Deployment: Manifest File

nginx-deployment-withrolling.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    environment: test
  name: testdeploy
spec:
  replicas: 3
  selector:
    matchLabels:
      environment: test
  minReadySeconds: 10
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      labels:
        environment: test
    spec:
      containers:
      - image: nginx:1.16
        name: nginx
```

Manages pods with label *environment:test*
and manages replicaset defined within

Manages pods with label
environment:test

Nginx:1.16 *environment:test*

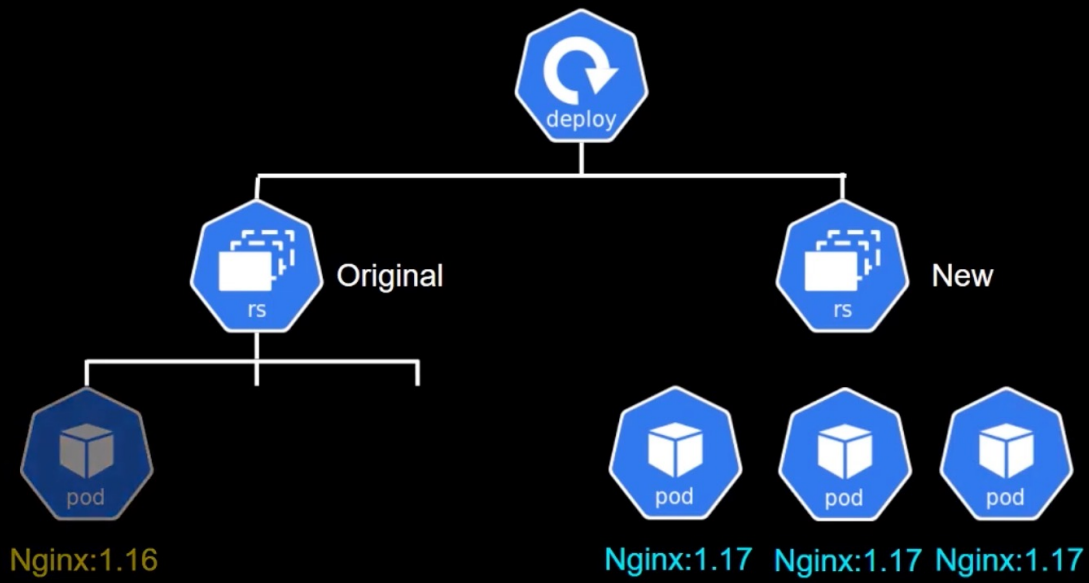
Nginx:1.16 *environment:test*

Nginx:1.16 *environment:test*

Deployment Rolling Update

nginx-deployment-withrolling.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    environment: test
  name: testdeploy
spec:
  replicas: 3
  selector:
    matchLabels:
      environment: test
  minReadySeconds: 10
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      labels:
        environment: test
    spec:
      containers:
      - image: nginx:1.17
        name: nginx
```



Kubernetes Namespaces

- Namespace as a virtual cluster inside your Kubernetes cluster
- Default namespace
- Creating Namespaces
- Viewing Namespaces
- Creating Resources in the Namespace
- Viewing resources in the Namespace
- Managing test, staging, and production environments within the same cluster

Kubernetes Services

- Exposing PODs to the outside world. Services are required for discovery.
- Labels and selectors are used to route to the appropriate POD
- Service Types
  - ClusterIP - Services makes internal pod accessible
  - NodePort - Allow external user traffic and load balancing
  - LoadBalancer - Service does load balancing with external load balancer
  - Ingress - Path based routing

Kubernetes Services

```yaml
kind: Service
apiVersion: v1

metadata:
  name: hostname-service

spec:
  type: NodePort
  selector:
    app: echo-hostname
  ports:
    - nodePort: 30163
      port: 8080
      targetPort: 80
```

Make the service available to network requests from external clients

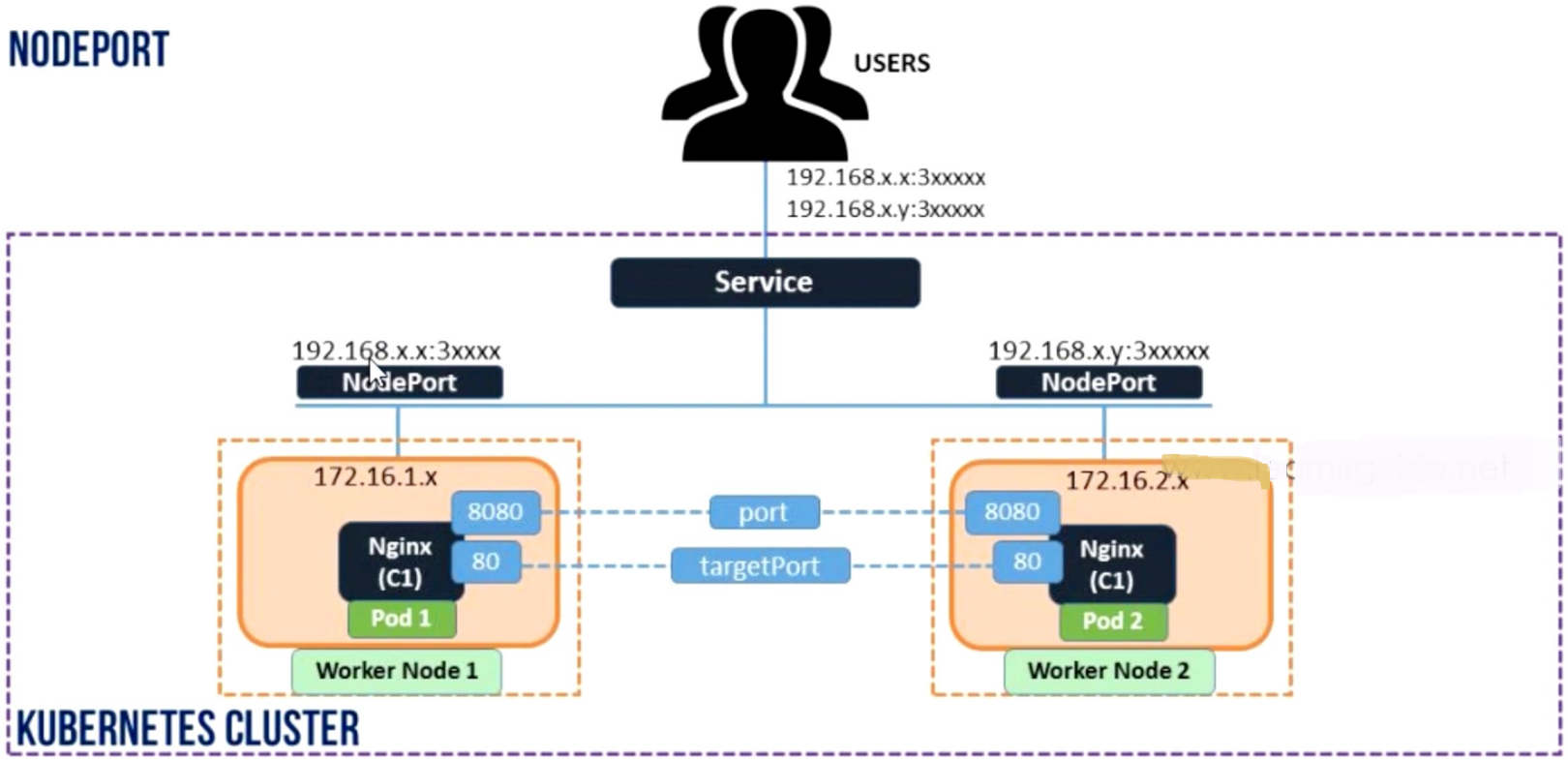
Forward requests to pods with label of this value

`nodePort`
access service via this external port number

`port`
port number exposed internally in cluster

`targetPort`
port that containers are listening on

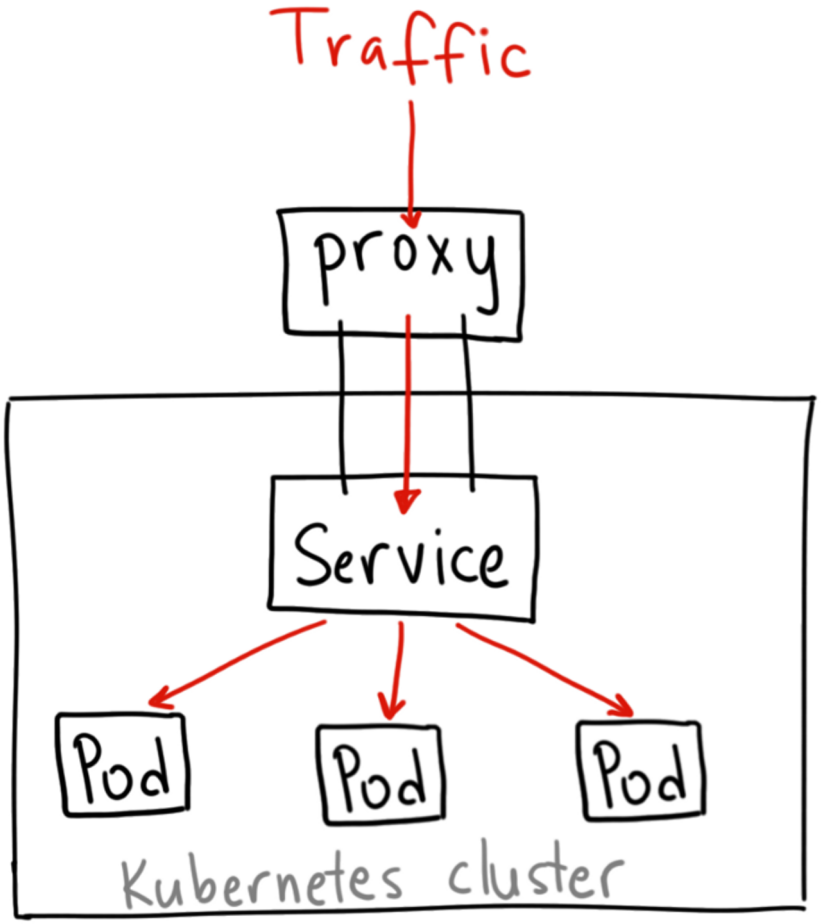
Kubernetes Services



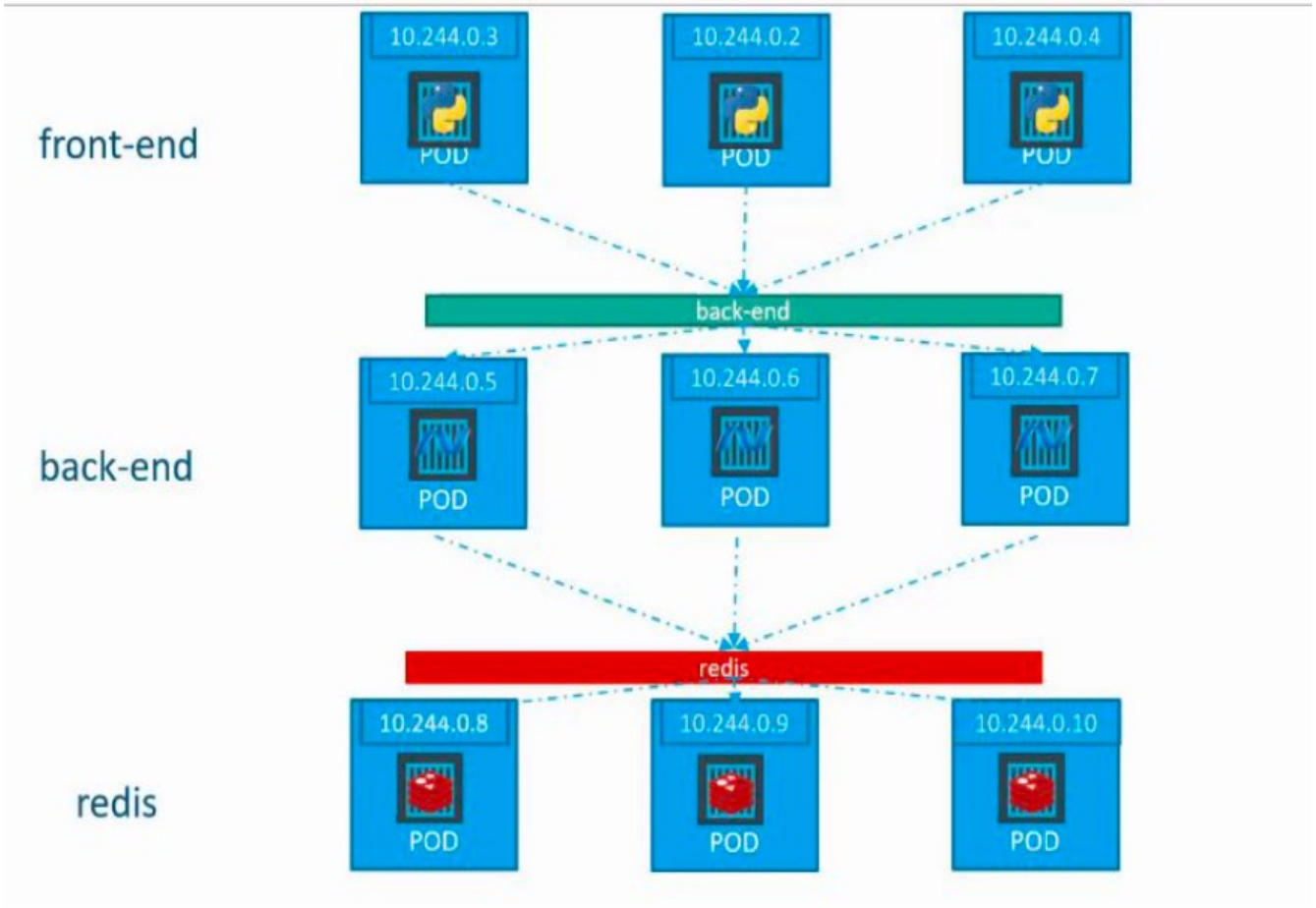
ClusterIP Service

- A ClusterIP service is the default Kubernetes service. It gives you a service inside your cluster that other apps inside your cluster can access. There is no external access.

```
apiVersion: v1
kind: Service
metadata:
 name: my-internal-service
spec:
 selector:
   app: my-app
 type: ClusterIP
 ports:
 - name: http
   port: 80
   targetPort: 80
   protocol: TCP
```



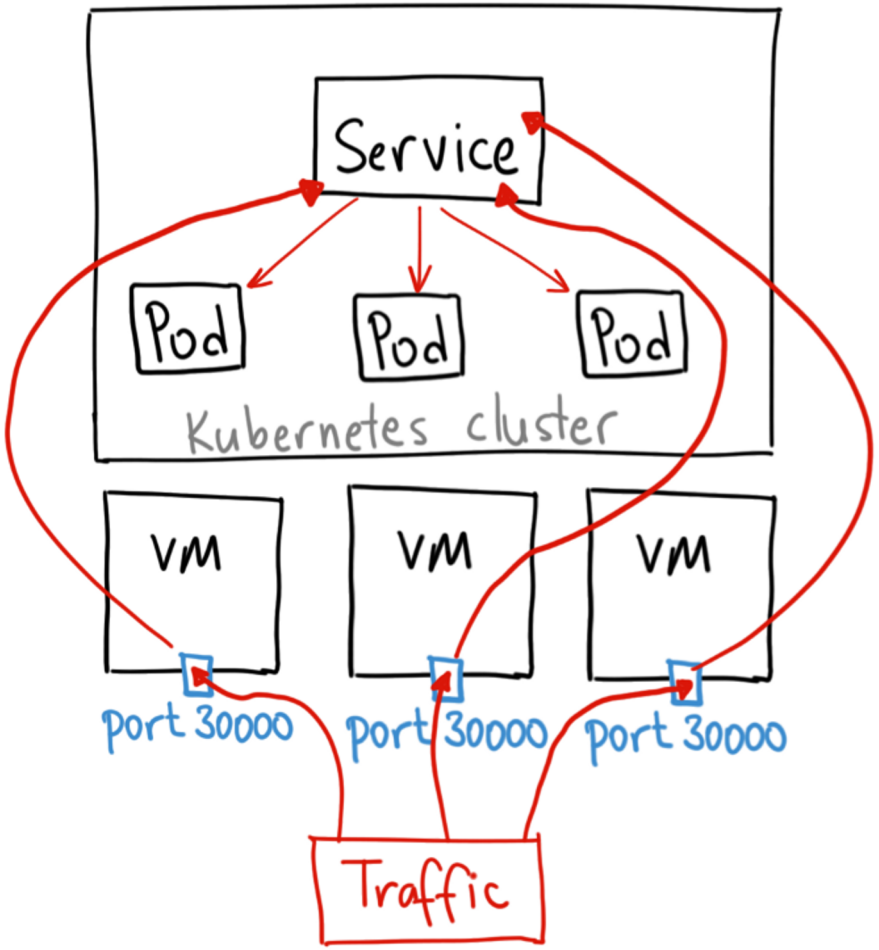
ClusterIP Service



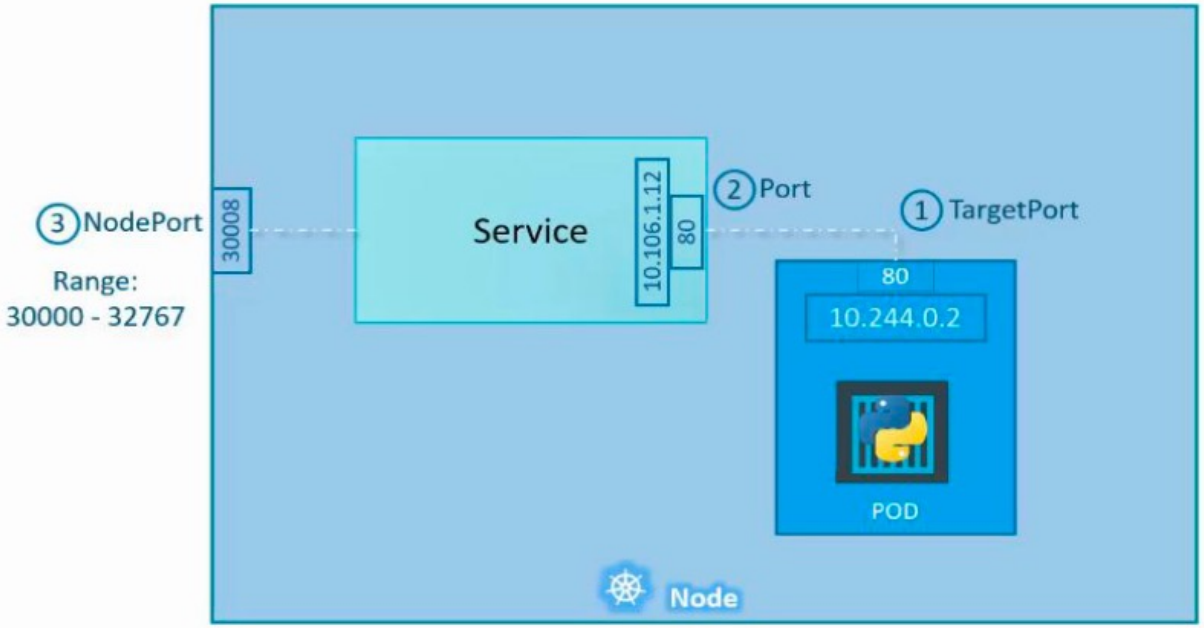
NodePort Service

- A NodePort service is the most primitive way to get external traffic directly to your service. NodePort, as the name implies, opens a specific port on all the Nodes (the VMs), and any traffic that is sent to this port is forwarded to the service.

```
apiVersion: v1
kind: Service
metadata:
 name: my-nodeport-service
spec:
 selector:
   app: my-app
 type: NodePort
 ports:
 - name: http
   port: 80
   targetPort: 80
   nodePort: 30036
   protocol: TCP
```



NodePort Service



service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
      *port: 80
       nodePort: 30008
```

NodePort Service

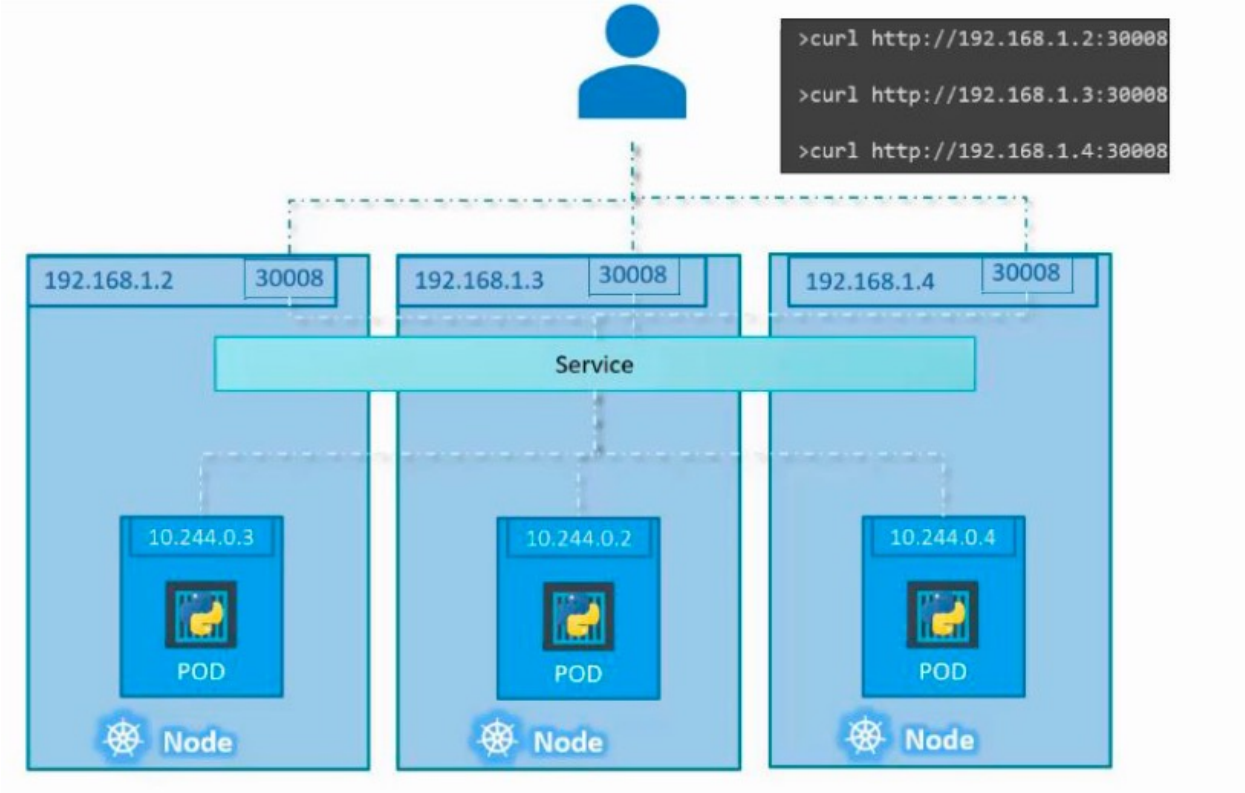
service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
    selector:
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

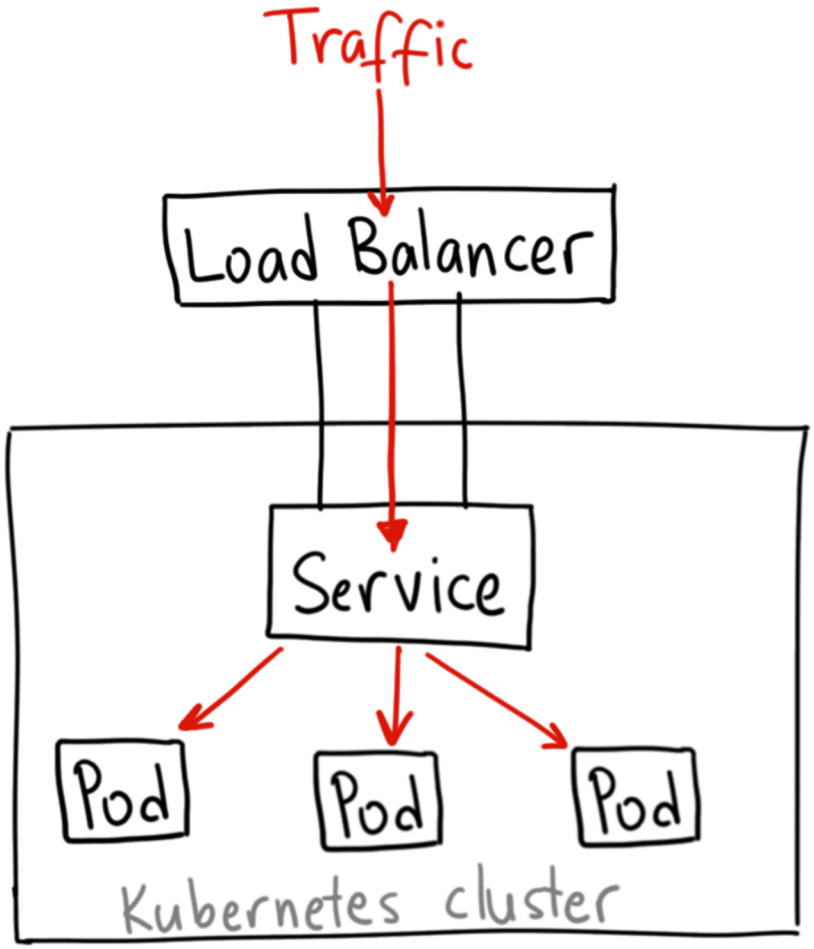
NodePort Service



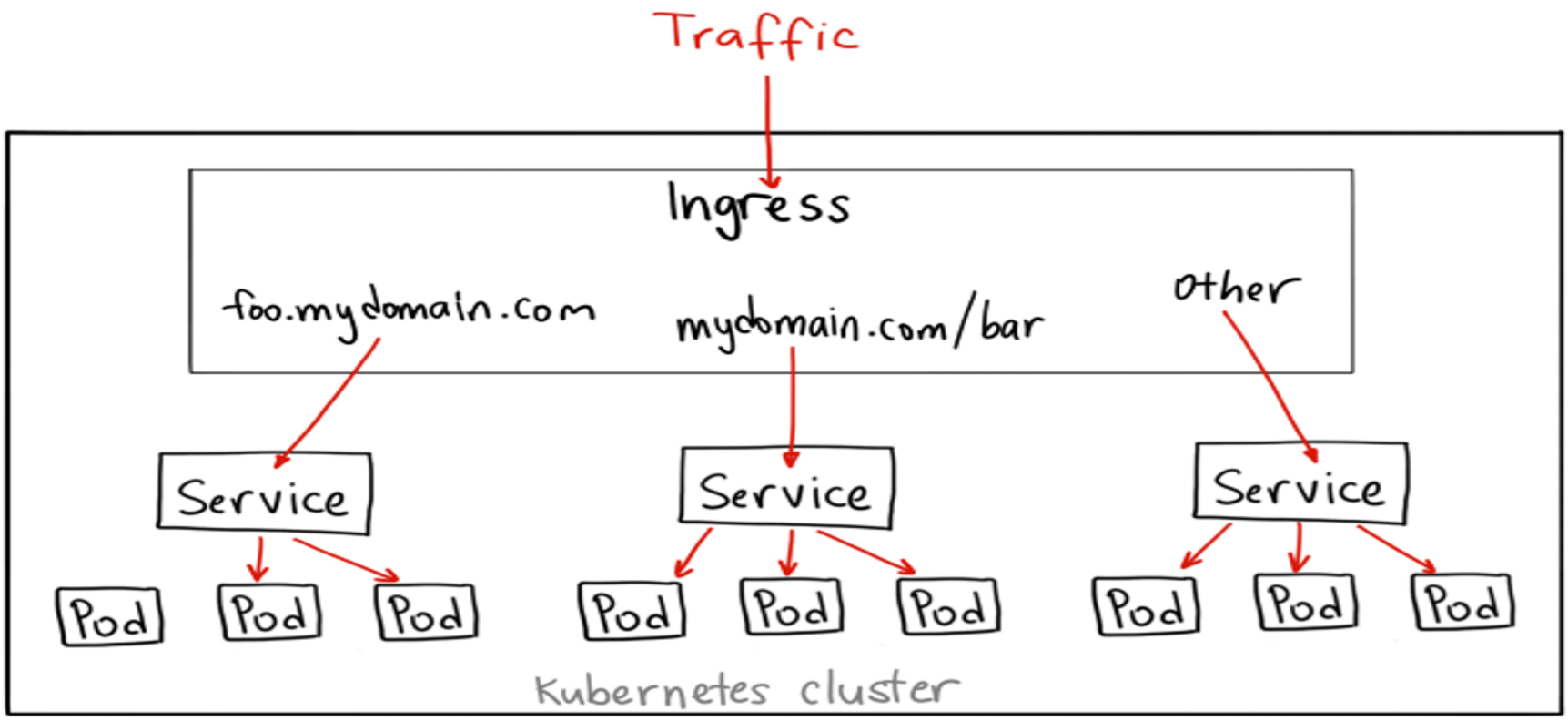
Load Balancer Service

A LoadBalancer service is the standard way to expose a service to the internet

The big downside is that each service you expose with a LoadBalancer will get its own IP address, and you have to pay for a LoadBalancer per exposed service, which can get expensive!



Ingress Service



Ingress Yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: my-ingress
spec:
 backend:
   serviceName: other
   servicePort: 8080
 rules:
 - host: foo.mydomain.com
   http:
     paths:
     - backend:
         serviceName: foo
         servicePort: 8080
 - host: mydomain.com
   http:
     paths:
     - path: /bar/*
       backend:
         serviceName: bar
         servicePort: 8080
```