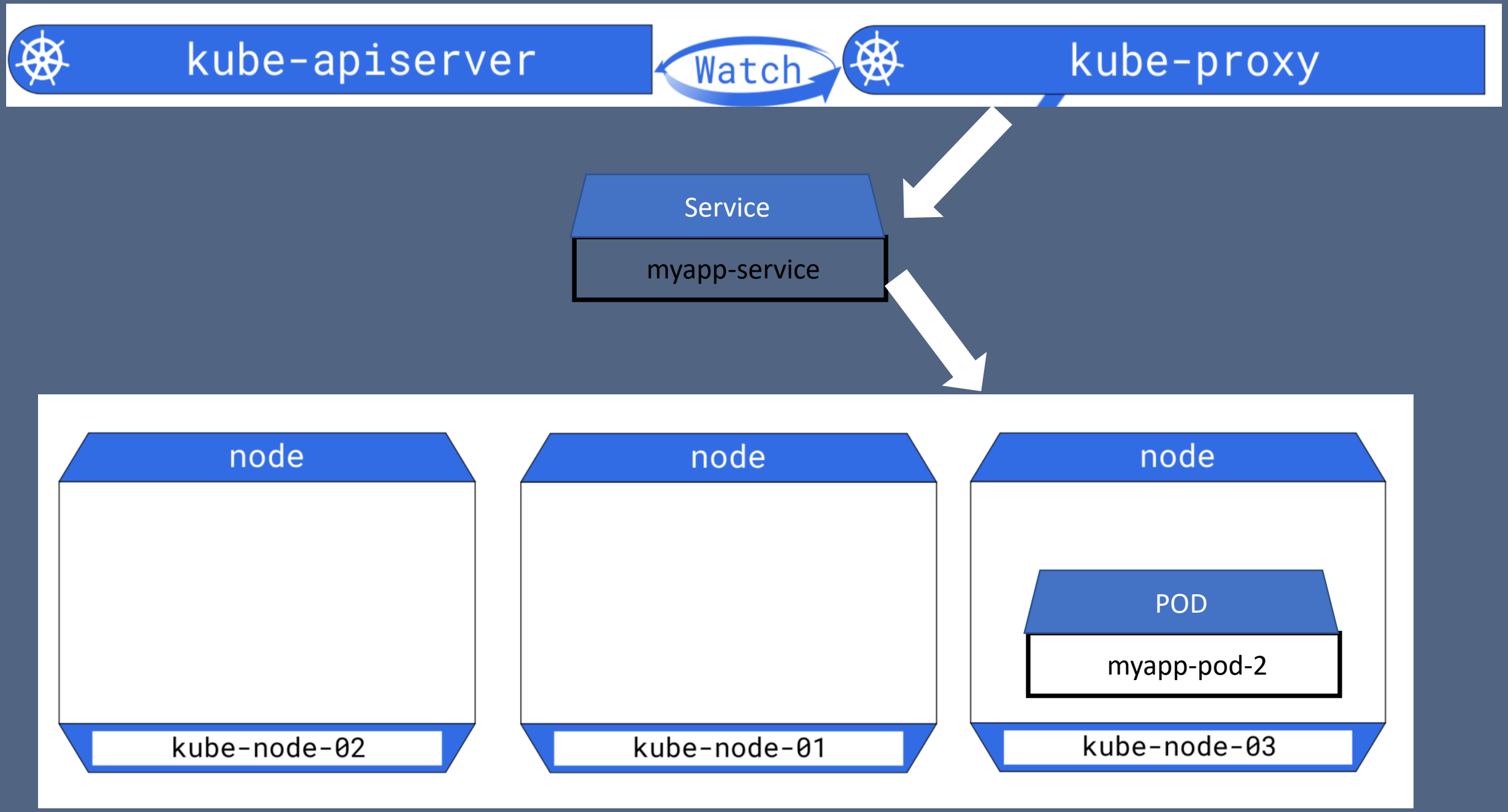
kubelet

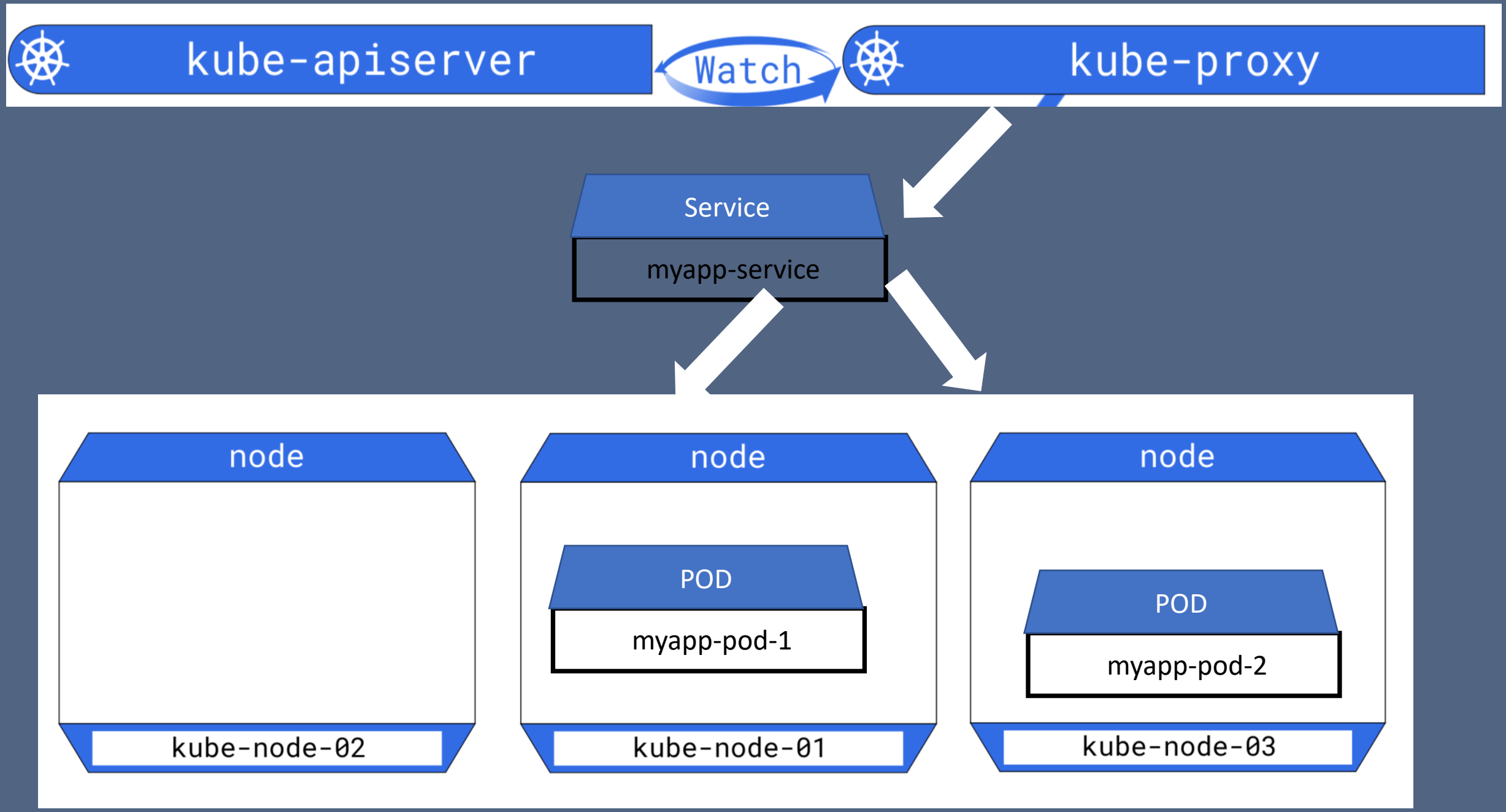
docker/rkt




kube-proxy


kube-apiserver
Watch
kube-proxy
Service
myapp-service
node
POD
myapp-pod-1
kube-node-02
node
kube-node-01
node
POD
myapp-pod-2
kube-node-03


kube-apiserver
Watch
kube-proxy
Service
myapp-service
node
node
node
POD
myapp-pod-2
kube-node-02
kube-node-01
kube-node-03


kube-apiserver
Watch
kube-proxy
Service
myapp-service
node
kube-node-02
node
POD
myapp-pod-1
kube-node-01
node
POD
myapp-pod-2
kube-node-03

Lab

- Try the various example files here:
  - https://github.com/seelam/k8s

Suggested Study Material

- Facebook Tupperware:
  - https://engineering.fb.com/data-center-engineering/tupperware/
- Kubernetes deconstructed
  - http://kube-decon.carson-anderson.com/Layers/0-Intro.sozi.html#frame5378
- The History of Kubernetes on a Timeline: https://blog.risingstack.com/the-history-of-kubernetes/
- The State of the Kubernetes Ecosystem
  - https://thenewstack.io/ebooks/kubernetes/state-of-kubernetes-ecosystem/