BATCH : Batch 85

LESSON : **Kubernetes-3**

DATE : 11.11.2022

SUBJECT : **Kubernetes Networking**

techproeducation

techproeducation

techproeducation

techproeducation

techproedu

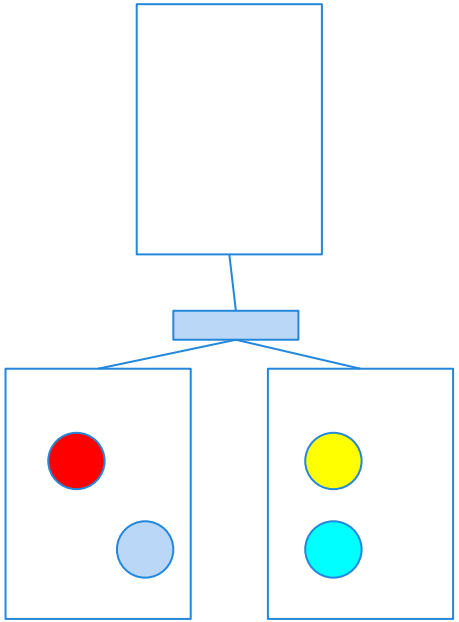
Kubernetes Networking

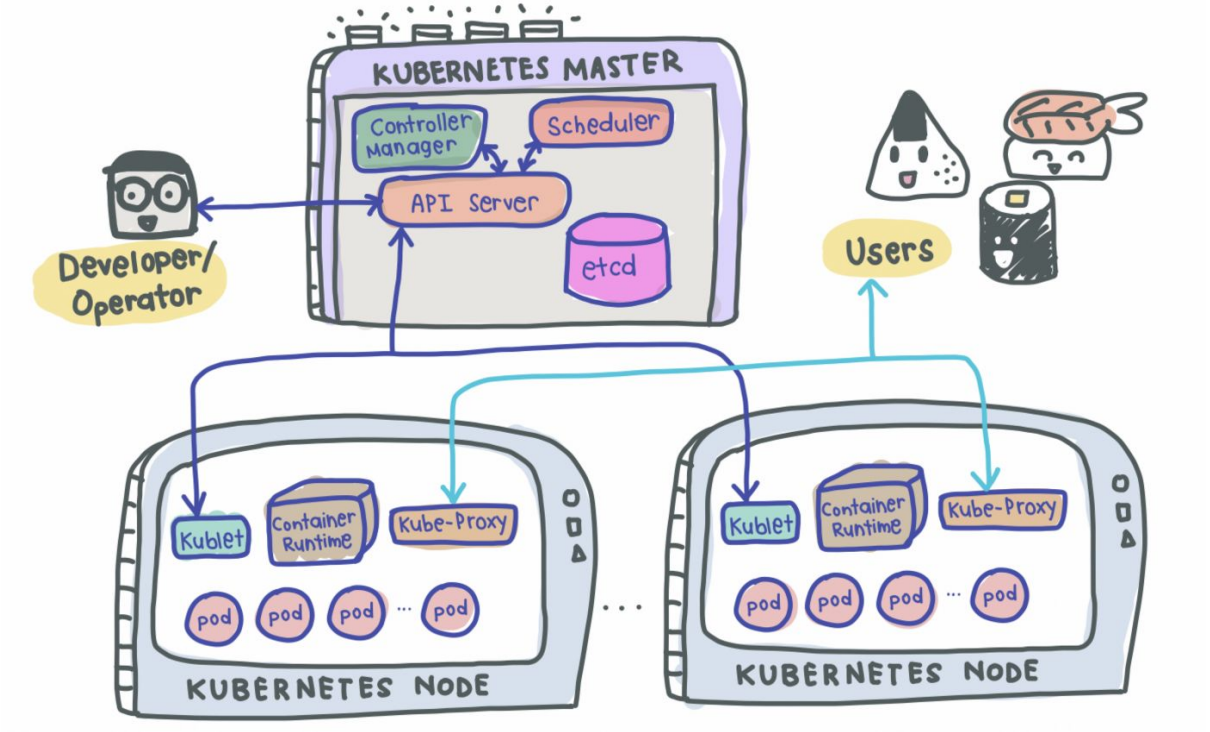
Cluster Networking

Service Types

NETWORKING

Cluster Networking





Cluster Networking

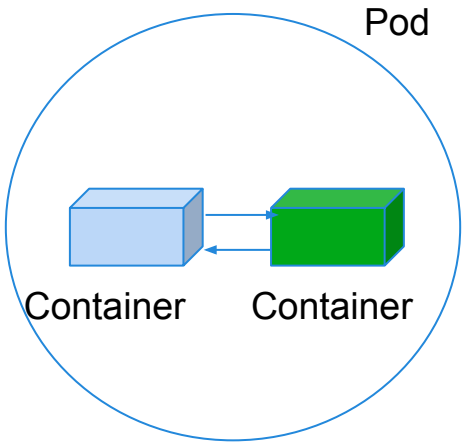
There are 4 distinct networking problems to address:

- Container-to-container communications:
- Pod-to-Pod communications:
- Pod-to-Service communications: this is covered by services.
- External-to-Service communications: this is covered by services.

Cluster Networking

- Container to Container

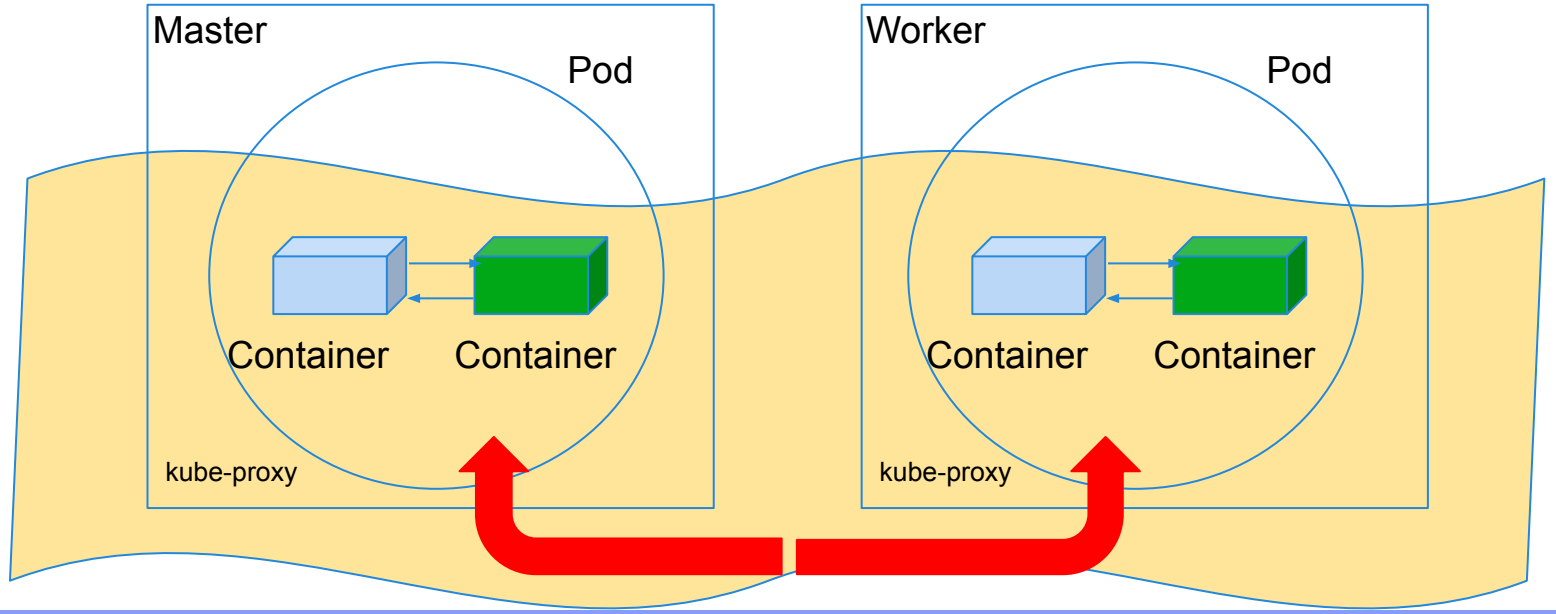
  Using “localhost:port”



Cluster Networking

- Pod to Pod

  Using networking plugins



Cluster Networking

- Pod to Pod

  Using networking plugins

  - CNI (Container Network Interface)

It is a framework for dynamically configuring networking resources.

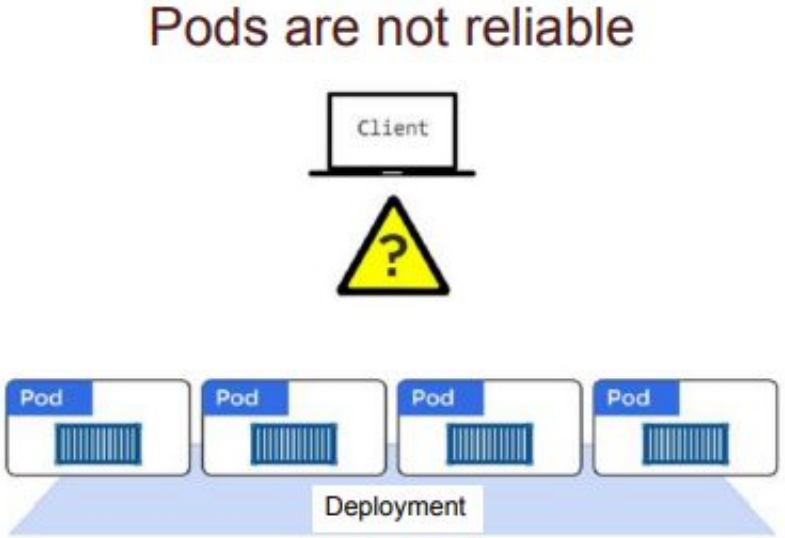
Some common plugins:

Calico, Flannel

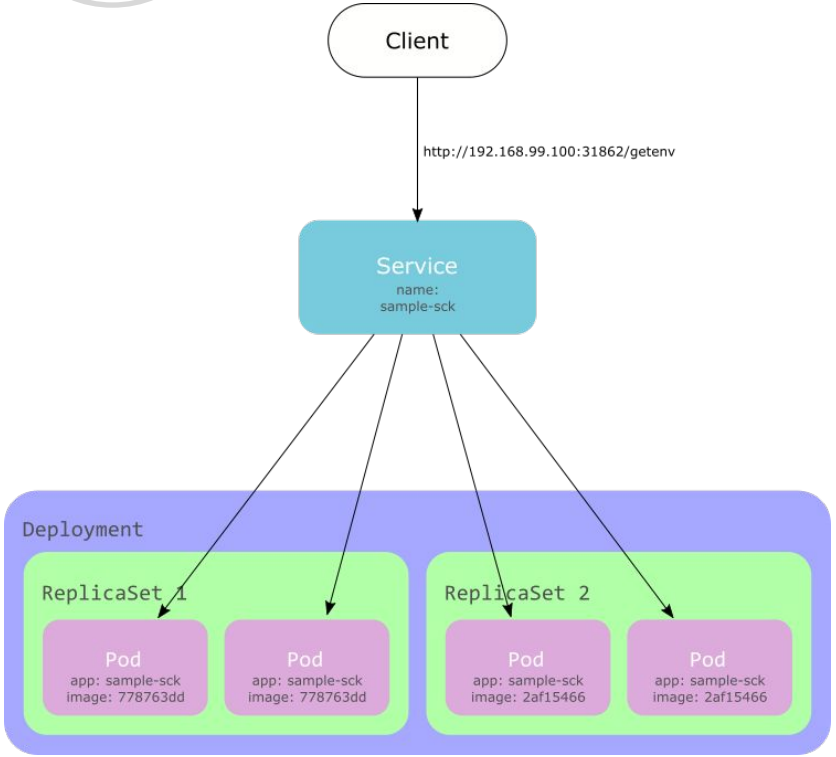
flannel

Services

Thanks to plugins, pods can communicate over IP addresses, however ..

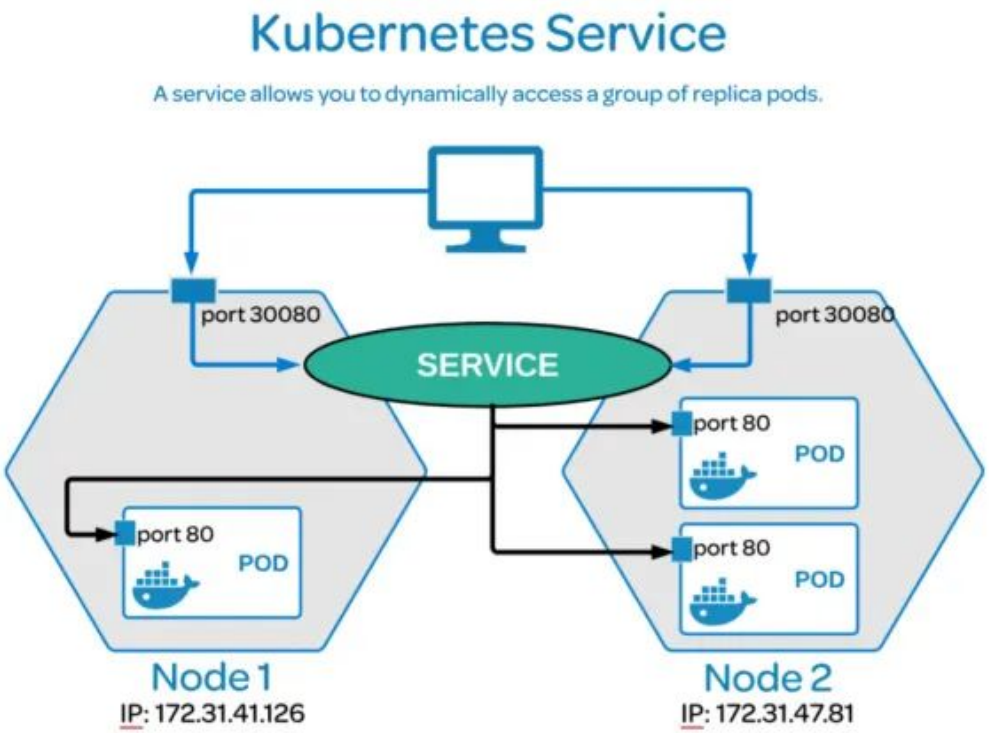


Services



- A Service offers a single DNS entry for a containerized application managed by the Kubernetes cluster
- The Service is associated with the Pods, and provides them with a stable 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.

Services



- Kubernetes Services enable communication between various components within and outside of the application.
- Kubernetes Services helps us connect applications together with other applications or users

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

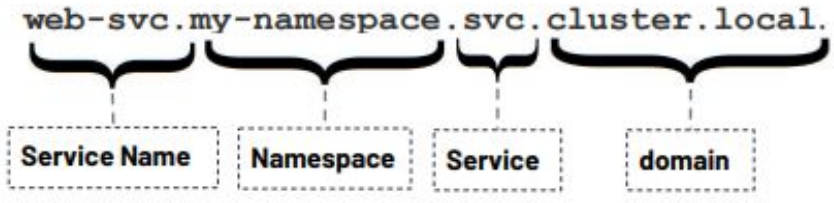
→For each new Service, on each node, kube-proxy configures **iptables rules** to capture the traffic for its ClusterIP and forwards it to one of the Service's endpoints.

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

→If **kube-proxy fails** the node goes into **Not Ready** state

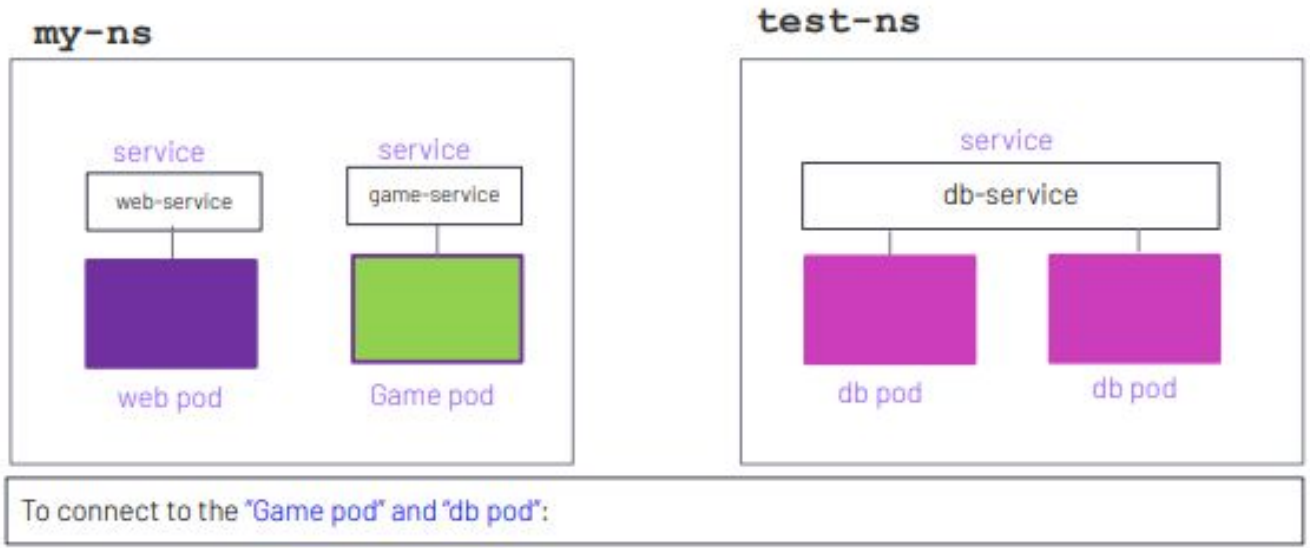
Service Discovery

➔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.



To connect to the "Game pod" and "db pod":

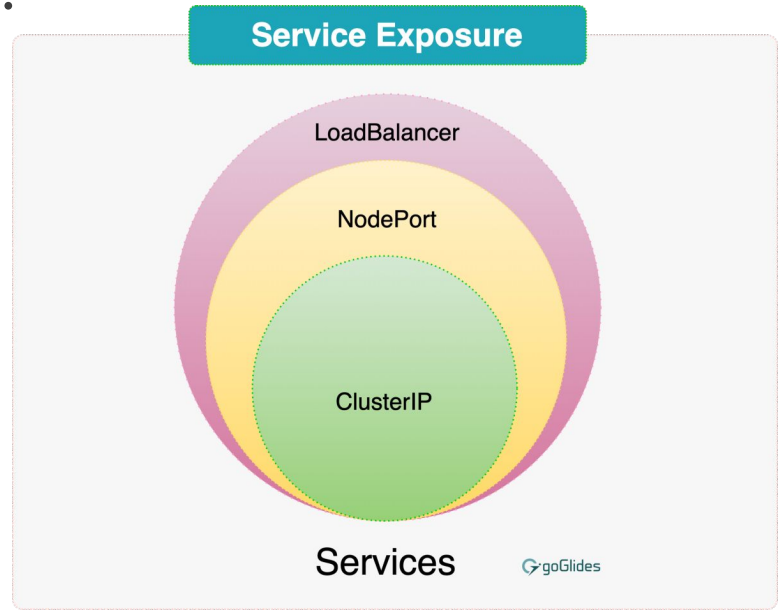
From " web pod" -> " Game pod" —> hostname: game-sevice.my-ns:port
game-service:port

From " web pod" -> "db pod —> hostname: db-service.test-ns.svc.cluster.local:port

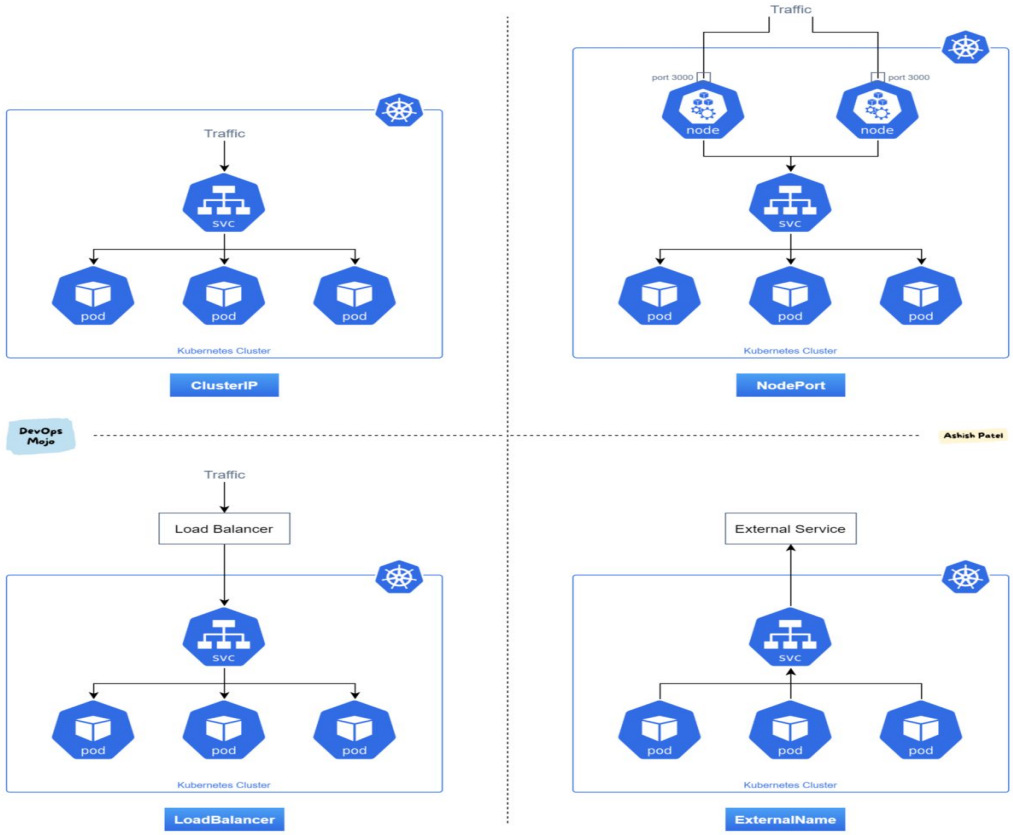
Service Types

There are 4 major service types:

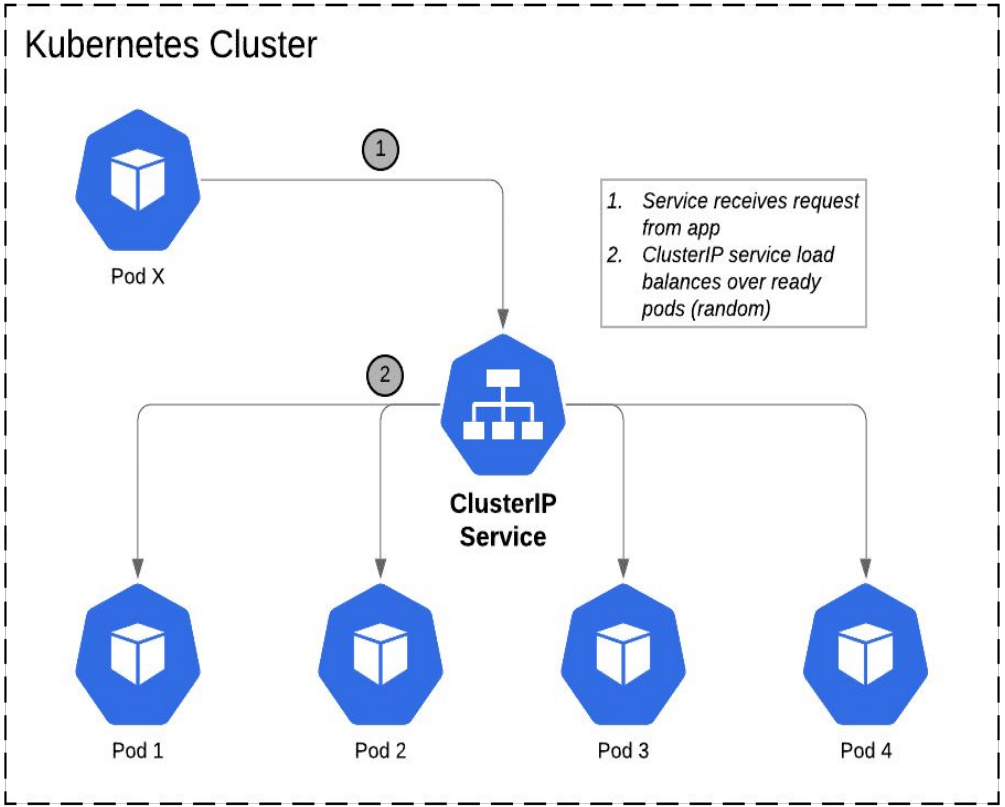
- ClusterIP (default)
  - **Network inside cluster**
- NodePort
  - **Network coming from Internet, usually for Frontend**
- LoadBalancer
  - **Used by cloud provider**
- ExternalName



Service Types



Services


Kubernetes Cluster
Pod X
1
1. Service receives request from app
2. ClusterIP service load balances over ready pods (random)
2
ClusterIP Service
Pod 1
Pod 2
Pod 3
Pod 4

ClusterIP Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  selector:
    app: nginx
    env: prod
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

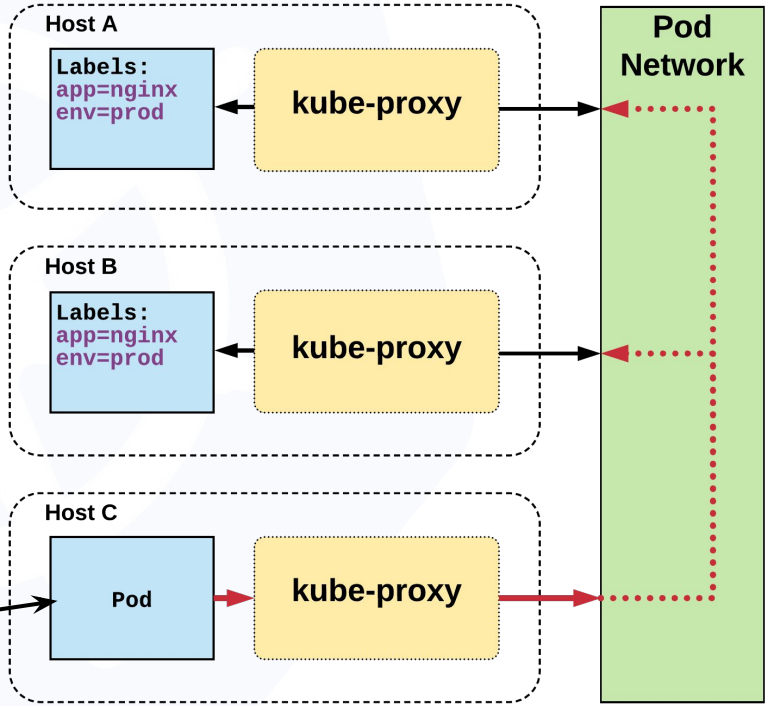
➔ ClusterIP services exposes a service on a strictly cluster internal virtual IP.

Cluster IP Service

```
Name:           example-prod
Selector:       app=nginx,env=prod
Type:           ClusterIP
IP:             10.96.28.176
Port:           <unset>  80/TCP
TargetPort:     80/TCP
Endpoints:      10.255.16.3:80,
                10.255.16.4:80
```

```
/ # nslookup example-prod.default.svc.cluster.local

Name:      example-prod.default.svc.cluster.local
Address 1: 10.96.28.176 example-prod.default.svc.cluster.local
```

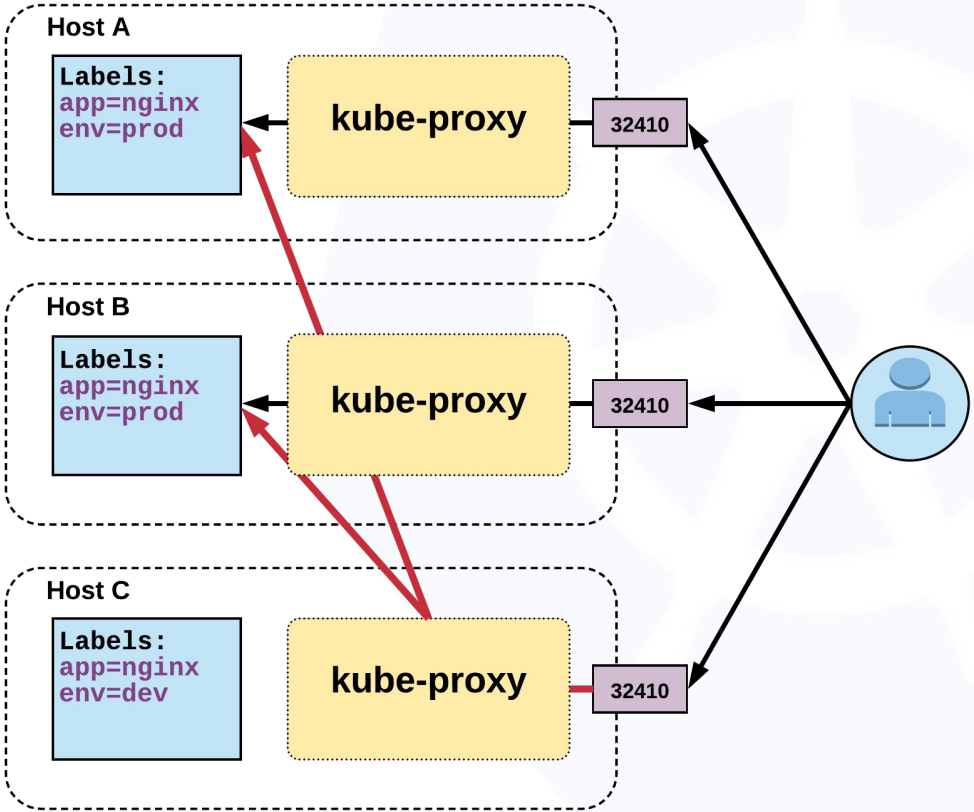


NodePort Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: NodePort
  selector:
    app: nginx
    env: prod
  ports:
  - nodePort: 32410
    protocol: TCP
    port: 80
    targetPort: 80
```

- NodePort services extend the ClusterIP service.
- Exposes a port on every node's IP.
- Port can either be statically defined, or dynamically taken from a range between 30000-32767.

NodePort Service



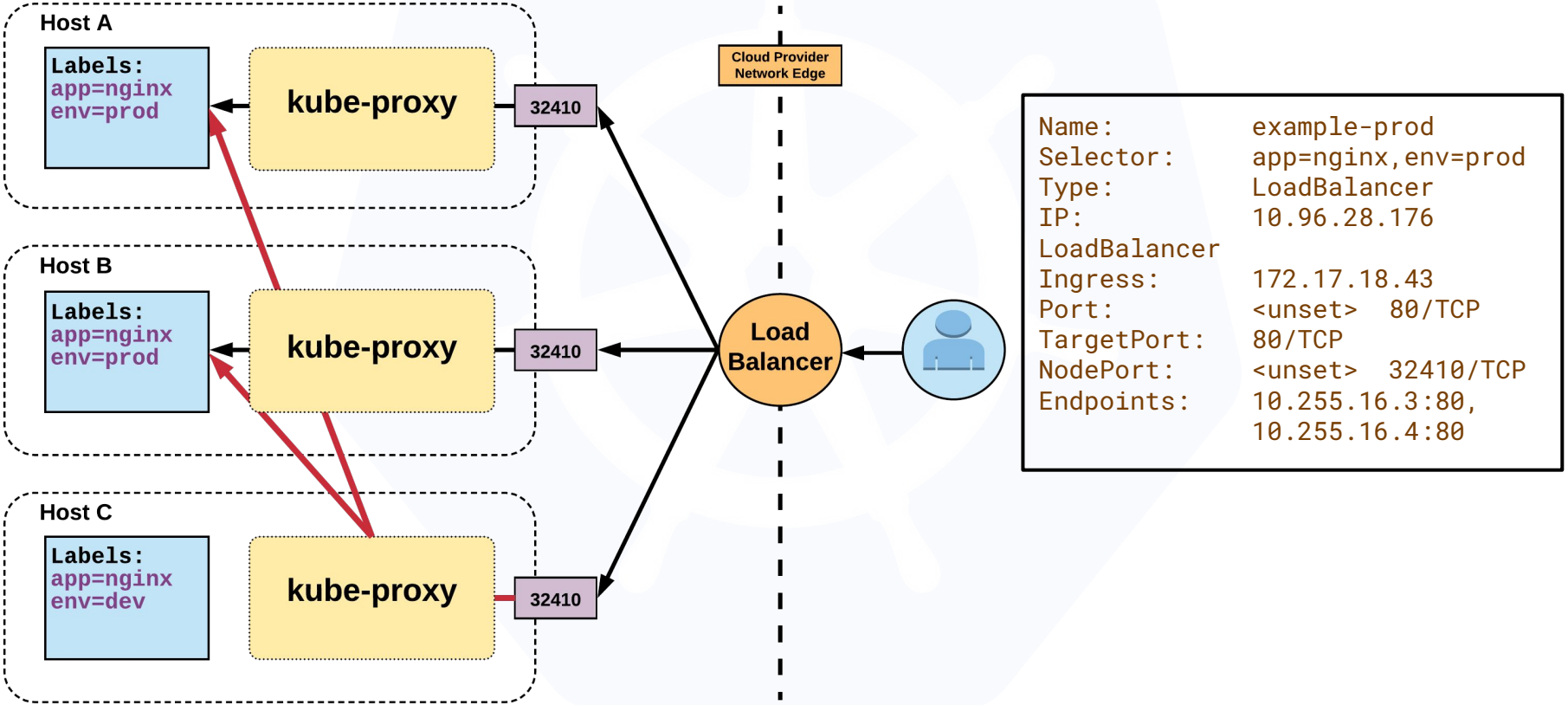

```
Name:          example-prod
Selector:      app=nginx,env=prod
Type:          NodePort
IP:            10.96.28.176
Port:          <unset>  80/TCP
TargetPort:    80/TCP
NodePort:      <unset>  32410/TCP
Endpoints:     10.255.16.3:80,
               10.255.16.4:80
```

LoadBalancer Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: LoadBalancer
  selector:
    app: nginx
    env: prod
  ports:
    protocol: TCP
    port: 80
    targetPort: 80
```

- **LoadBalancer** services extend **NodePort.**
- Works in conjunction with an external system (cloud providers) to map a cluster external IP to the exposed service.

LoadBalancer Service



ExternalName Service

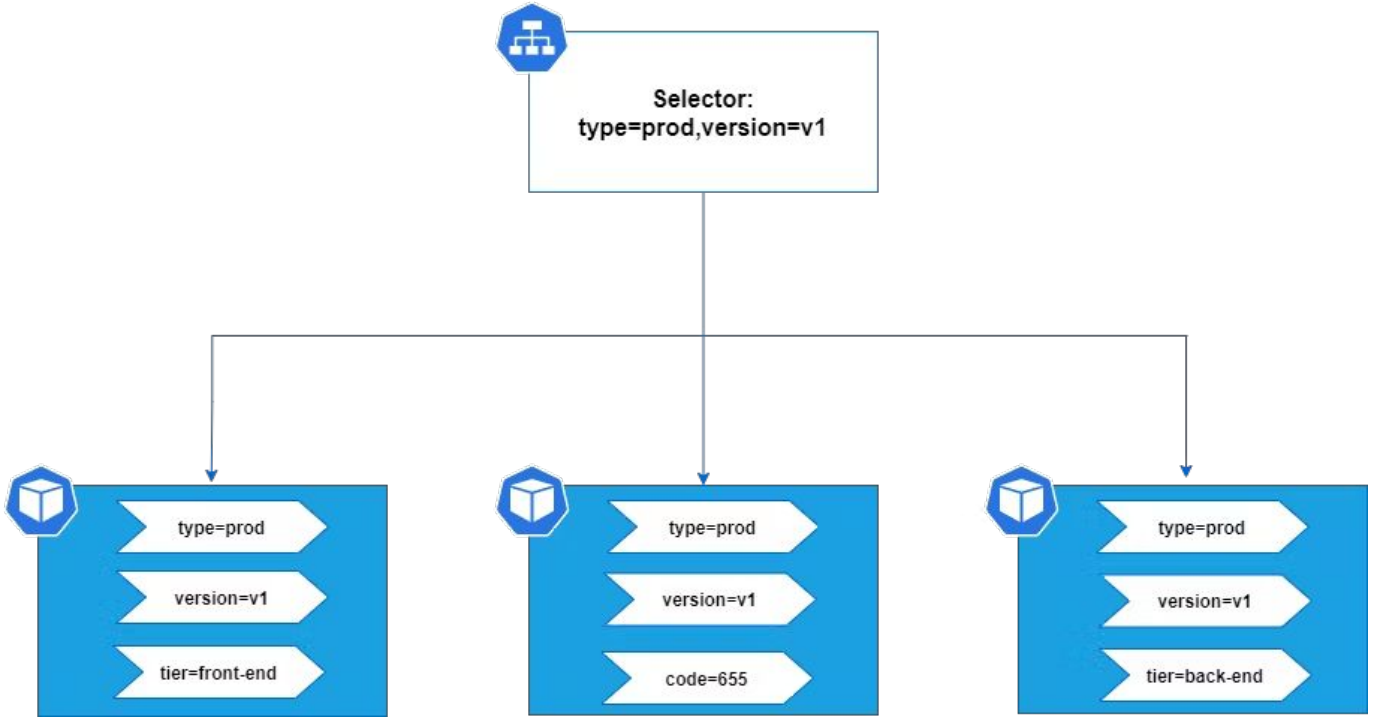
- **ExternalName** is used to reference endpoints **OUTSIDE** the cluster.
- Creates an internal **CNAME** DNS entry that aliases another.

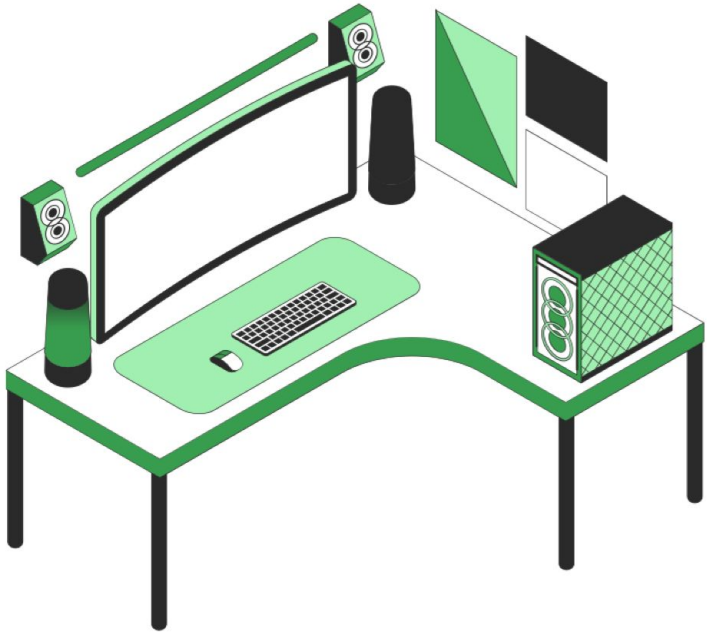
```yaml
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: ExternalName
spec:
  externalName: example.com
```

Labels and loose coupling

- Labels and Selectors use a key/value pair format.
- Pods and Services are loosely coupled via labels and label selectors.
- For a Service to match a set of Pods, and therefore provide stable networking and load-balance, it only needs to match some of the Pods labels.
- However, for a Pod to match a Service, the Pod must match all of the values in the Service's label selector.

Labels and loose coupling





Do you have any questions?

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