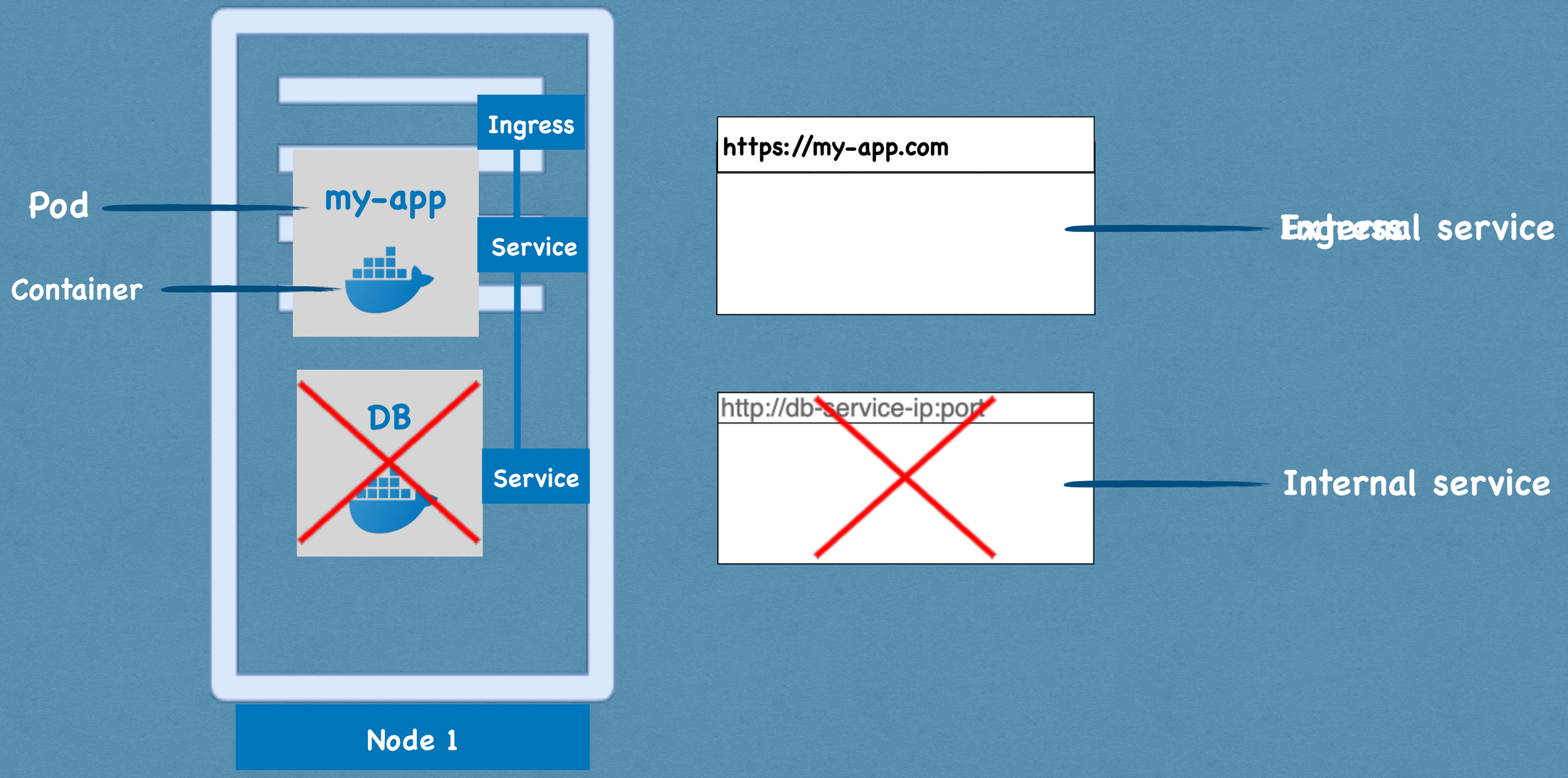
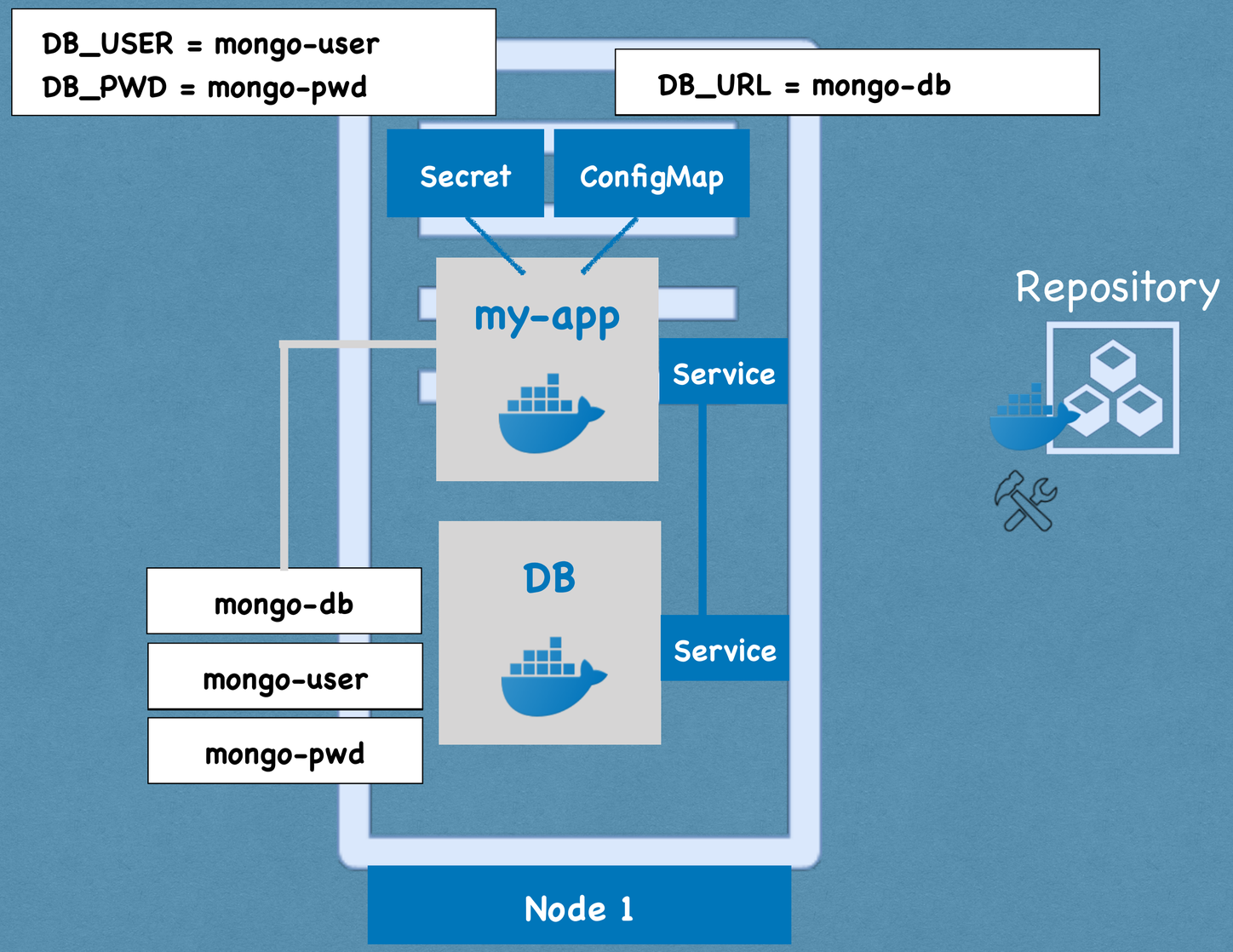
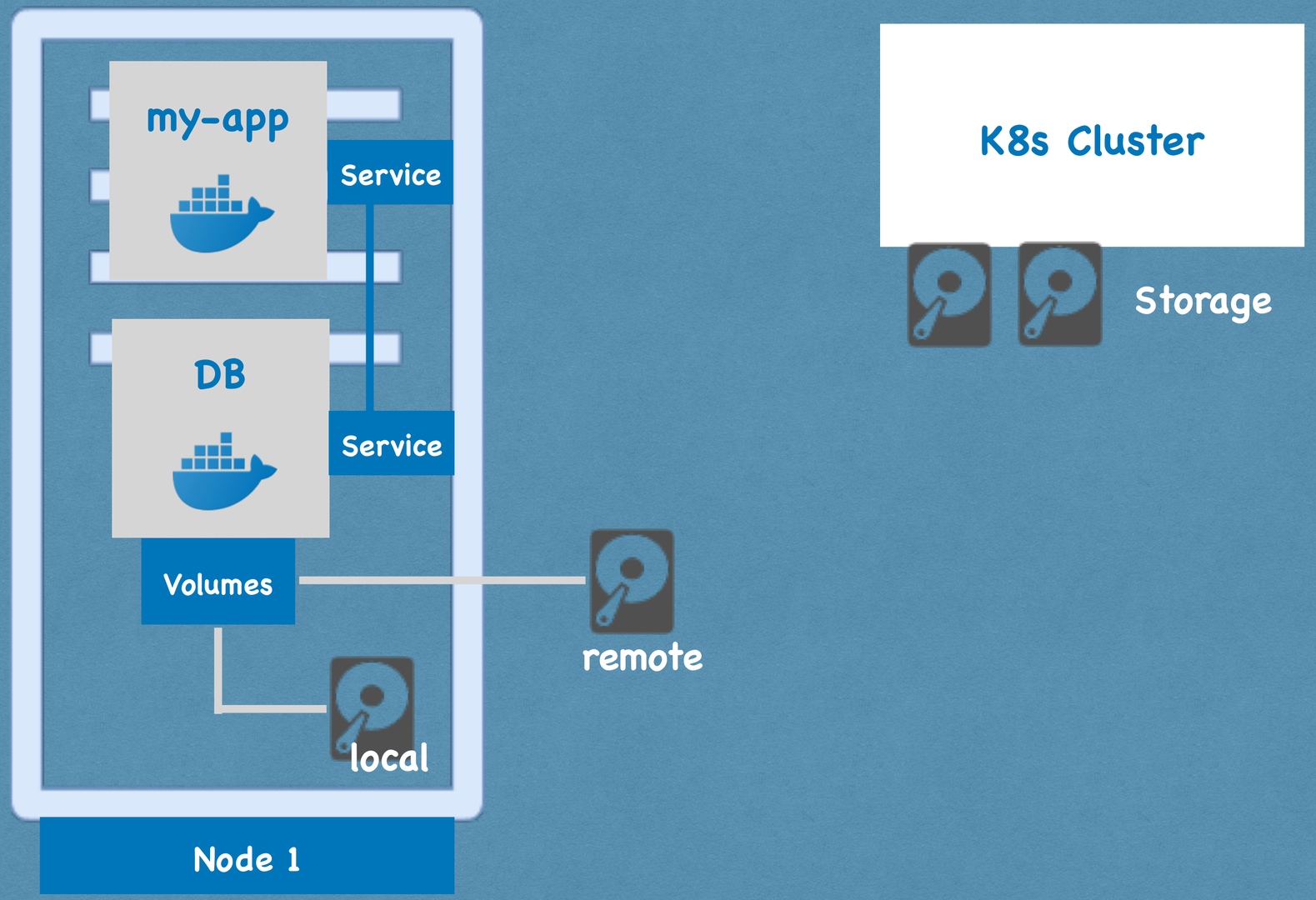
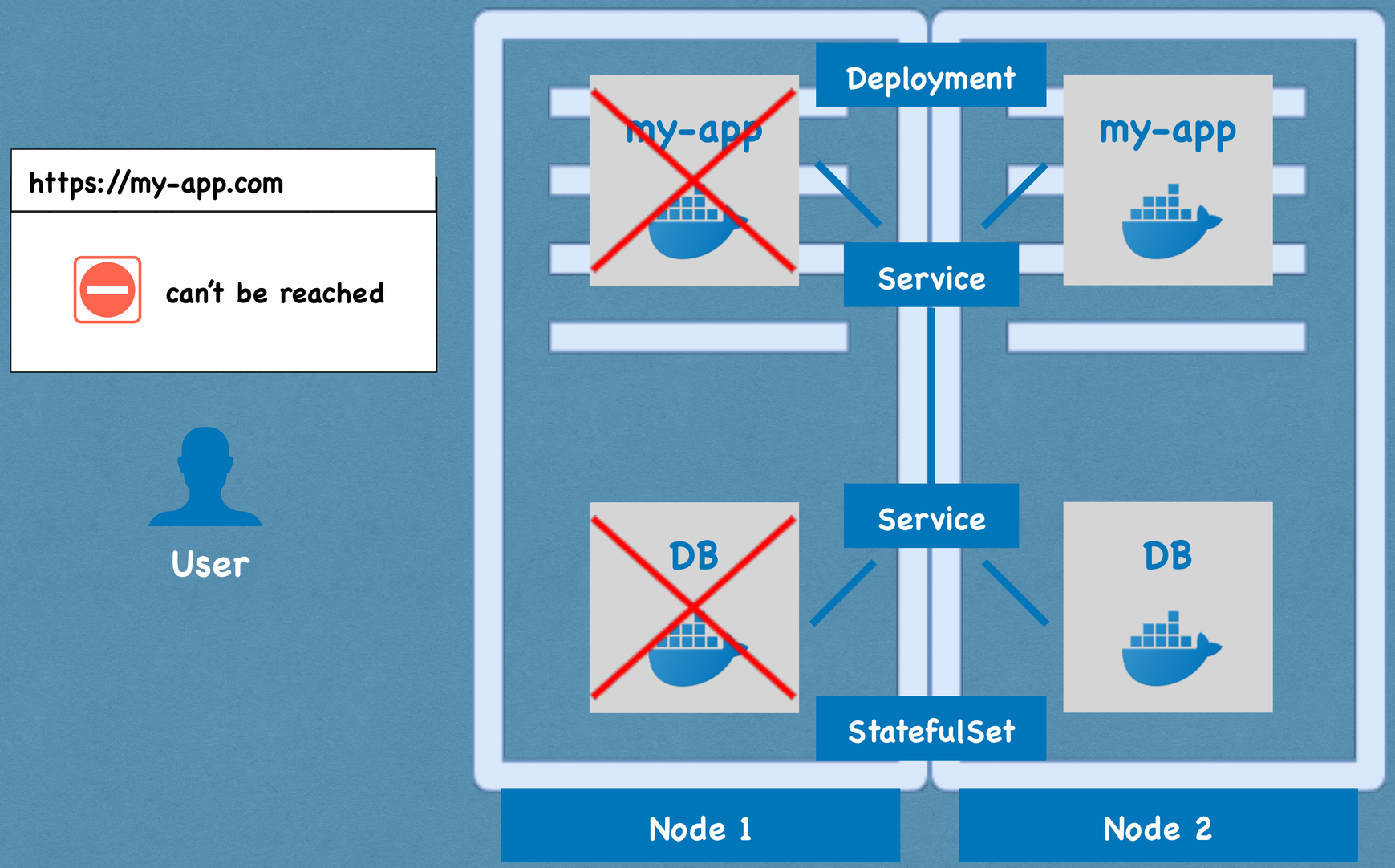
Basic Components


Pod
Container
my-app
Ingress
Service
DB
Service
Node 1
https://my-app.com
External service
http://db-service-ip:port
Internal service

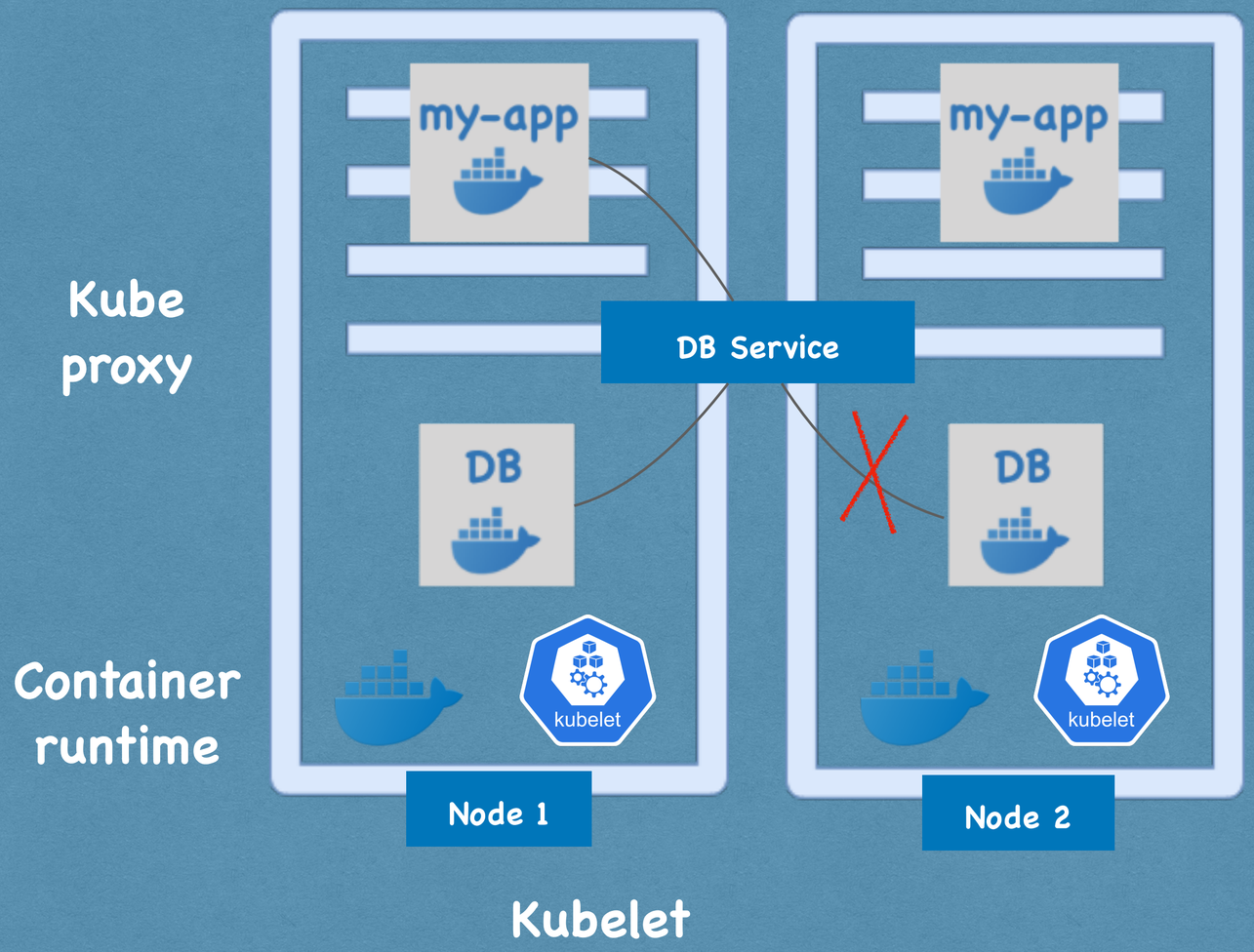

DB_USER = mongo-user
DB_PWD = mongo-pwd
DB_URL = mongo-db
Secret
ConfigMap
my-app
Service
DB
Service
mongo-db
mongo-user
mongo-pwd
Node 1
Repository


my-app
Service
DB
Service
Volumes
remote
local
Node 1
K8s Cluster
Storage

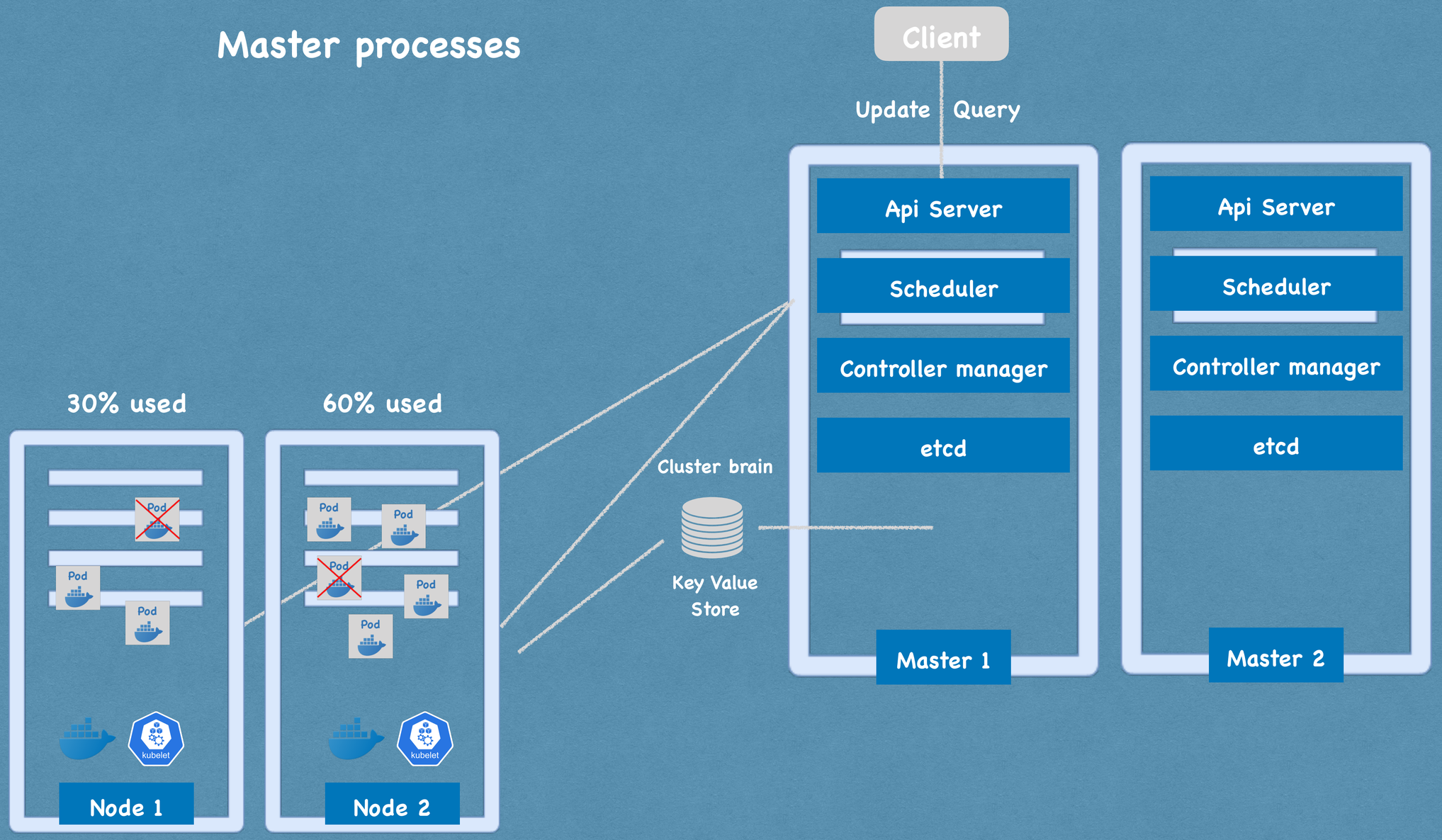

https://my-app.com
can't be reached
User
Deployment
my-app
my-app
Service
Service
DB
DB
StatefulSet
Node 1
Node 2

K8s architecture

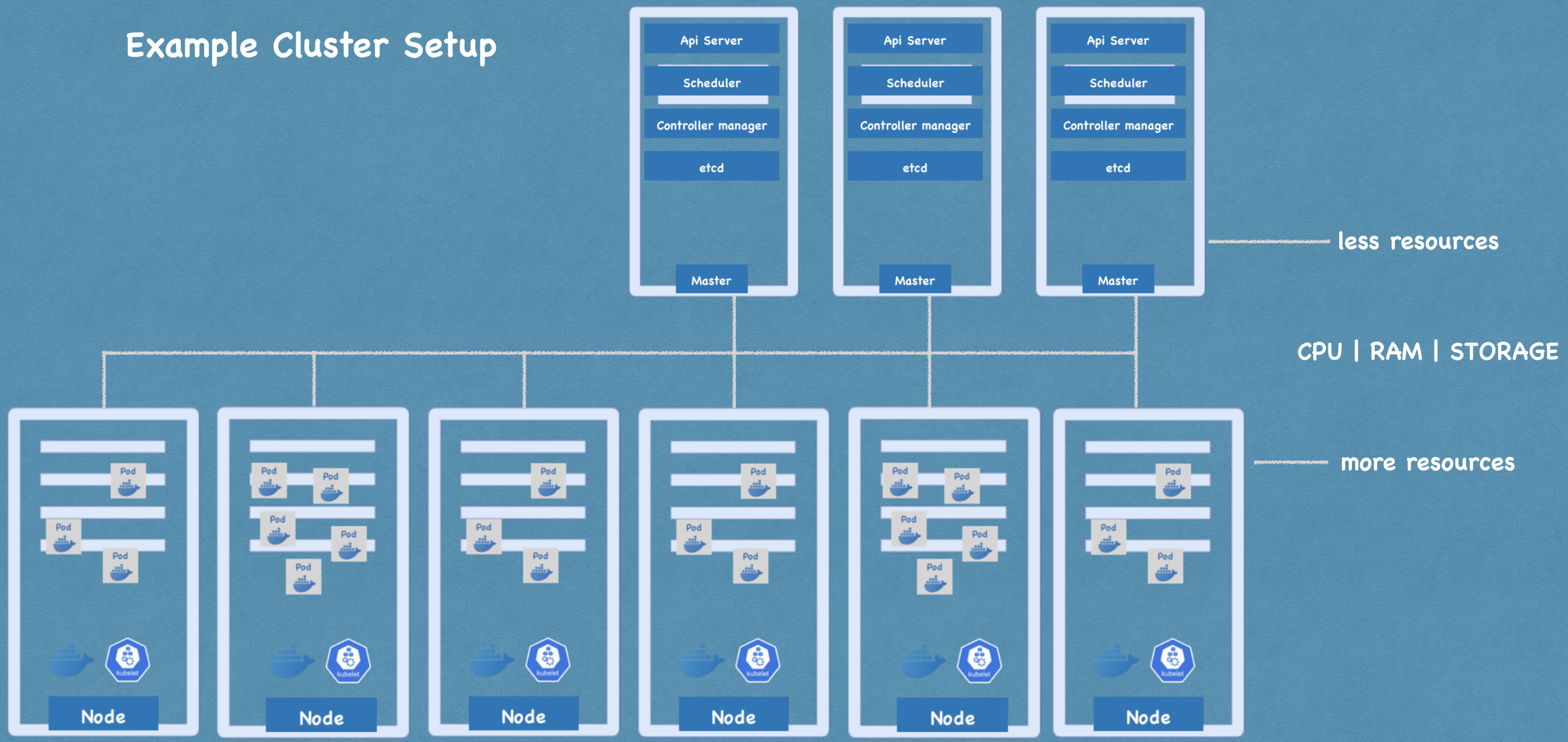
Node processes - k8s Worker



Master processes

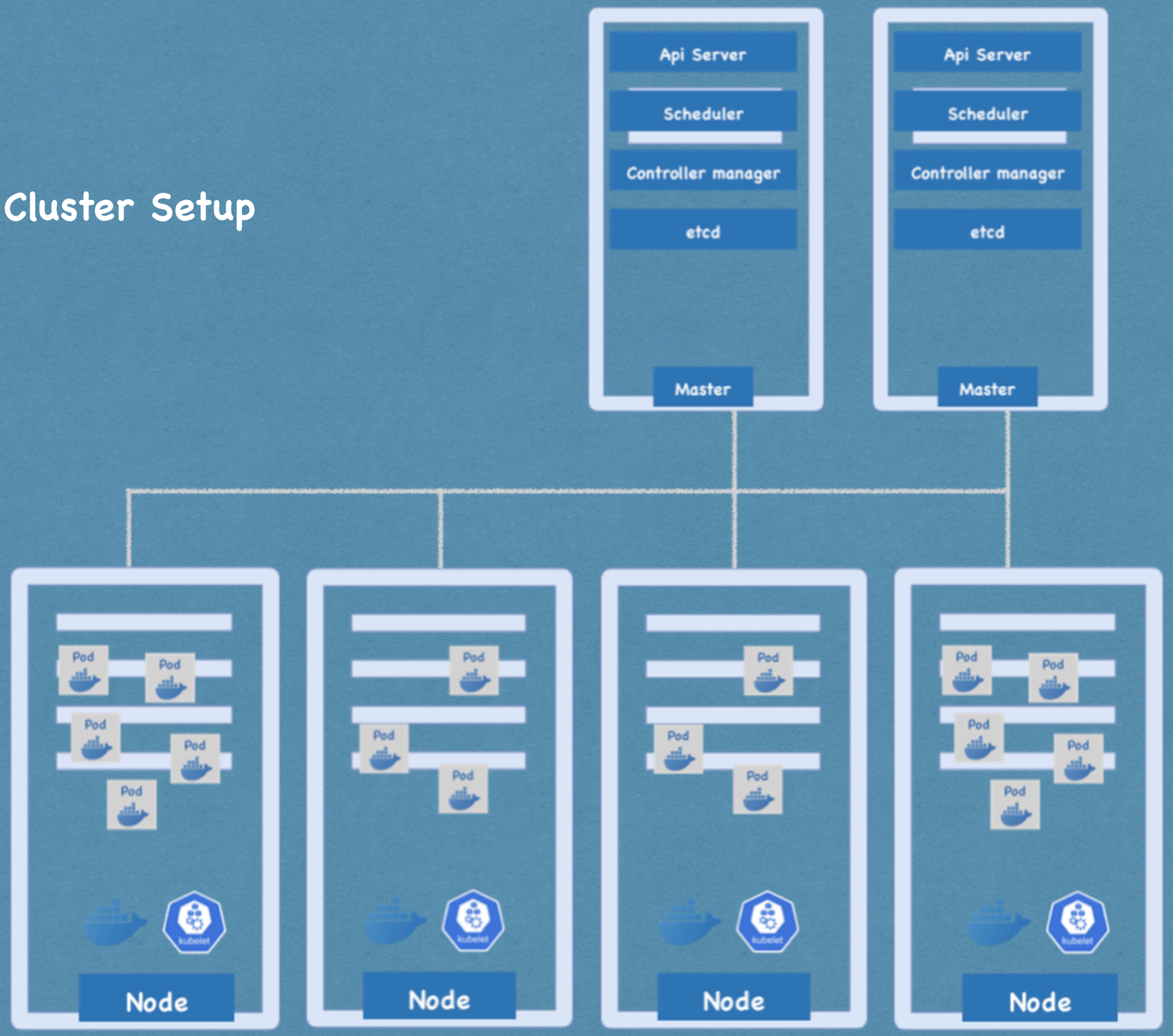


Example Cluster Setup

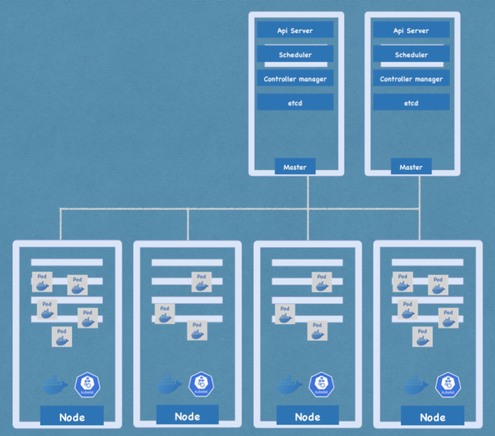


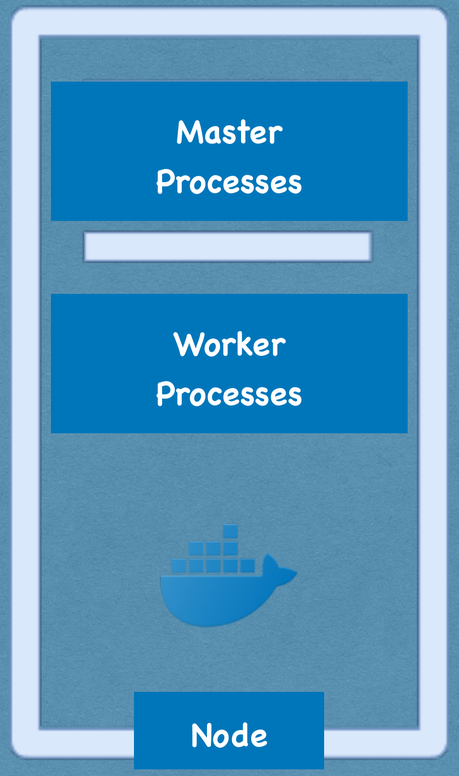
Minikube and kubectl

Production Cluster Setup

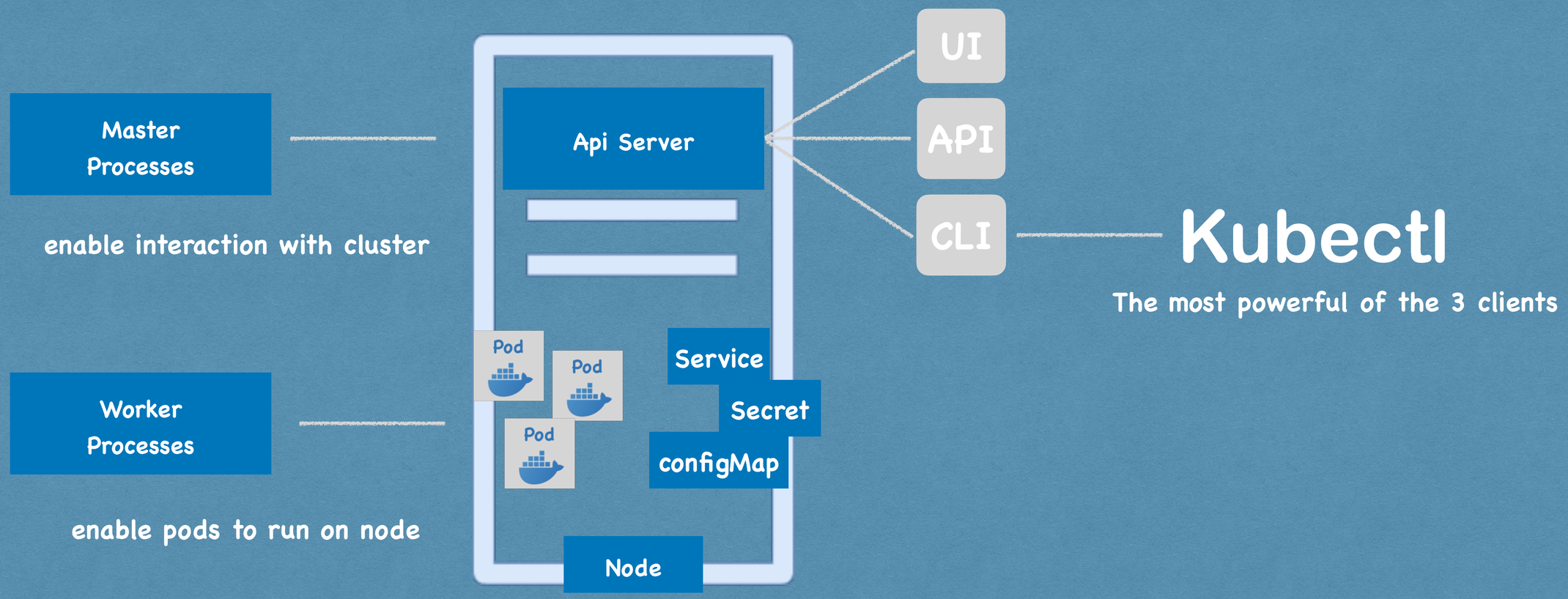


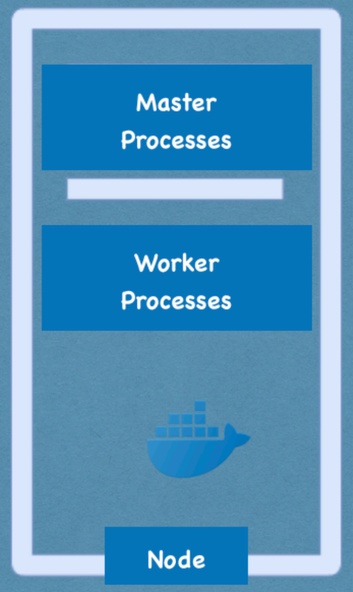
Test/Local Cluster Setup



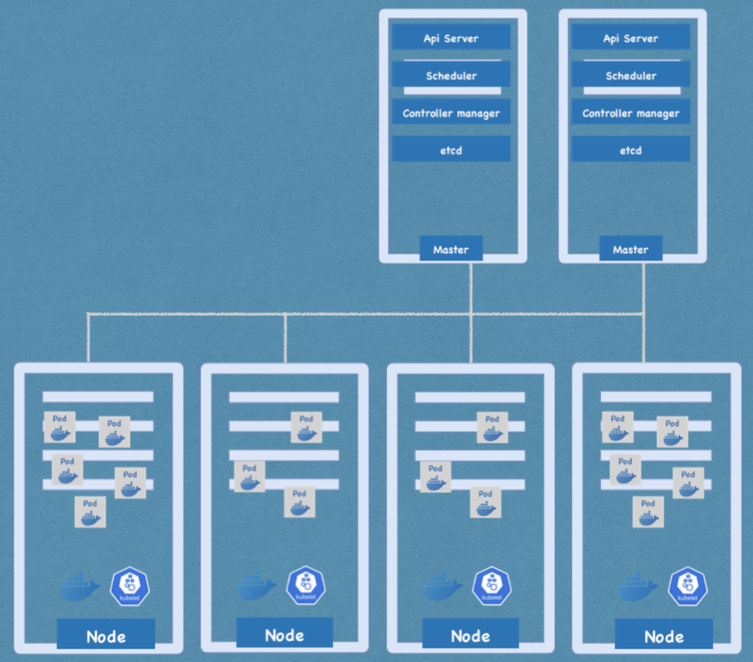


What is Kubectl


Master Processes
enable interaction with cluster
Worker Processes
enable pods to run on node
Api Server
Pod
Pod
Pod
Service
Secret
configMap
Node
UI
API
CLI
Kubectl
The most powerful of the 3 clients



Minikube Cluster



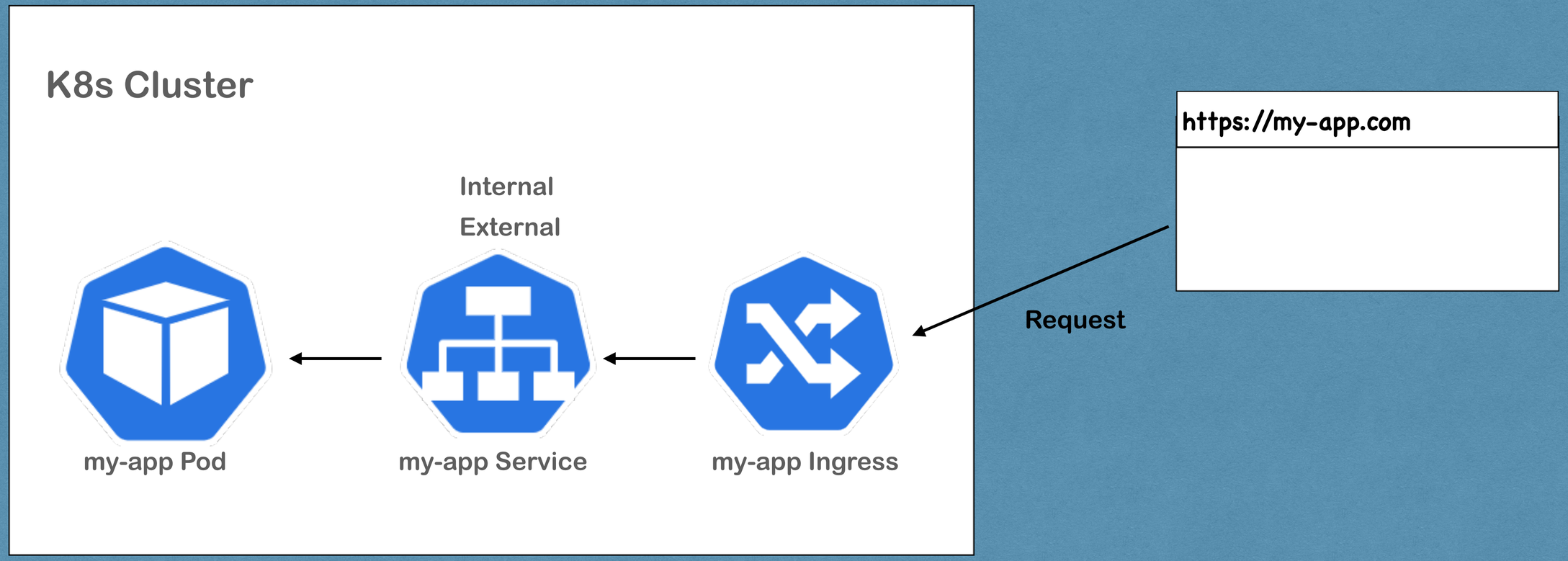
Cloud Cluster

Kubectl

What is Ingress

What is Ingress

External service vs Ingress



Example yaml files

```yaml
apiVersion: v1
kind: Service
metadata:
  name: myapp-external-service
spec:
  selector:
    app: myapp
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
      nodePort: 35010
```

External Service

http://124.89.101.2:35010

Example yams files

Ingress Syntax explained

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

Example yams files

Ingress

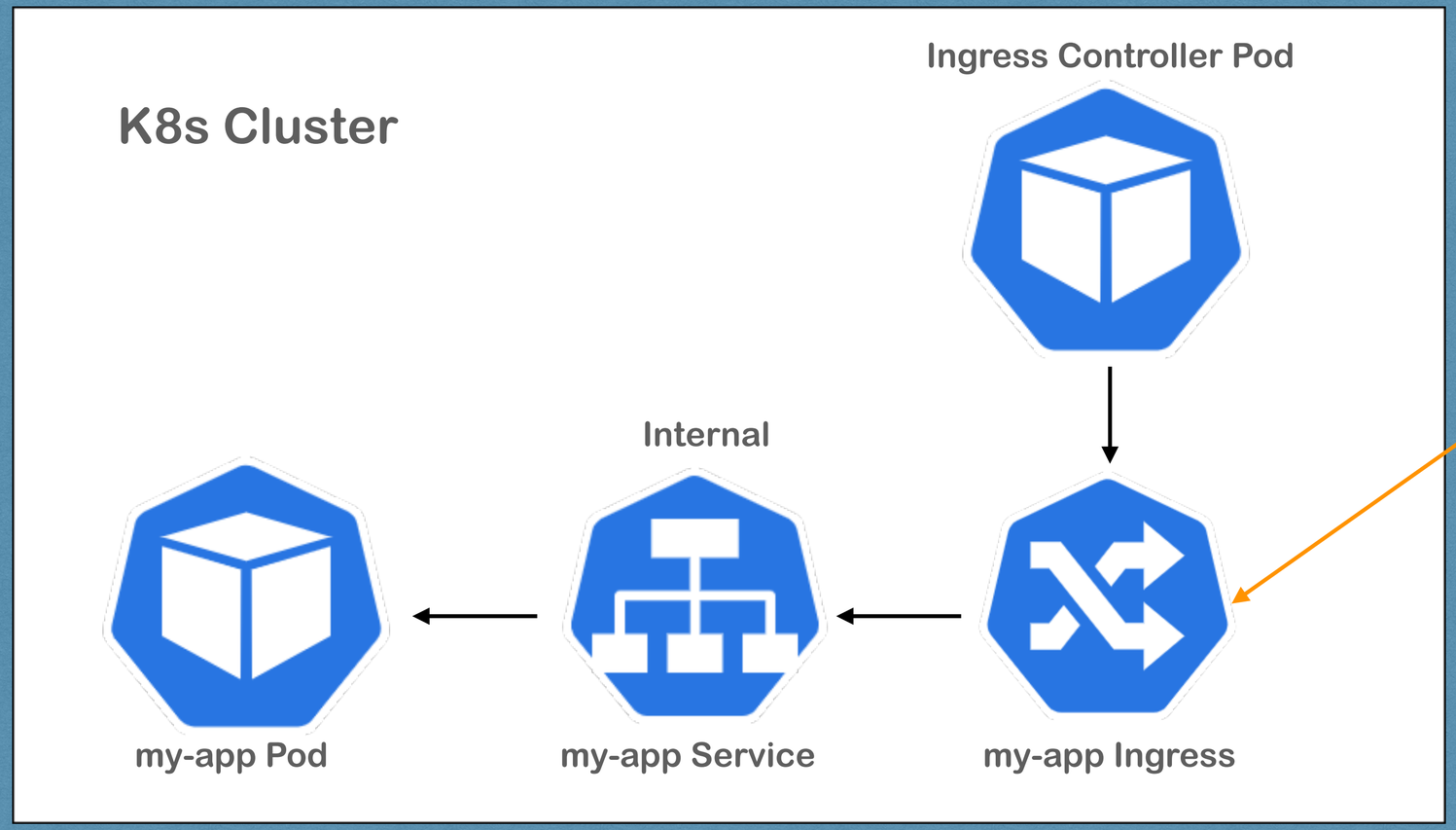
```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: myapp-ingress
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - backend:
          serviceName: myapp-internal-service
          servicePort: 8080
```

Internal Service

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

How to configure Ingress in your cluster

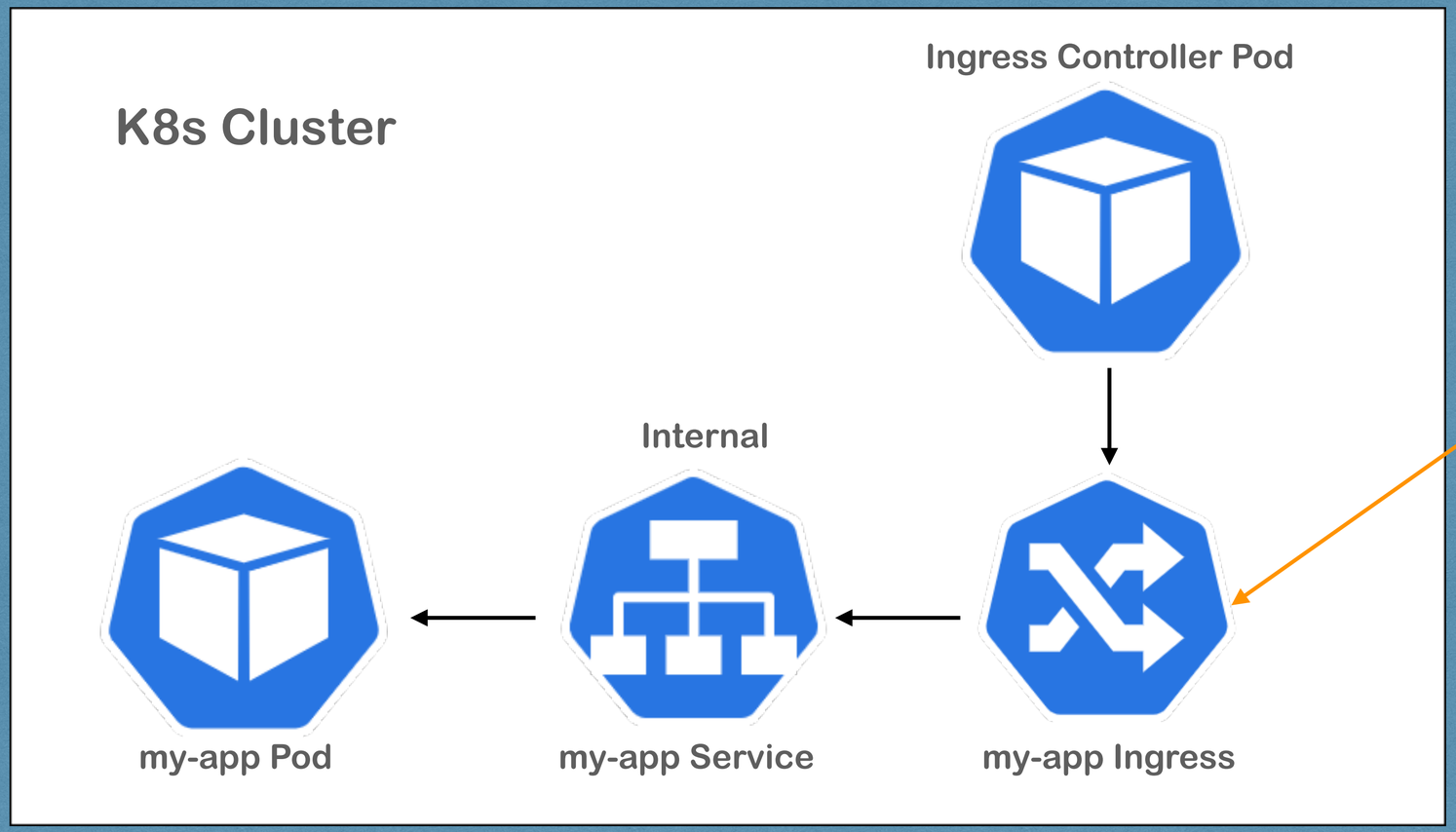
Step 1: install ingress controller



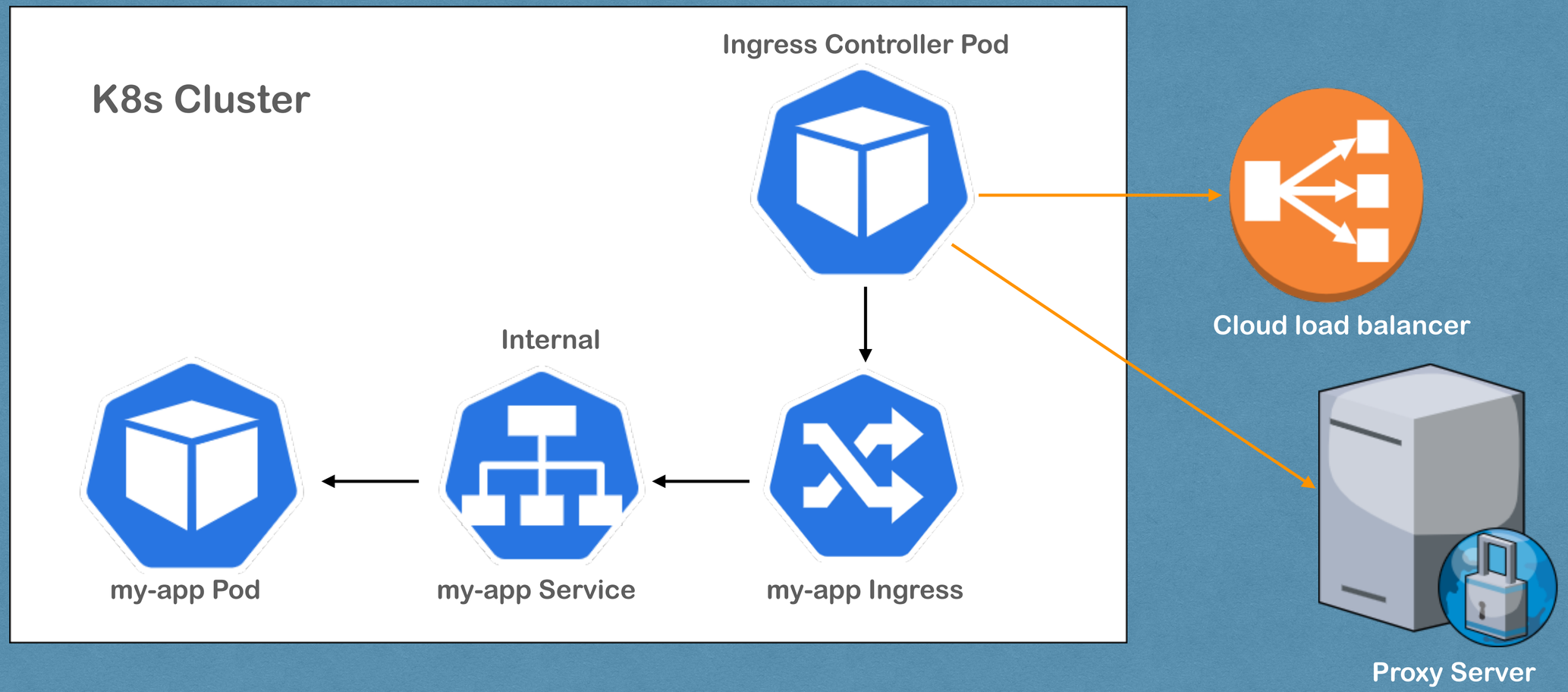

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: myapp-ingress
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - backend:
          serviceName: myapp-internal-service
          servicePort: 8080
```

{The yaml created Ingress component. But it's not gonna work without ingress controller running. So that's the first step}

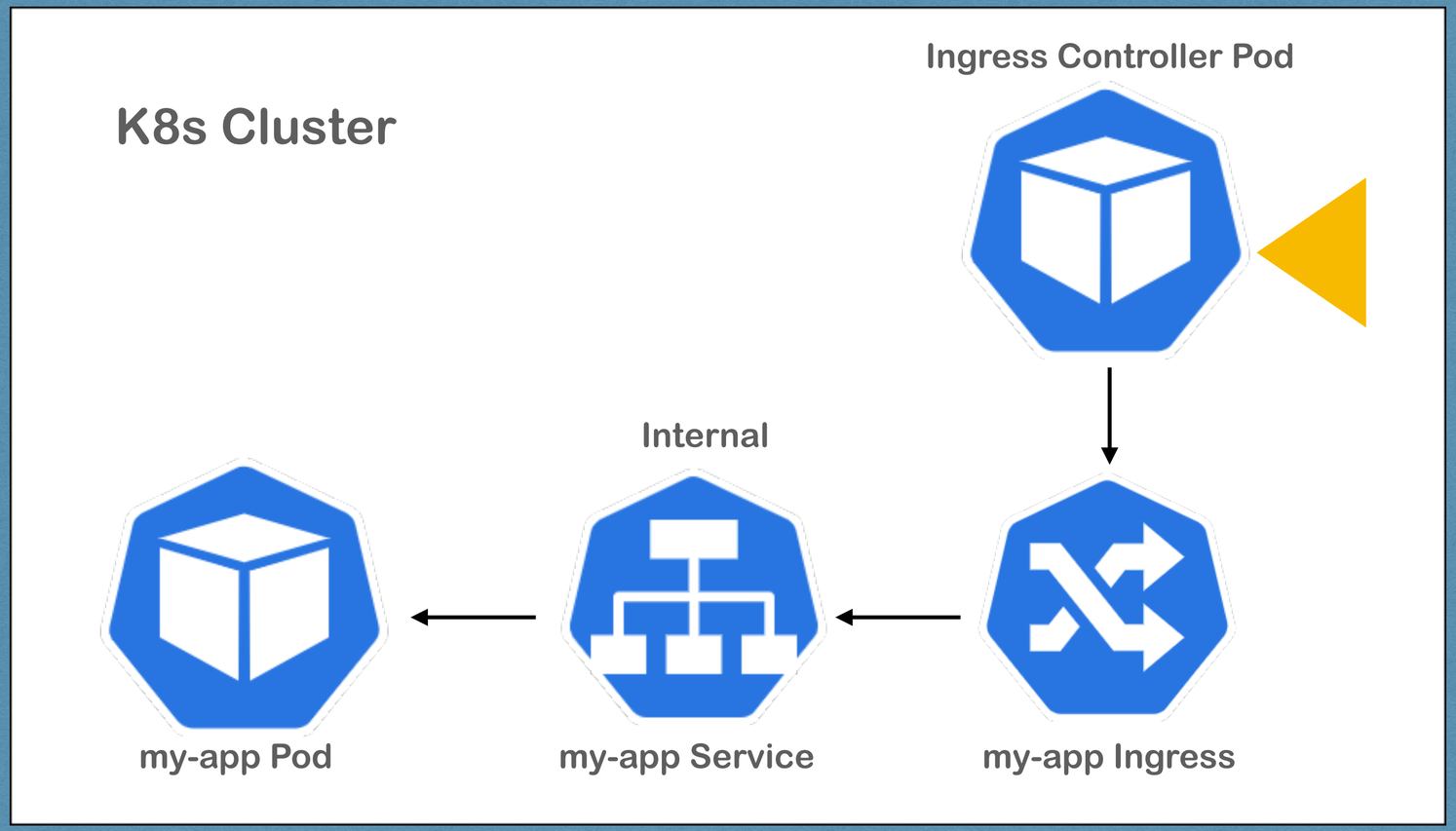
So what is Ingress controller?




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


K8s Cluster
Ingress Controller Pod
Cloud load balancer
Internal
my-app Pod
my-app Service
my-app Ingress
Proxy Server

Ingress controller in Minikube

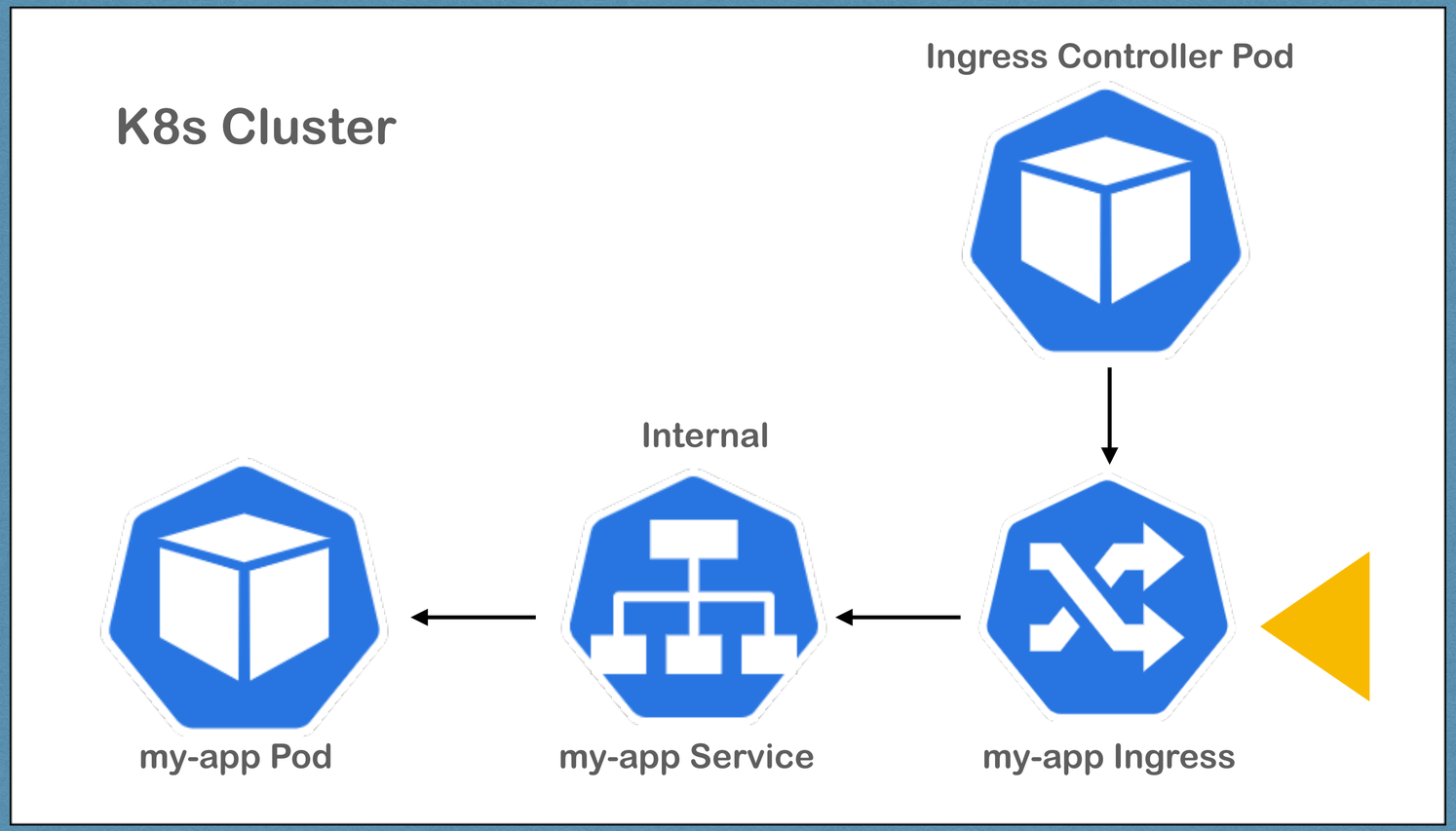


Demo in Minikube with Kuberenets Ingress controller

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

```
[~]$ kubectl get pod -n kube-system
NAME                                        READY   STATUS    RESTARTS   AGE
coredns-6955765f44-bdrxd                    1/1     Running   0          5d1h
coredns-6955765f44-x94rx                    1/1     Running   0          5d1h
etcd-minikube                               1/1     Running   3          22d
kube-addon-manager-minikube                 1/1     Running   3          22d
kube-apiserver-minikube                     1/1     Running   3          22d
kube-controller-manager-minikube            1/1     Running   50         22d
kube-proxy-6g8k4                            1/1     Running   3          22d
kube-scheduler-minikube                     1/1     Running   49         22d
nginx-ingress-controller-6fc5bcc8c9-wd4g9   1/1     Running   0          30h
storage-provisioner                         1/1     Running   0          32h
```

Ingress controller in Minikube



Step 2: create an Ingress Rule for your app

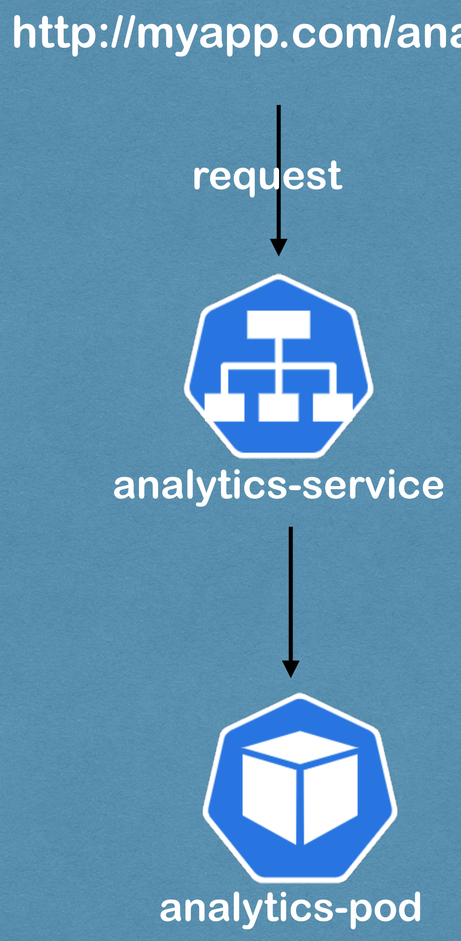
Example yaml file for Ingress rule

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

Examples of Routing - More details in Ingress Yaml

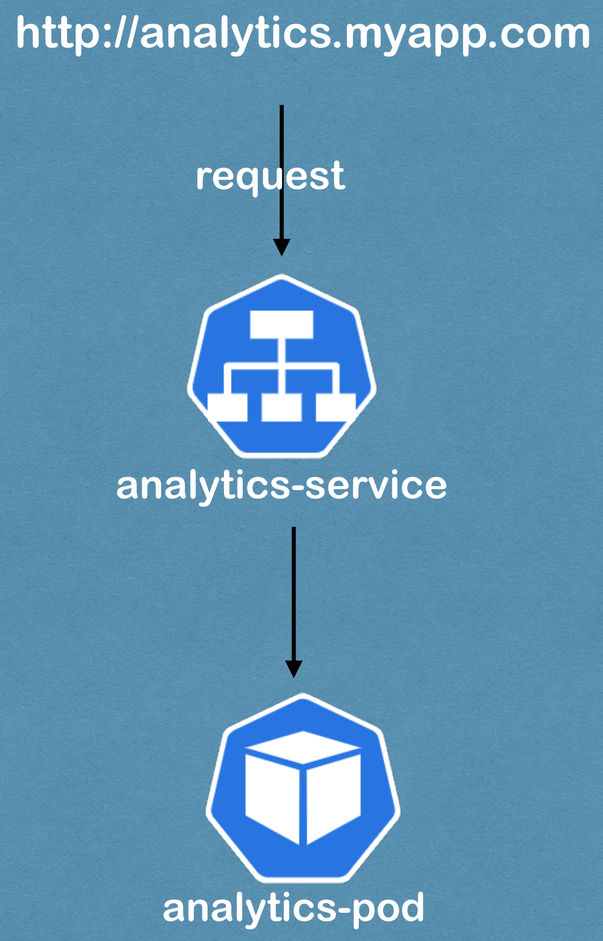
```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - path: /analytics
        backend:
          serviceName: analytics-service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

Multiple Paths of the same host



```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: analytics.myapp.com
    http:
      paths:
        backend:
          serviceName: analytics-service
          servicePort: 3000
  - host: shopping.myapp.com
    http:
      paths:
        backend:
          serviceName: shopping-service
          servicePort: 8080
```

Multiple sub-domains or domains



TLS certificate

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

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: myapp-secret-tls
  namespace: default
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
type: kubernetes.io/tls
```