DATE : 22.03.2024

DT/NT : DT

LESSON : DEVOPS

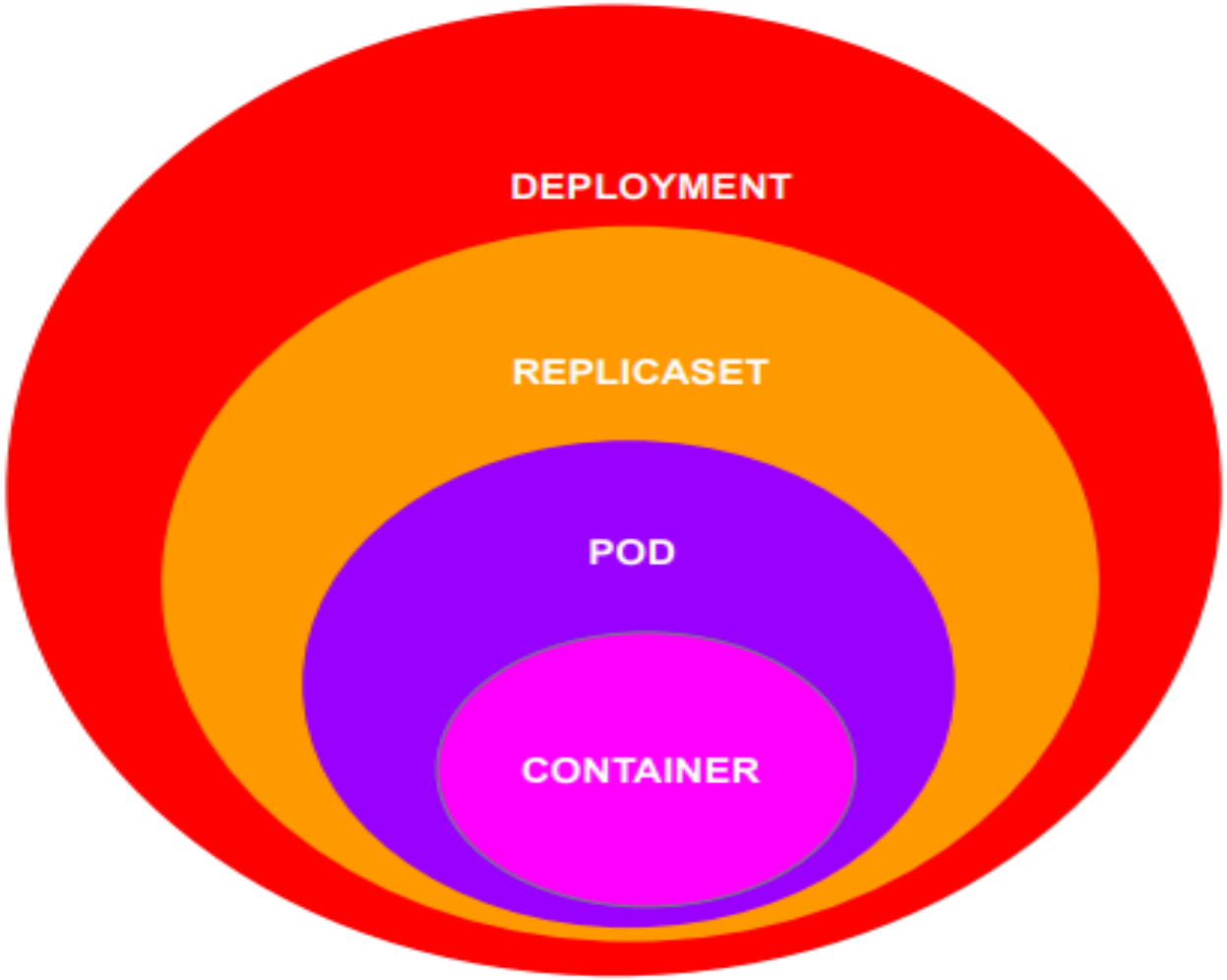
SUBJECT : Kubernetes-3 Network

BATCH : B 224

AWS-DEVOPS

TECHPRO EDUCATION

Deployment



TECHPRO EDUCATION

Cluster Networking

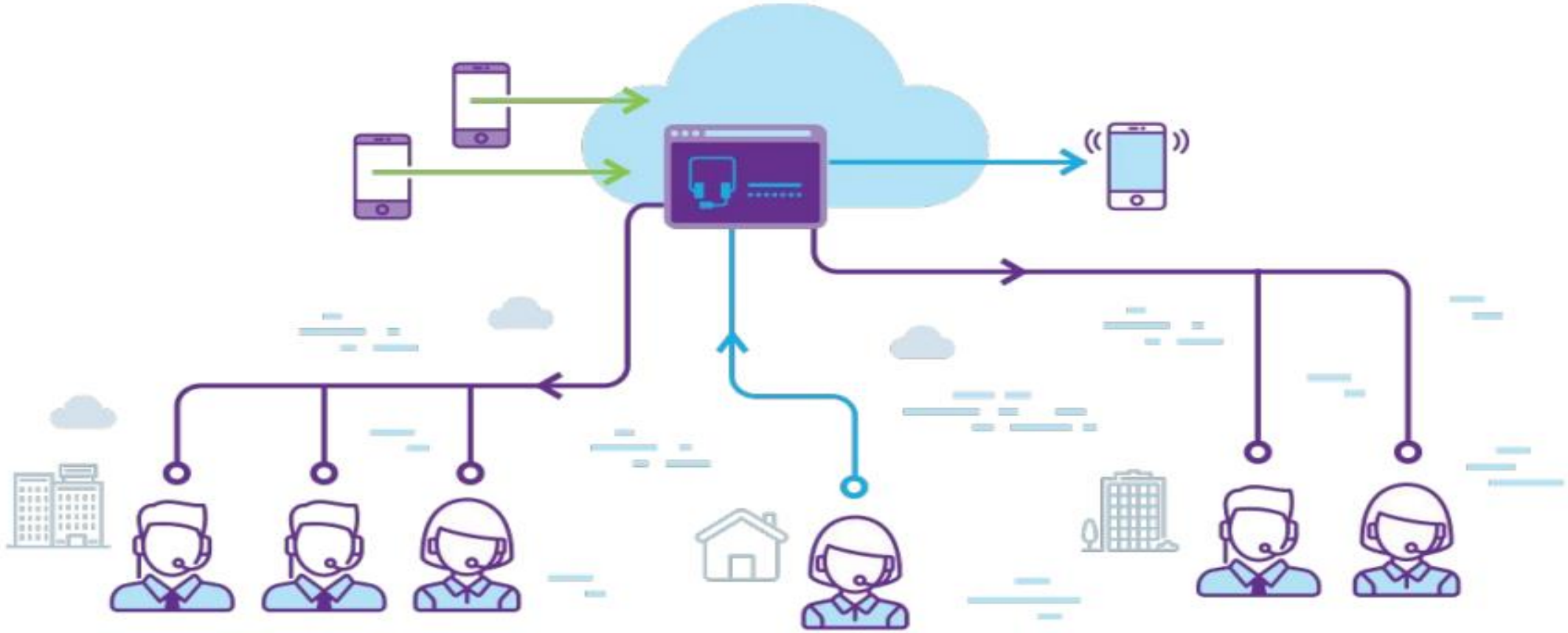


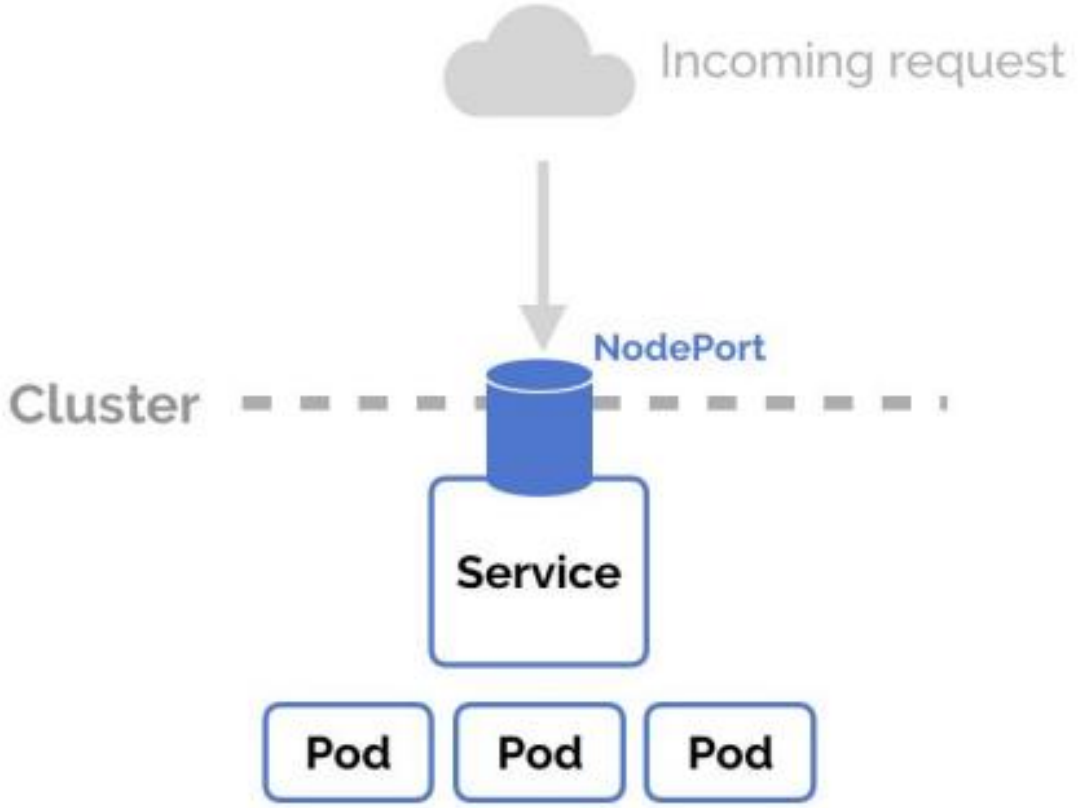
Table of Contents

Cluster Networking

There are 4 distinct networking problems to address:

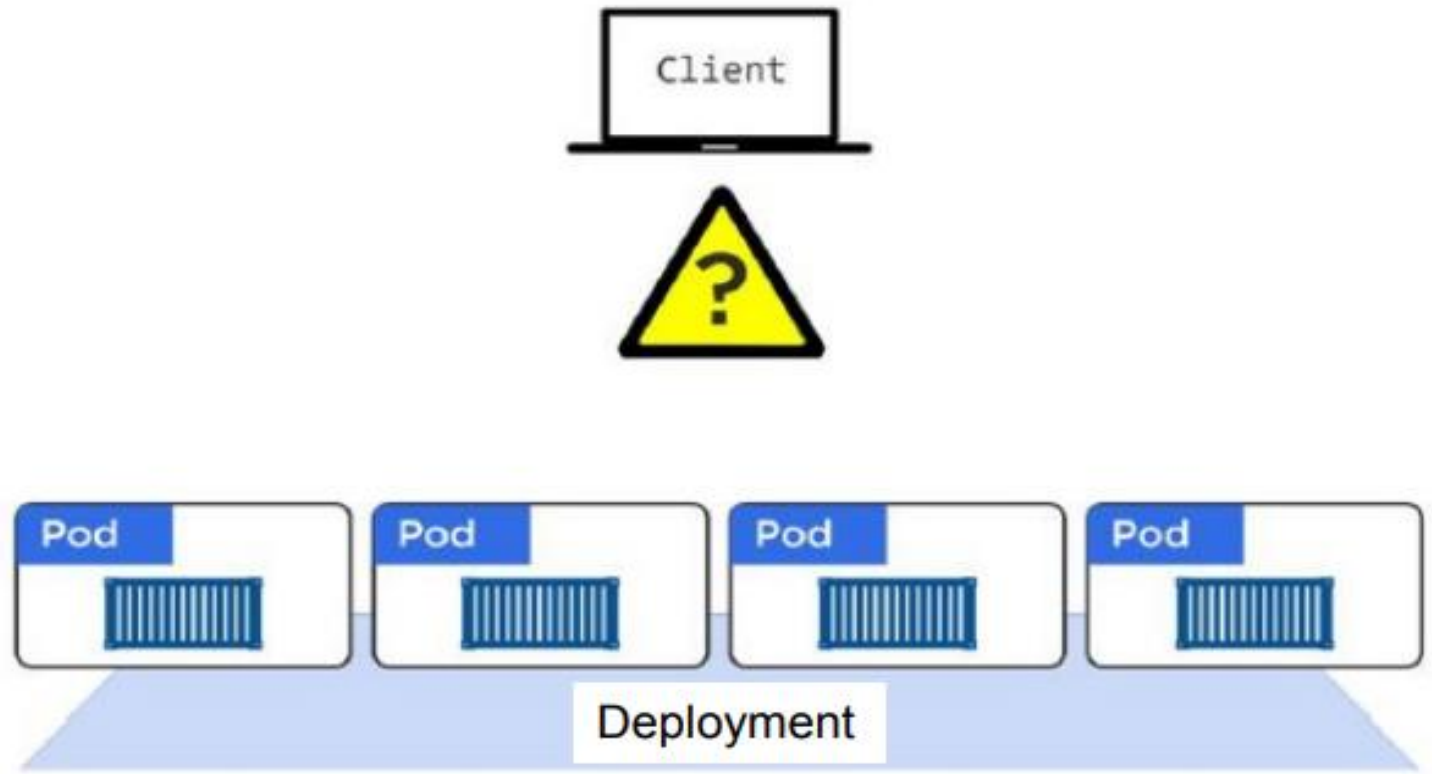
1. container-to-container communications:
2. Pod-to-Pod communications:
3. Pod-to-Service communications: this is covered by services.
4. External-to-Service communications: this is covered by services.

Services



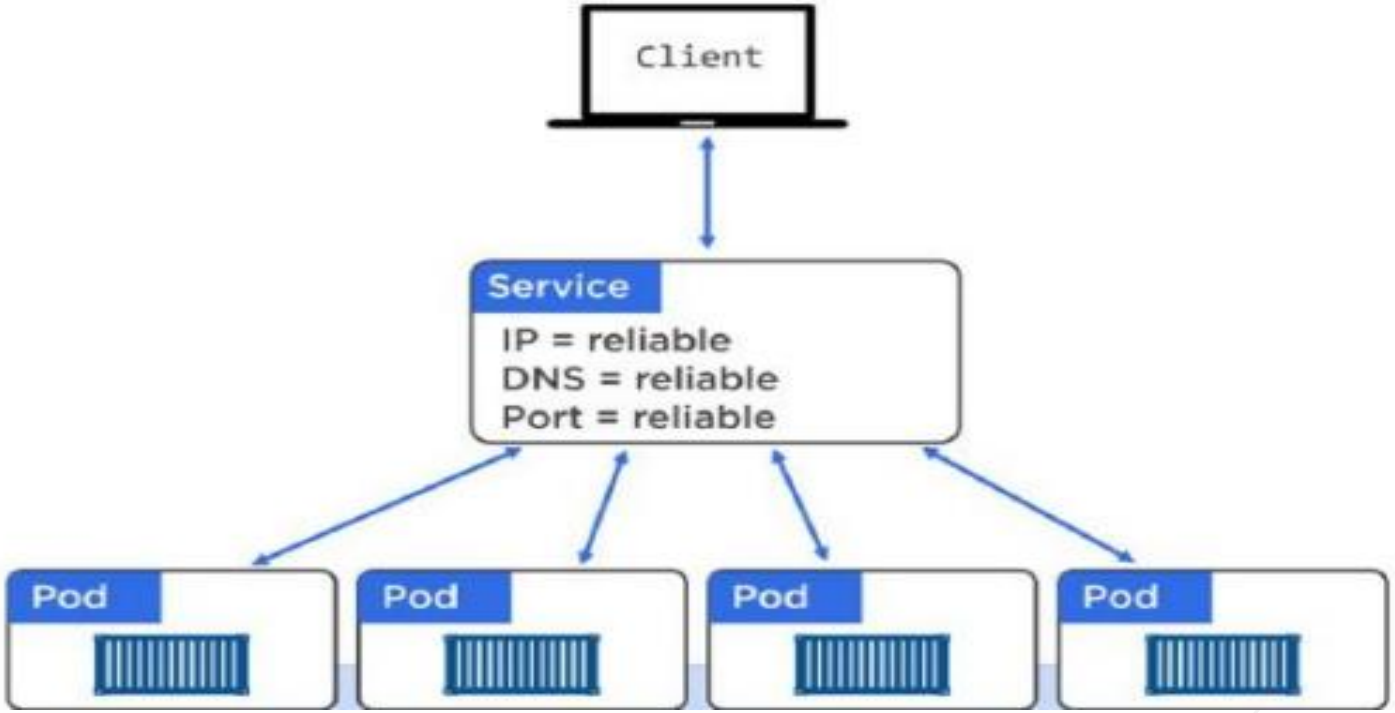
Services

Pods are not reliable



Services

A **Service** offers a single **DNS entry** for a containerized application managed by the Kubernetes cluster

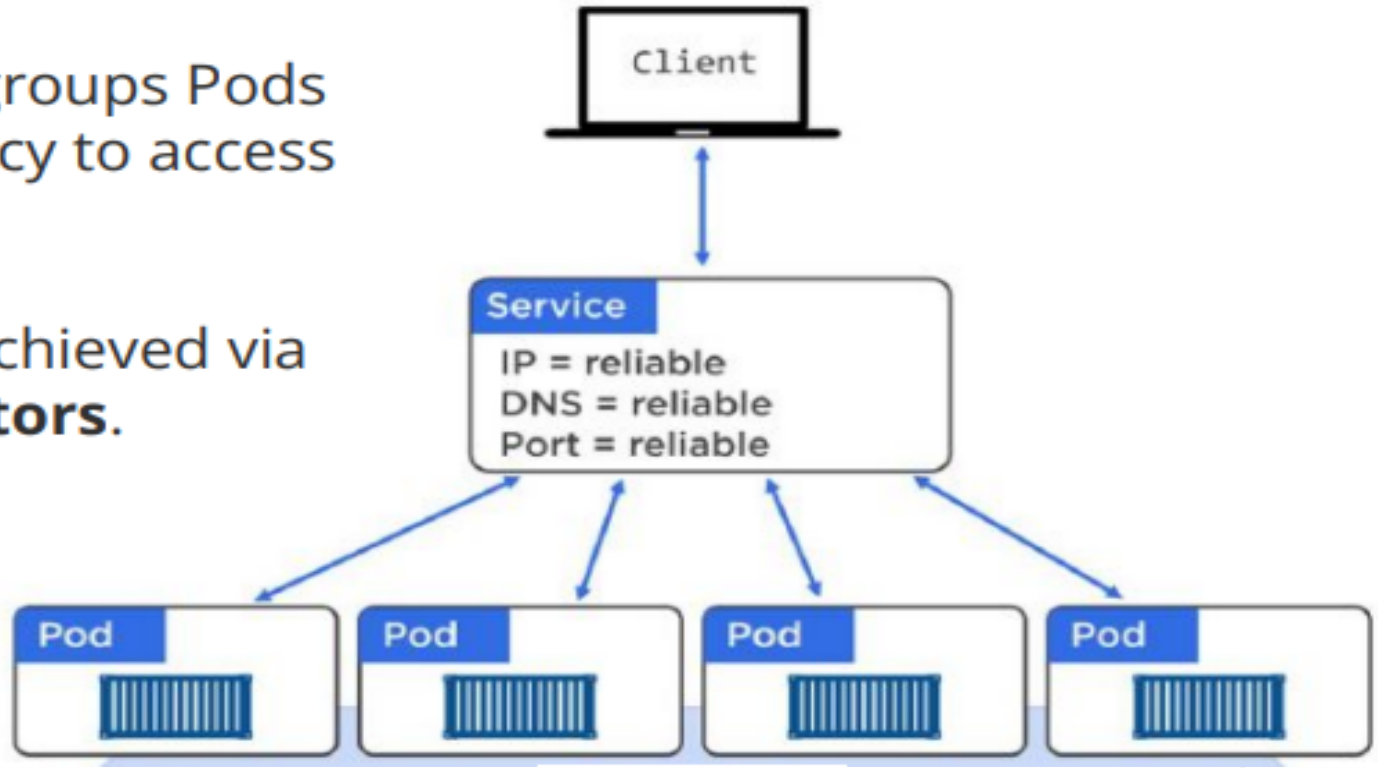


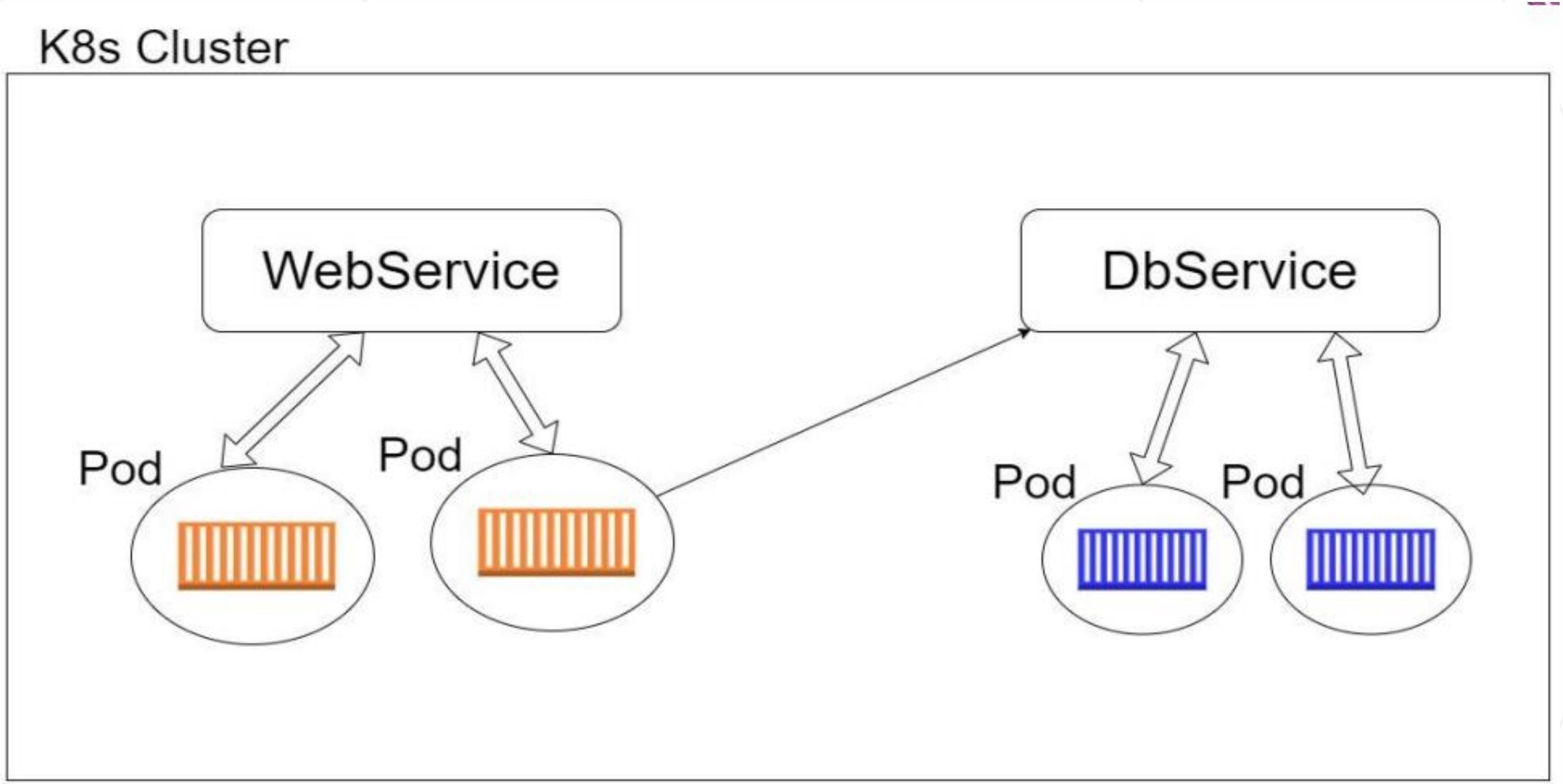
Services

The **Service** is associated with the Pods, and provides them with a stab IP, DNS and port. It also **loadbalances** requests across the Pods.

Service logically groups Pods and defines a policy to access them.

This grouping is achieved via **Labels** and **Selectors**.




K8s Cluster
WebService
DbService
Pod
Pod
Pod
Pod

kube-proxy

Each cluster node runs a daemon called **kube-proxy**

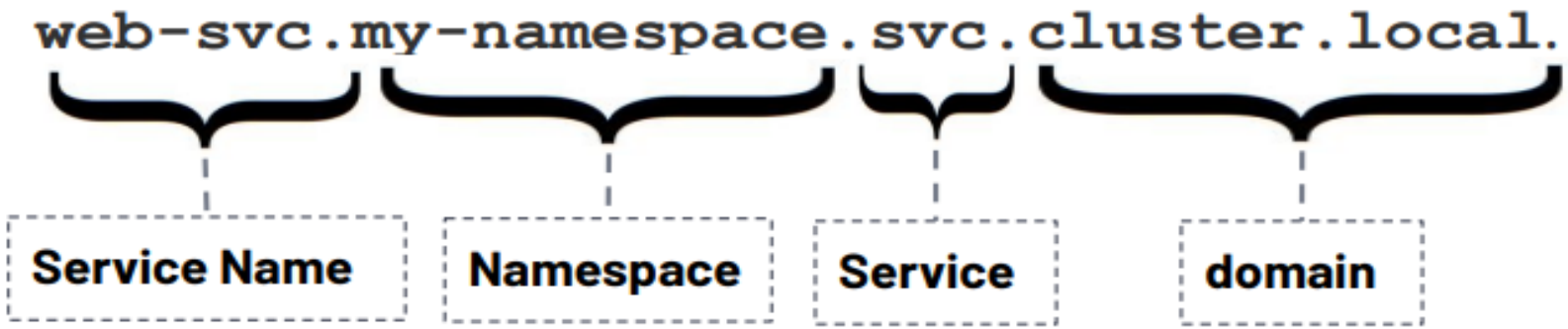
`kube-proxy` is responsible for ***implementing the Service configuration*** on behalf of an administrator or developer

For each new Service, on each node, **`kube-proxy`** configures **iptables** rules to capture the traffic for its **`ClusterIP`** and forwards it to one of the Service's endpoints.

When the Service is removed, **`kube-proxy`** removes the corresponding **iptables** rules on all nodes as well.

Service Discovery

Kubernetes has an add-on for **DNS**, which creates a DNS record for each Service and its format is



Services within the same Namespace find other Services just by their names.

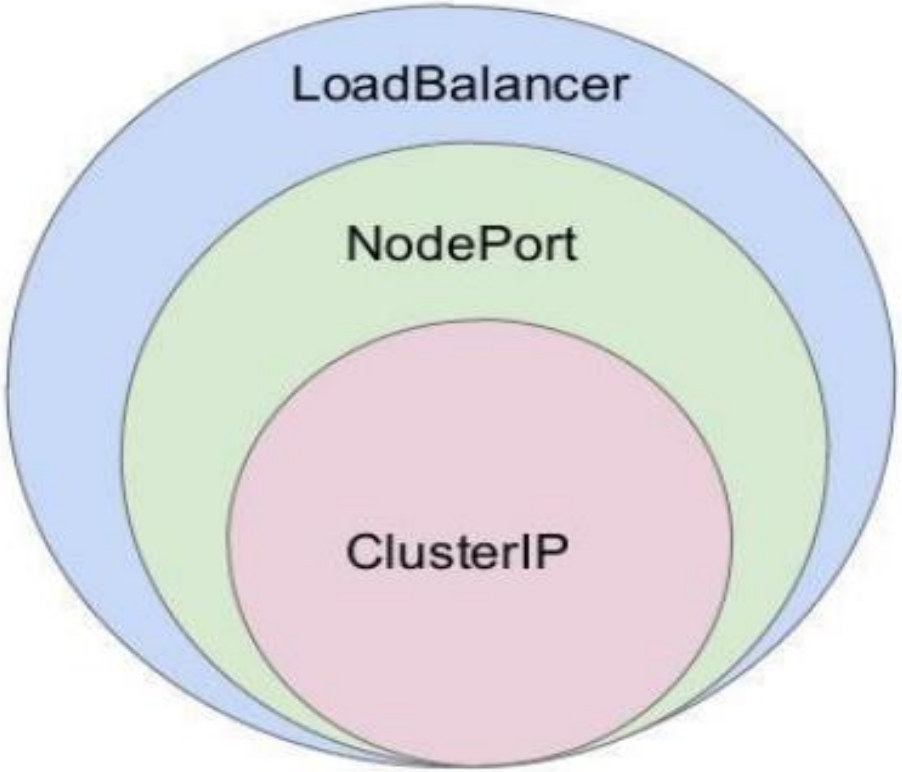
If we add a Service **`redis-master`** in **`my-ns`** Namespace, all Pods in the same **`my-ns`** Namespace lookup the Service just by its name, **`redis-master`**.

Service Types

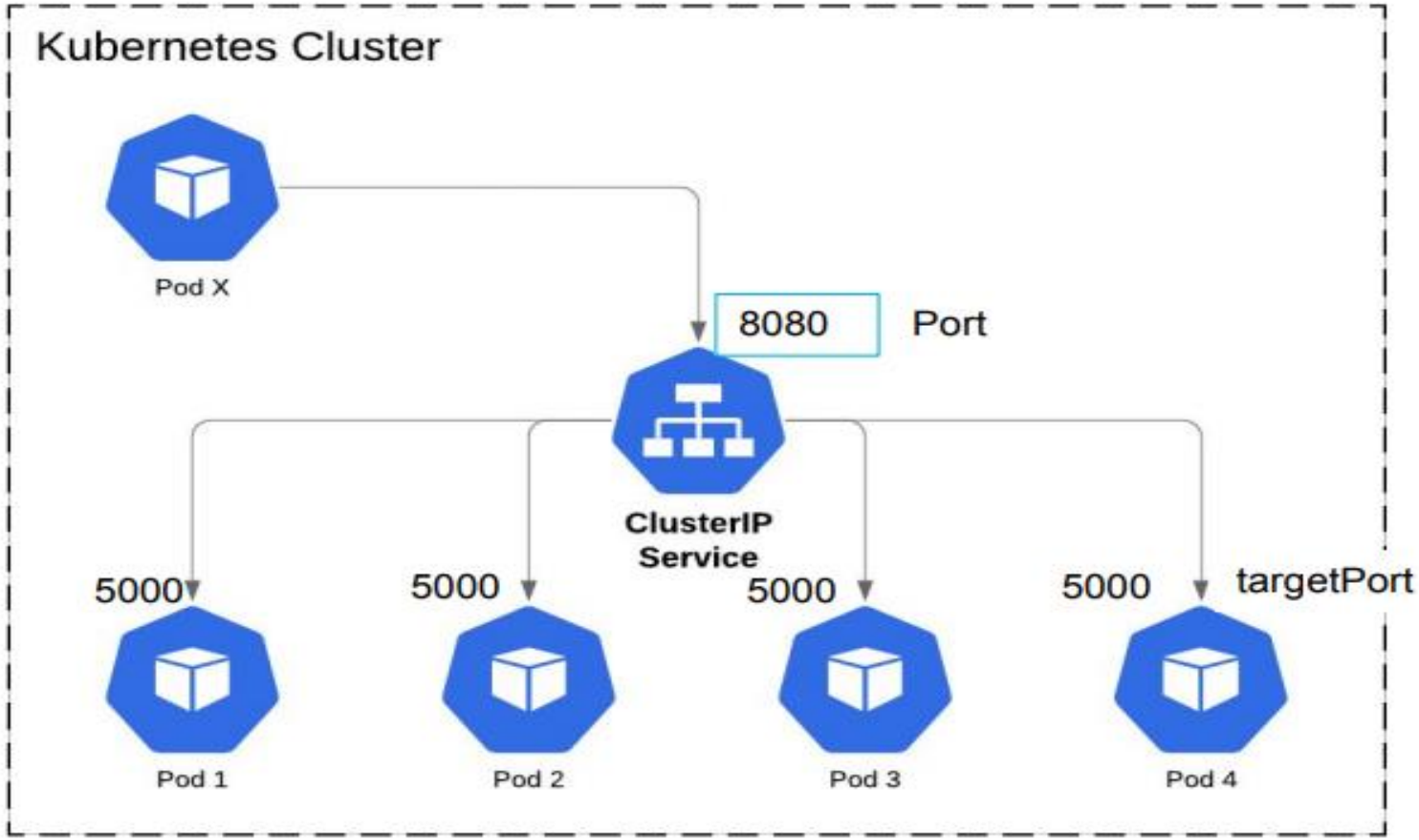
Service Types

There are 4 major service types:

- ClusterIP (default)
- NodePort
- LoadBalancer
- ExternalName



Service Types

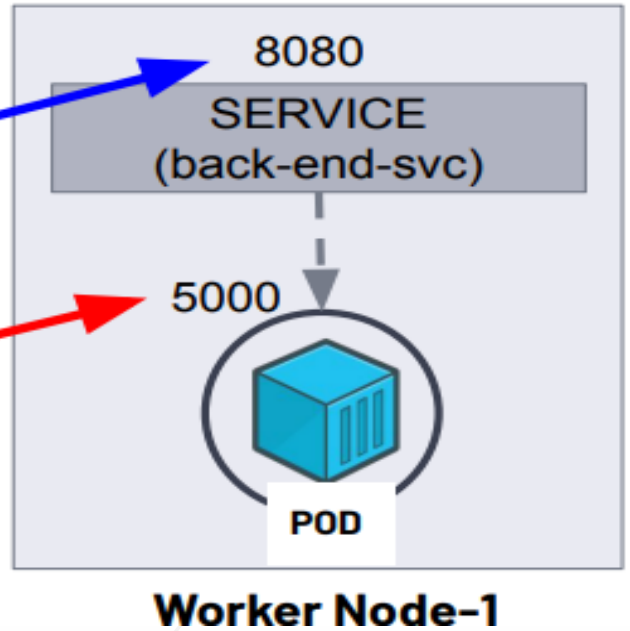


ClusterIP:
Expose traffic internally

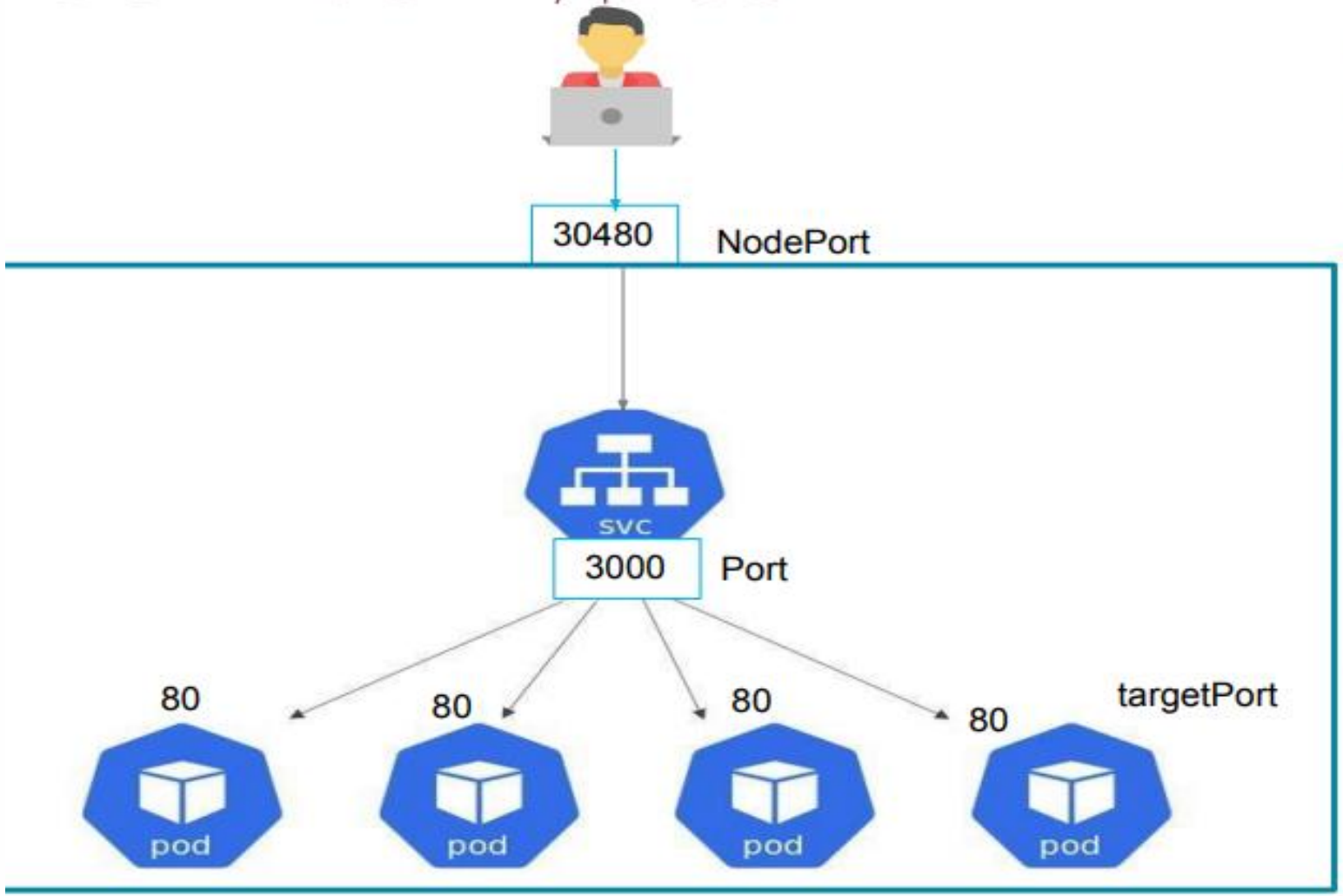
Example Usecase:

Good for service o
database & back-enc
apps.

```
apiVersion: v1
kind: Service
metadata:
 name: back-end-svc
 labels:
   app: back-end
spec:
 type: ClusterIP (default)
 selector:
   app: back-end
 ports:
 - port: 8080
   protocol: TCP
   targetPort: 5000
```



Service Types

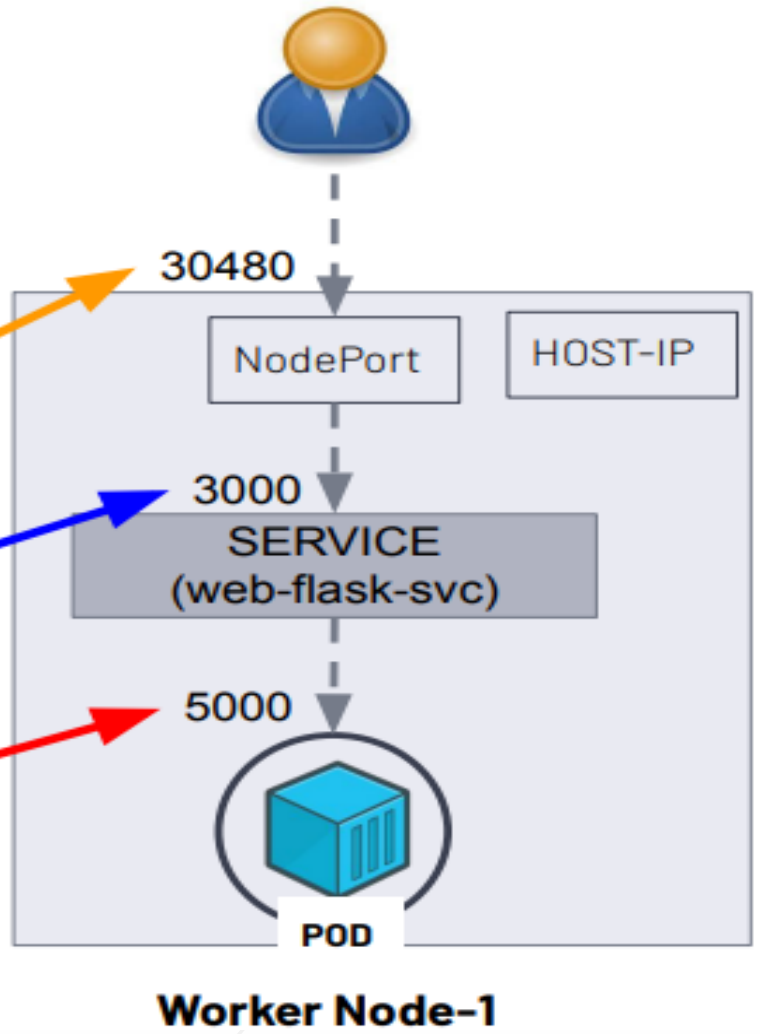


NodePort:
Exposes traffic to the outside.

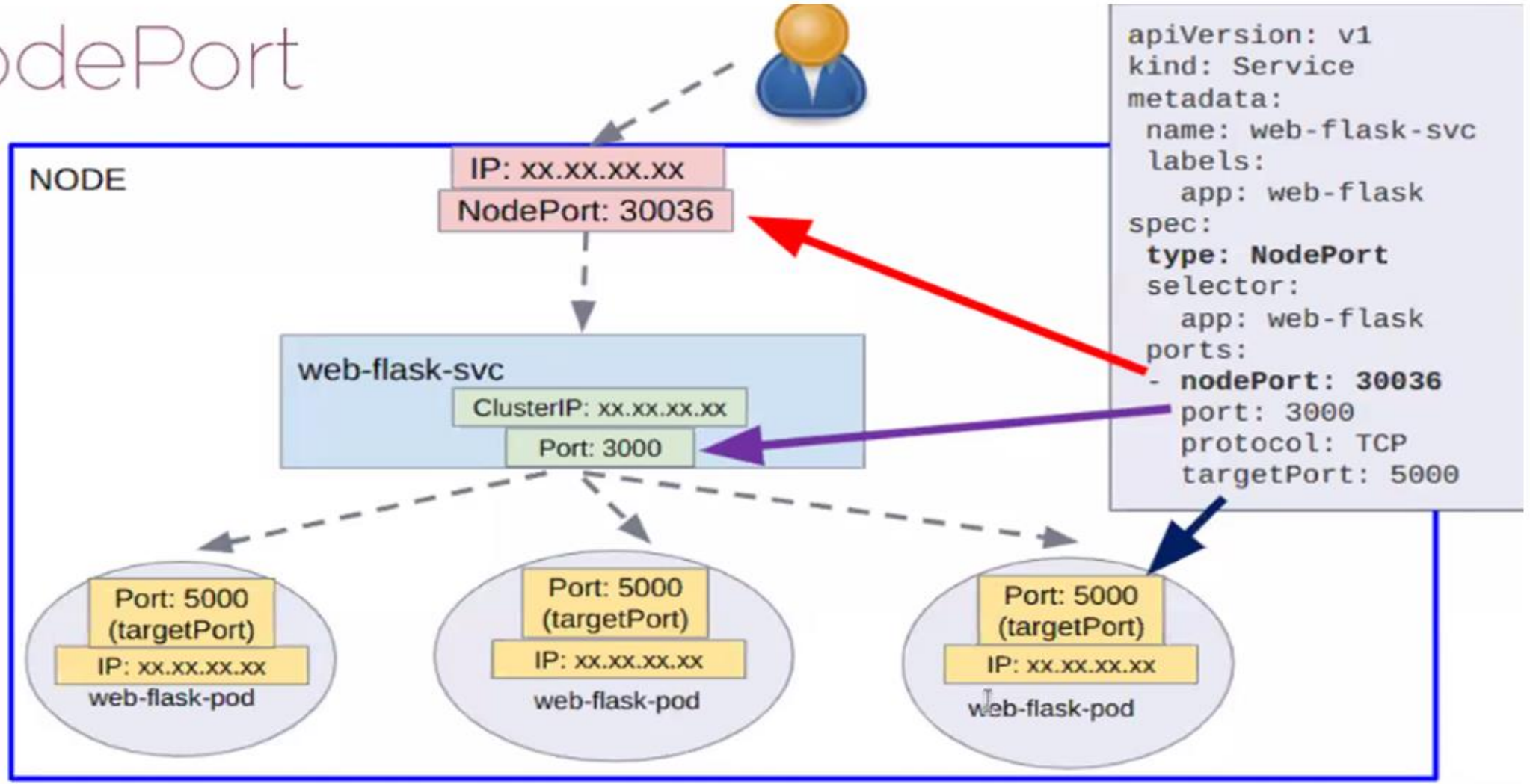
Example Use Case:

when we want to make our Services which has our app or website accessible from the external world.

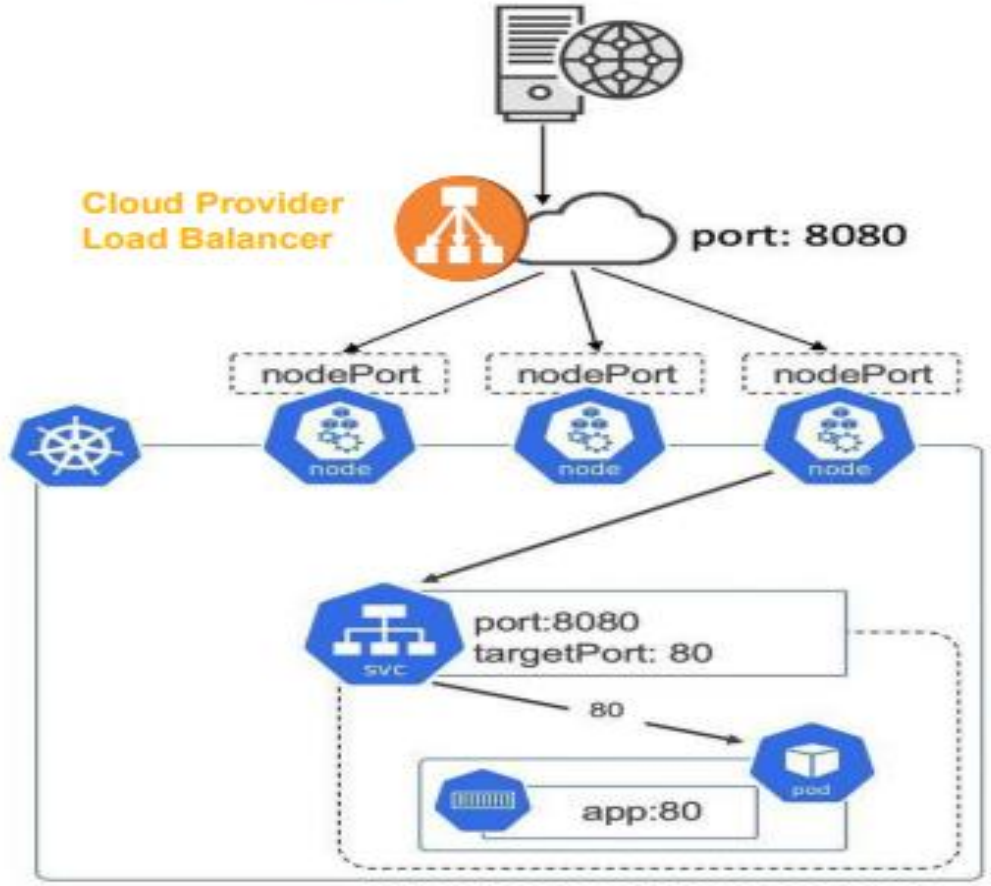
```yaml
apiVersion: v1
kind: Service
metadata:
  name: web-flask-svc
  labels:
    app: web-flask
spec:
  type: NodePort
  selector:
    app: web-flask
  ports:
  - nodePort: 30480
    port: 3000
    protocol: TCP
    targetPort: 5000
```



odePort


NODE
IP: xx.xx.xx.xx
NodePort: 30036
web-flask-svc
ClusterIP: xx.xx.xx.xx
Port: 3000
Port: 5000
(targetPort)
IP: xx.xx.xx.xx
web-flask-pod
Port: 5000
(targetPort)
IP: xx.xx.xx.xx
web-flask-pod
Port: 5000
(targetPort)
IP: xx.xx.xx.xx
web-flask-pod
apiVersion: v1
kind: Service
metadata:
name: web-flask-svc
labels:
app: web-flask
spec:
type: NodePort
selector:
app: web-flask
ports:
- nodePort: 30036
port: 3000
protocol: TCP
targetPort: 5000

Service Types

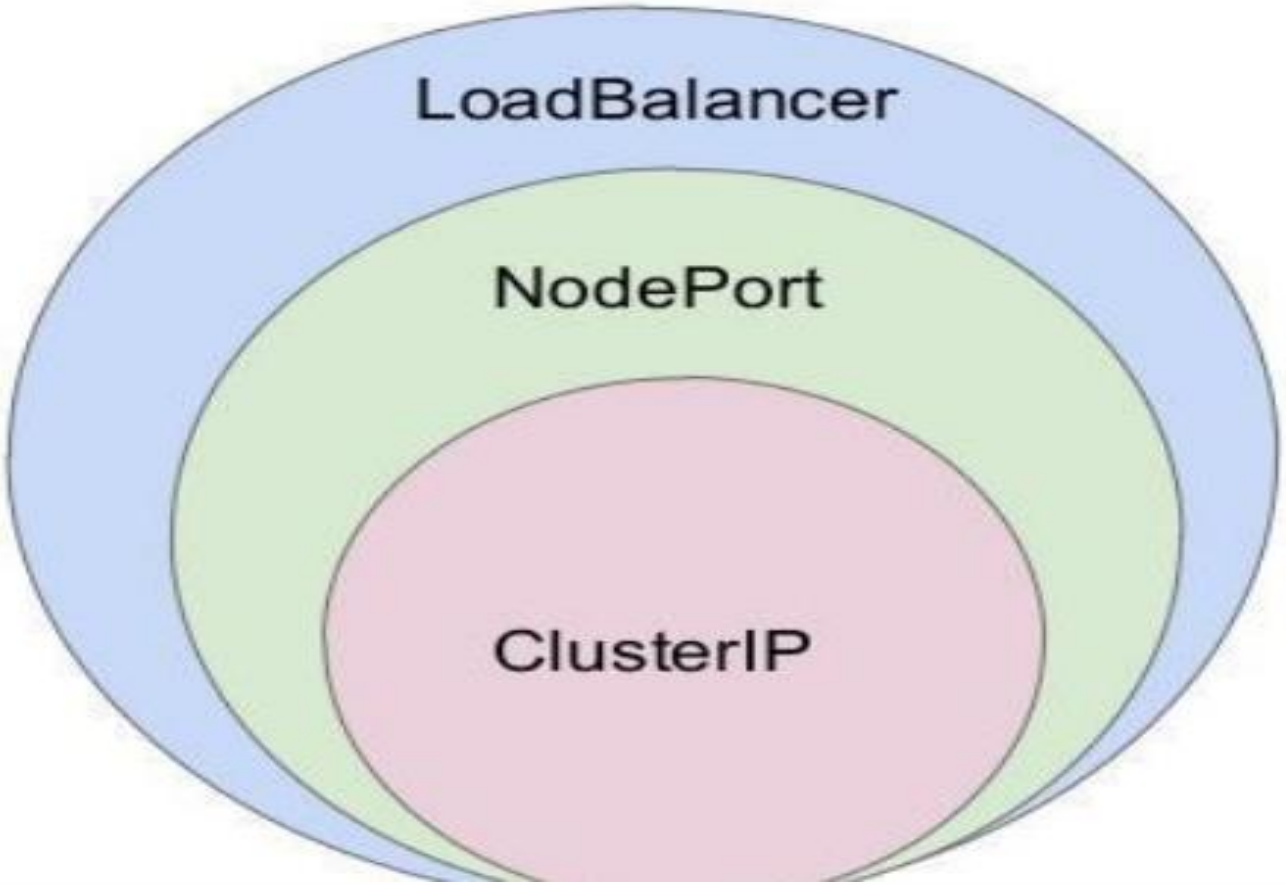


LoadBalancer:
Exposes traffic outside with load balancing feature.

Example Use Case:

when we want to load balance our Services which has our app or website accessible from the external world.

Service Types



Service Types

ExternalName:

Maps the Service to the contents of the ExternalName field (e.g. example.com), by returning a CNAME record with its value.

Example Use Cases:

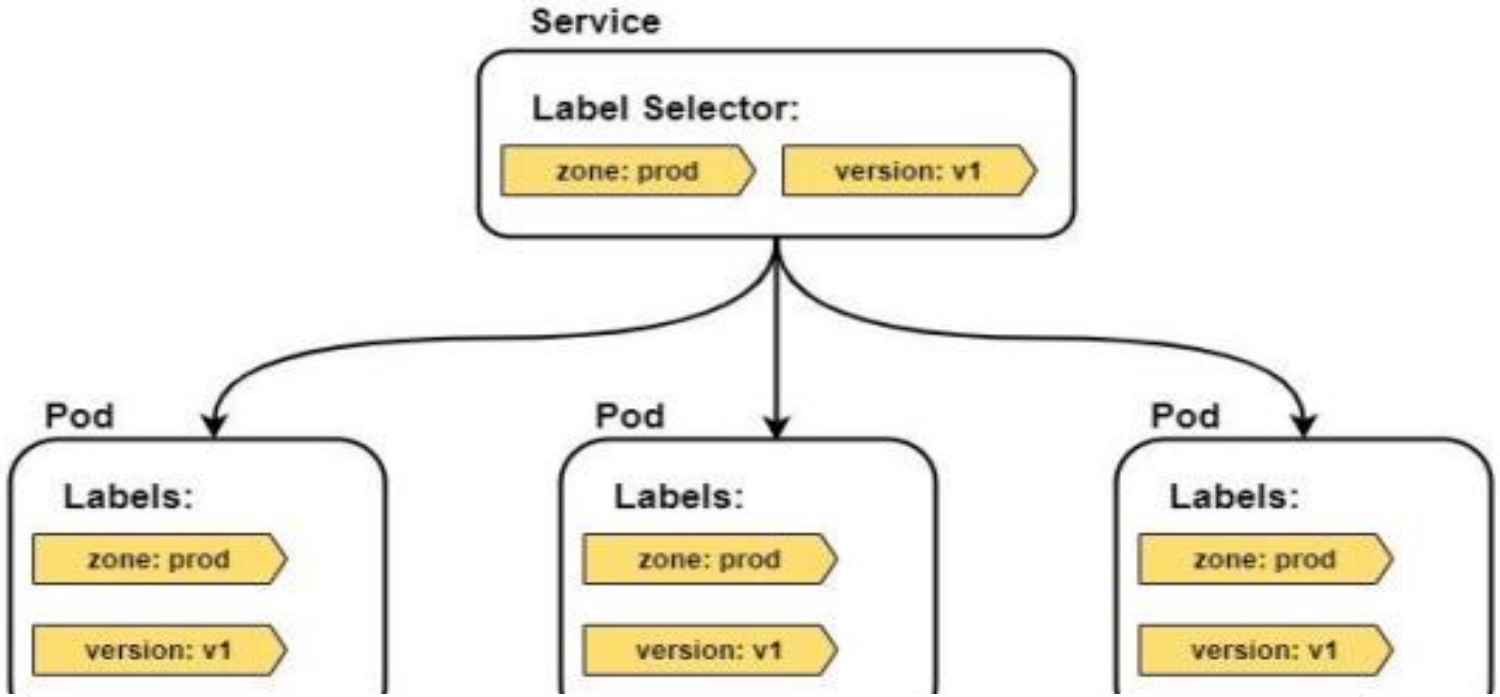
to make externally configured services like;

remote.server.url.com

available to applications inside the cluster.

Labels and loose coupling

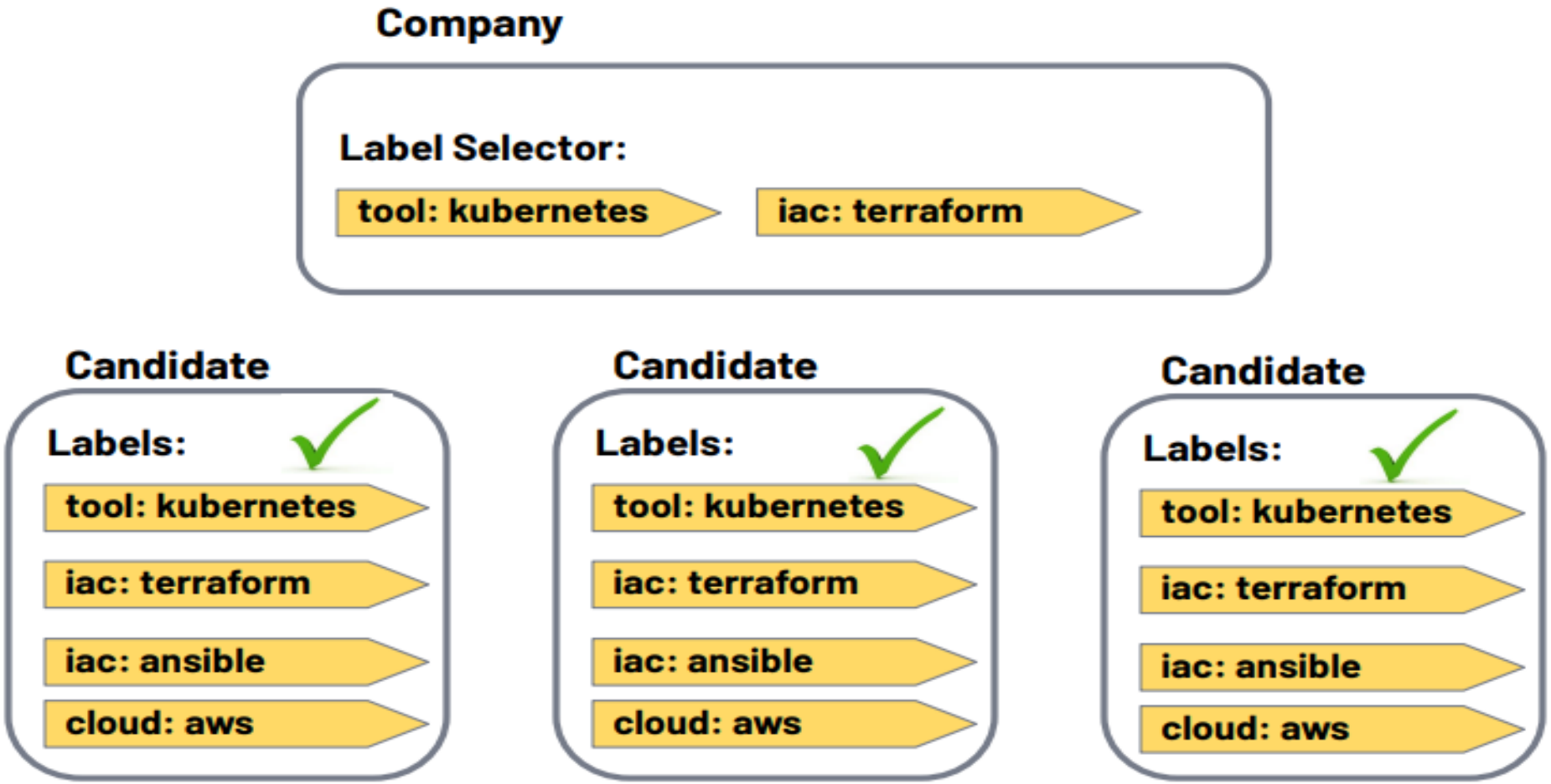
The figure below shows an example where 3 Pods are labeled as **`zone=prod`** and **`version=v1`**, and the Service has a label selector that matches. This Service provides stable networking to all three Pods. It also provides simple load-balancing.



Labels and loose coupling

The figure below shows an example where the Service does not match any of the Pods. This is because the Service is selecting on two labels, but the Pods only have one of them. The logic behind this is a Boolean AND operation.

Labels and loose coupling



- Pod to Pod
  - Using networking plugins
- CNI (Container Network Interface)

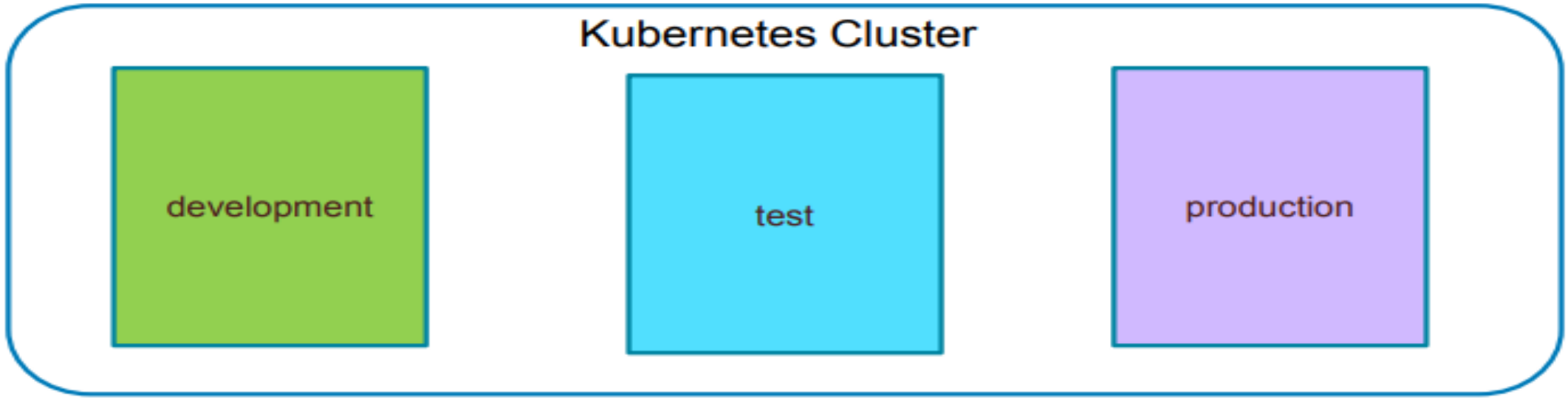
It is a framework for dynamically configuring networking resources.

Some common plugins: Calico, Flannel

flannel

Namespaces

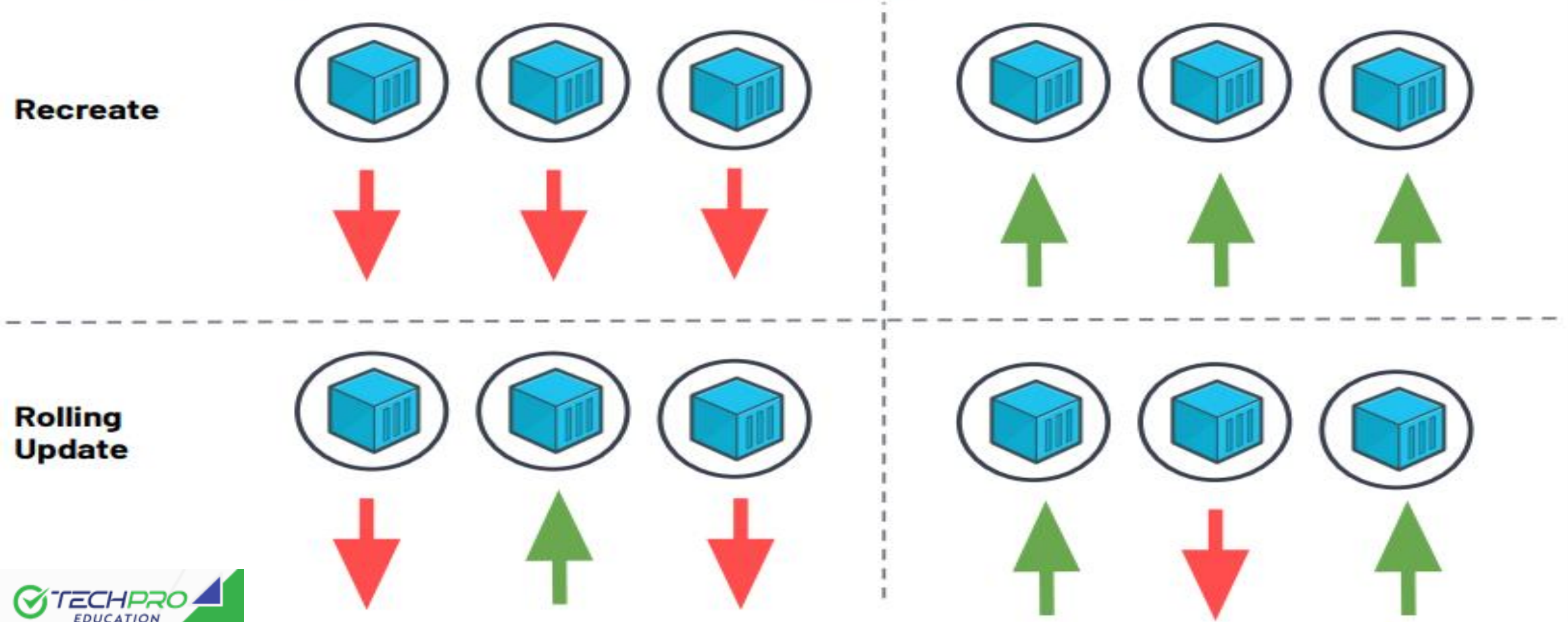
- Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called **namespaces**.
- Namespaces are intended for use in environments with many users spread across multiple teams, or projects.



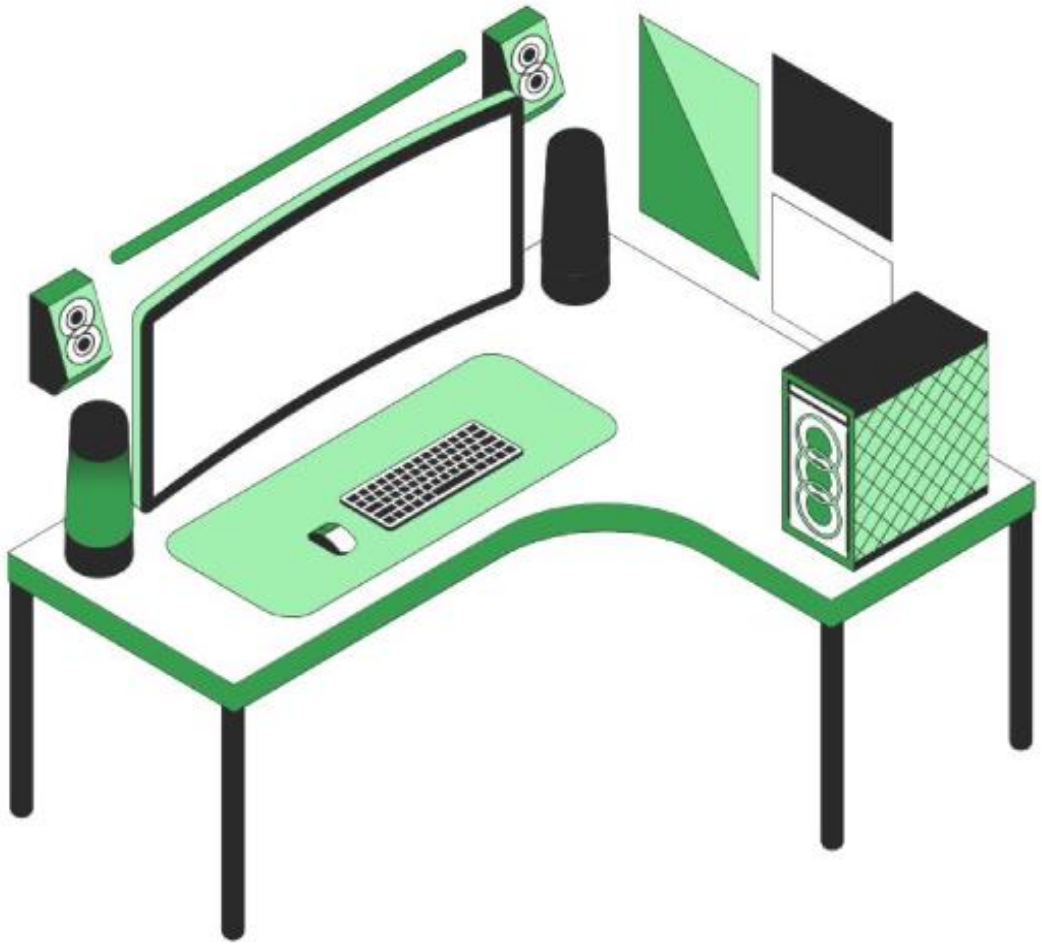
Pod Selector

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: nginx-rs
 labels:
   environment: dev
spec:
 replicas: 3
 selector:
   matchLabels:
     app: nginx
 template:
   metadata:
     labels:
       app: nginx
   spec:
     containers:
     - name: mynginx
       image: nginx:1.19
       ports:
       - containerPort: 80
```

Deployment Strategy



TECHPRO EDUCATION



Do you have any questions?

Send it to us! We hope you learned something new.