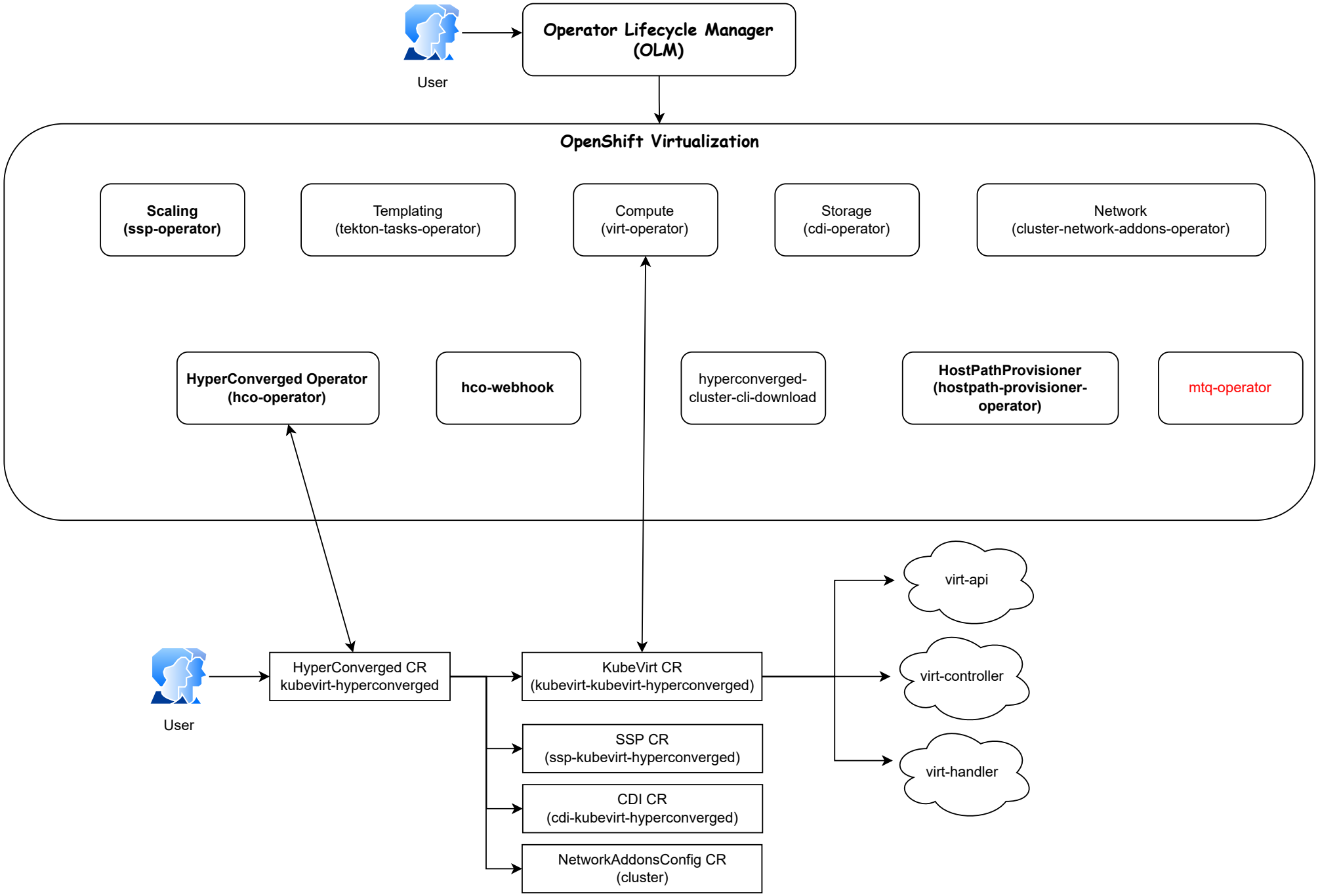
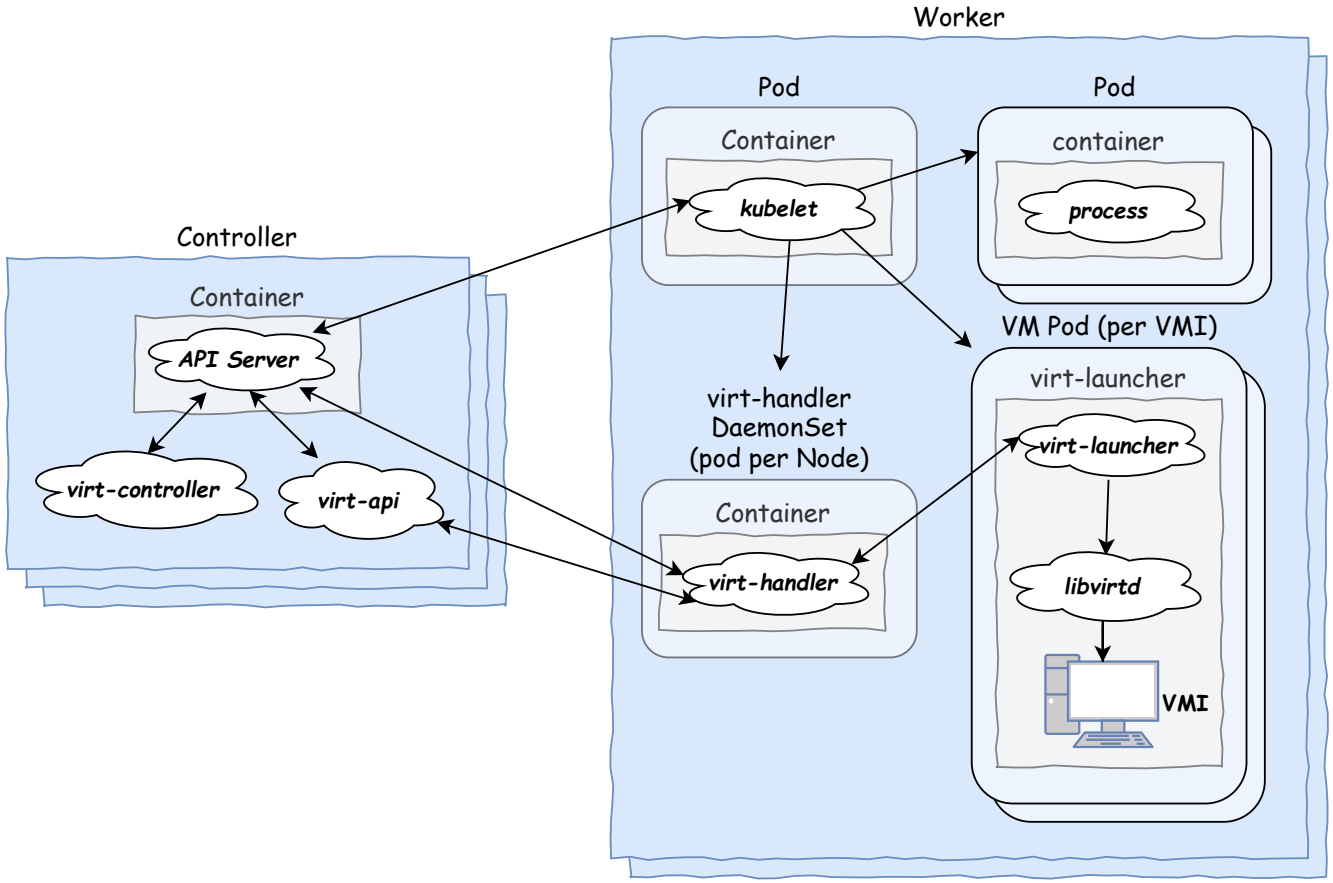
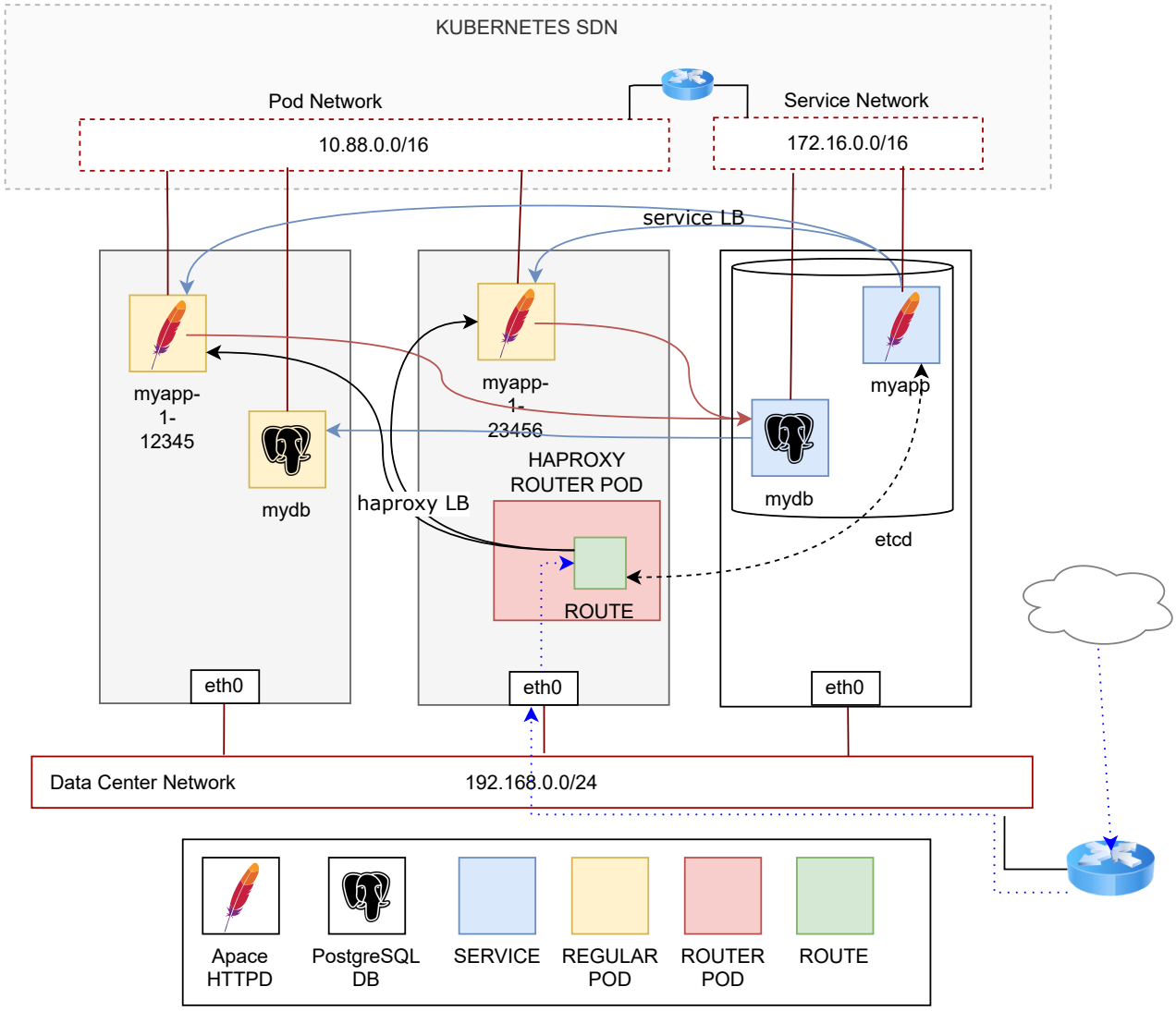
# OpenShift Virtualization



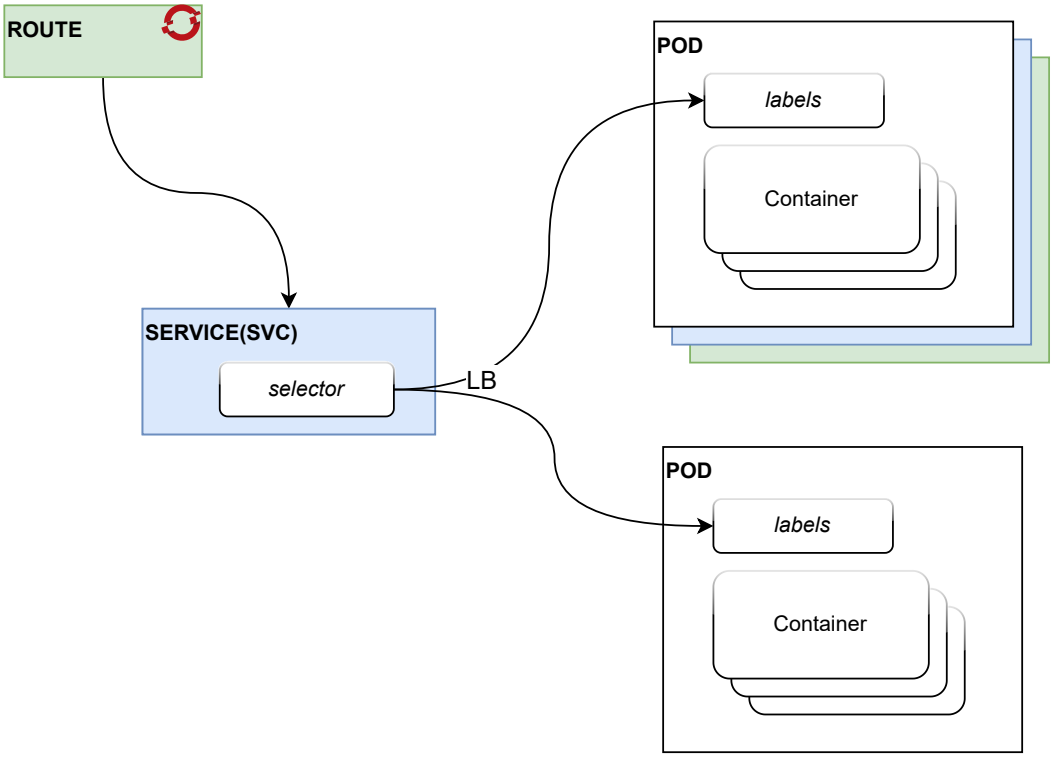

Worker
Pod
Container
kubelet
Pod
container
process
Controller
Container
API Server
virt-controller
virt-api
virt-handler
DaemonSet
(pod per Node)
Container
virt-handler
VM Pod (per VMI)
virt-launcher
virt-launcher
libvirtd
VMI

# Network


KUBERNETES SDN
Pod Network
10.88.0.0/16
Service Network
172.16.0.0/16
service LB
myapp-
1-
12345
mydb
myapp-
1-
23456
HAPROXY
ROUTER POD
haproxy LB
ROUTE
mydb
myapp
etcd
eth0
eth0
eth0
Data Center Network
192.168.0.0/24
Apace
HTTPD
PostgreSQL
DB
SERVICE
REGULAR
POD
ROUTER
POD
ROUTE

Service LB Pods using
Linux Netfilter feature

# Route, Service and Pod Relationship




**POD**
A pod contains one or more containers.

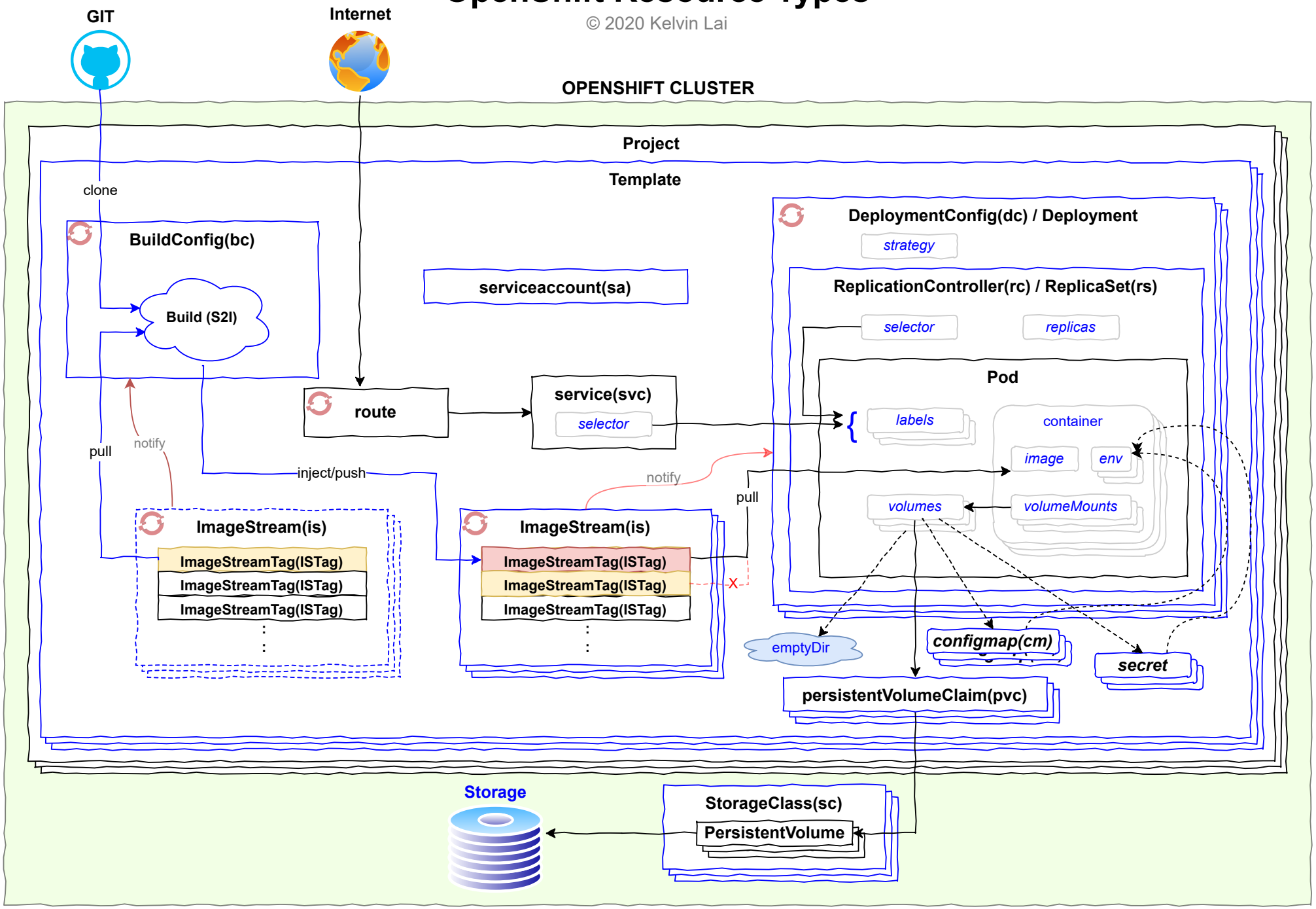
**SERVICE**
A service references the pod(s) by using the label selector.
The service load balances the connections between all the pods.

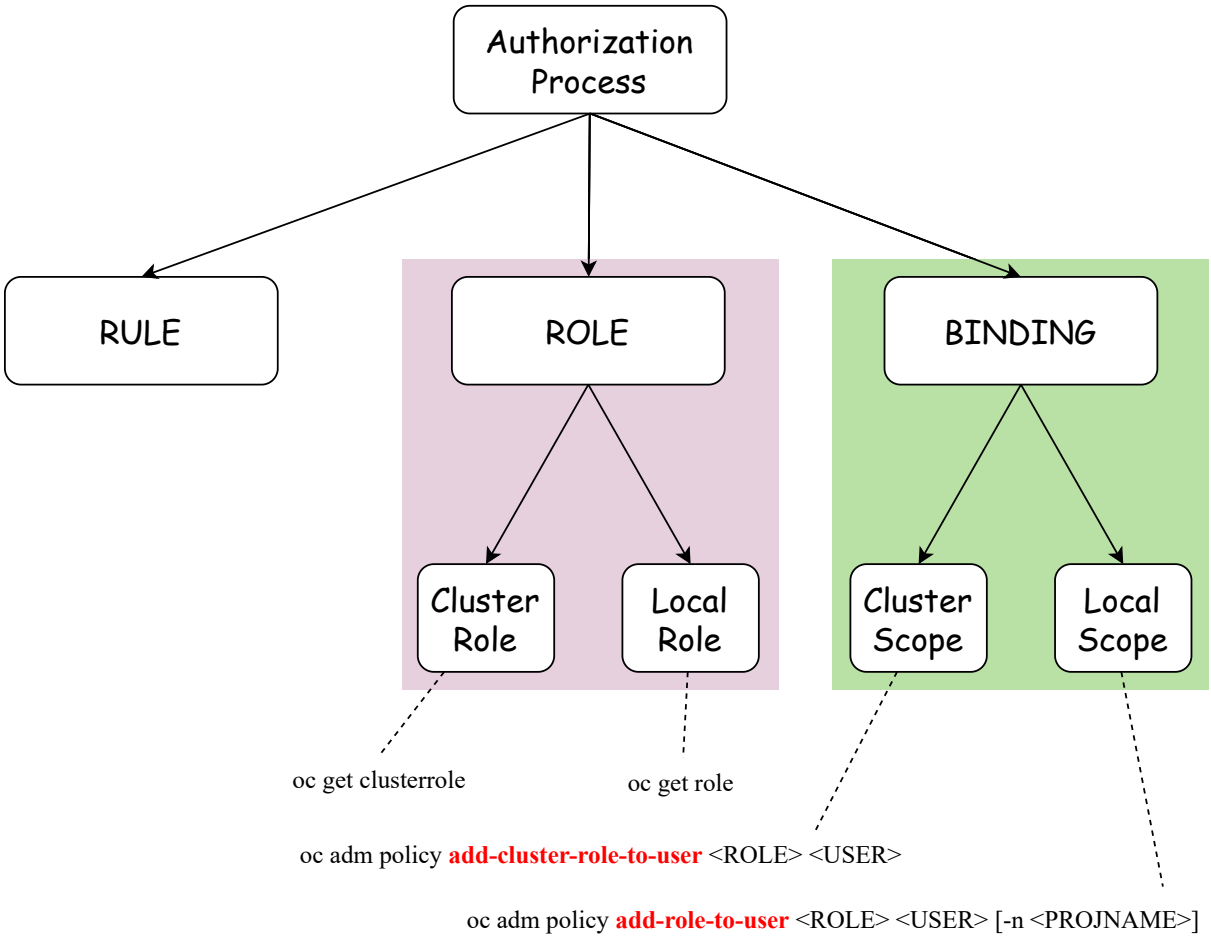
**ROUTE**
A route exposes the service to the external world.

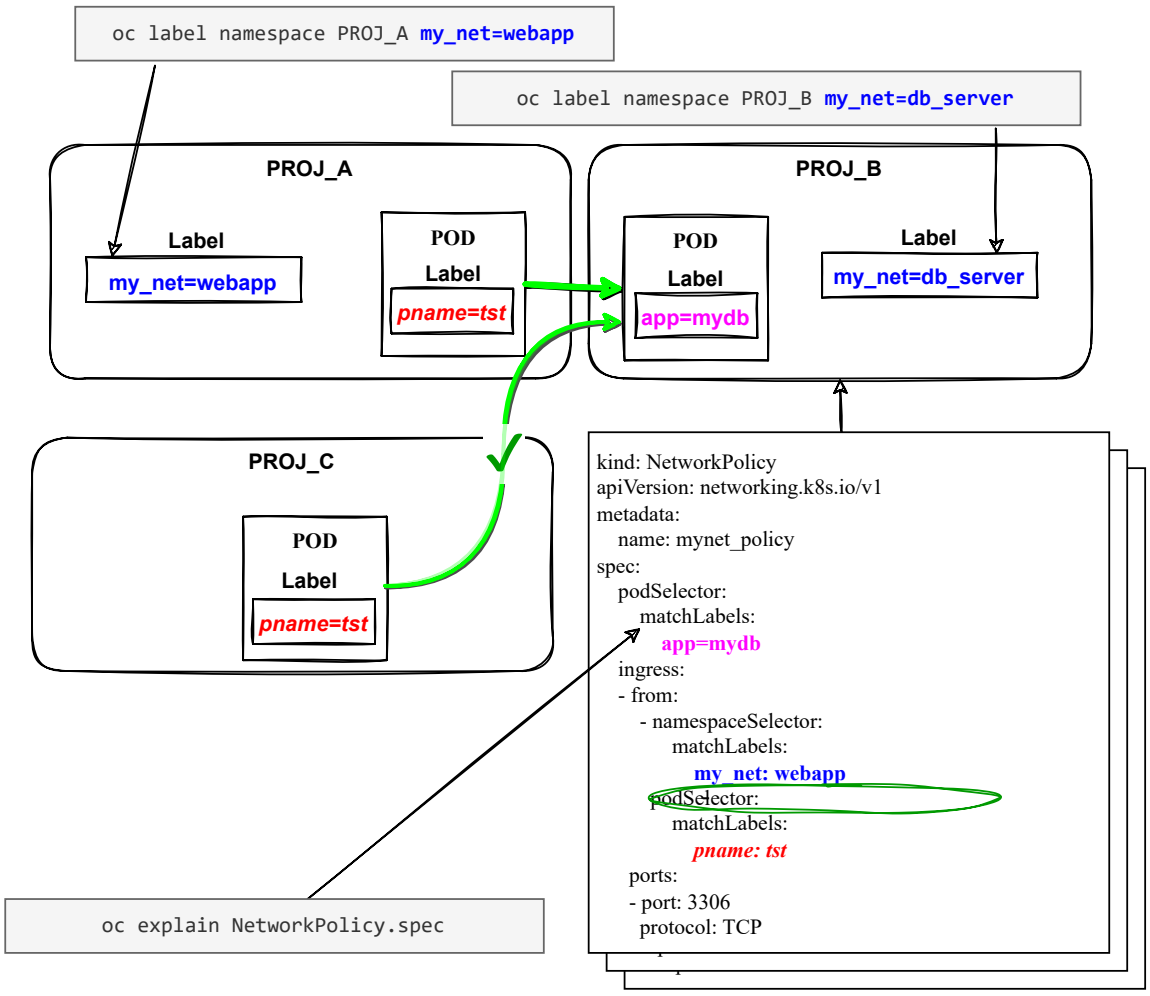
Warning: A service "can" refer to different pods, if the pods have the same label.

# OpenShift Resource Types

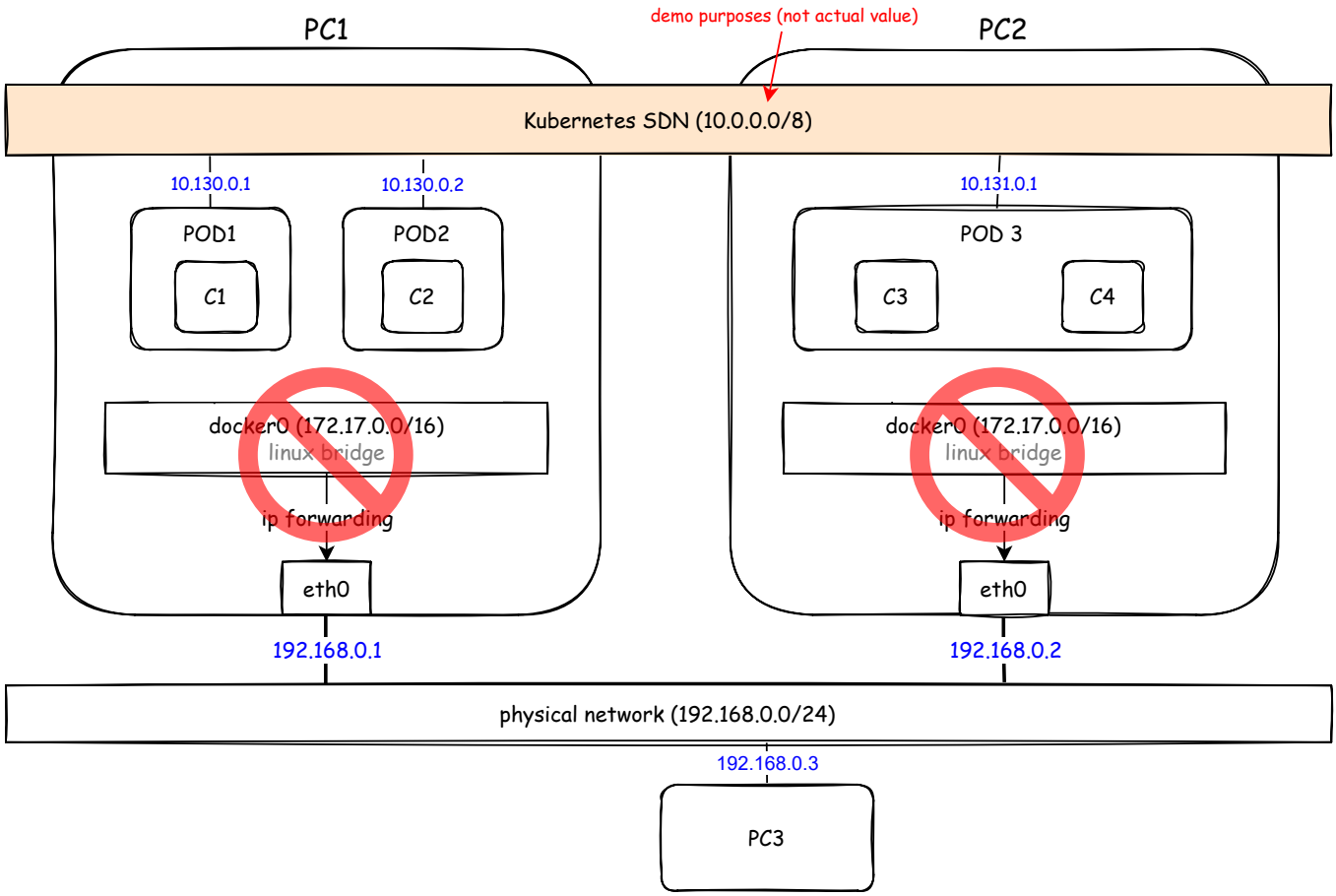


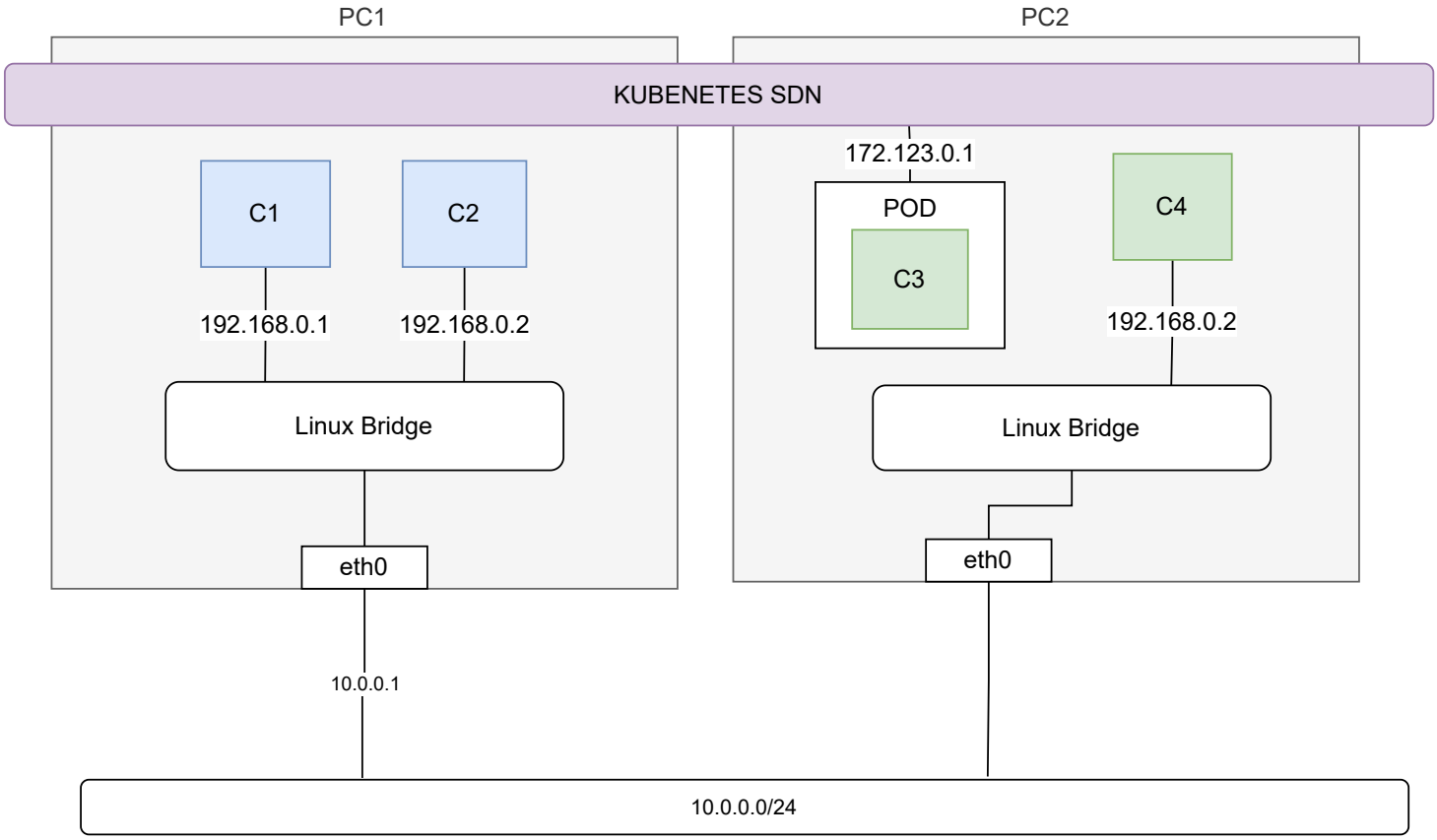



Authorization Process
RULE
ROLE
BINDING
Cluster Role
Local Role
Cluster Scope
Local Scope
oc get clusterrole
oc get role
oc adm policy add-cluster-role-to-user <ROLE> <USER>
oc adm policy add-role-to-user <ROLE> <USER> [-n <PROJNAME>]


oc label namespace PROJ_A my_net=webapp
oc label namespace PROJ_B my_net=db_server
PROJ_A
Label
my_net=webapp
POD
Label
pname=tst
PROJ_B
POD
Label
app=mydb
Label
my_net=db_server
PROJ_C
POD
Label
pname=tst
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
name: mynet_policy
spec:
podSelector:
matchLabels:
app=mydb
ingress:
- from:
- namespaceSelector:
matchLabels:
my_net: webapp
podSelector:
matchLabels:
pname: tst
ports:
- port: 3306
protocol: TCP
oc explain NetworkPolicy.spec

# Basic Network - Container vs Kubernetes



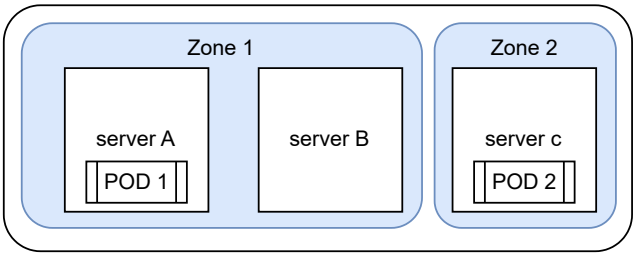
# POD Scheduling

1. Get a list of all NODES

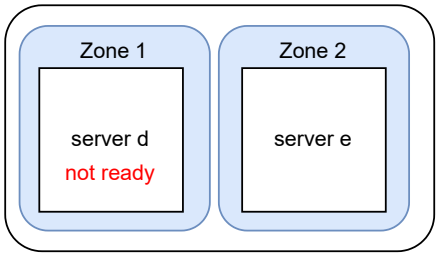
2. Go through all the predicates for FILTERing. If NODE fails predicate rule, remove from list. Region affinity.

3. With remainder list of NODES, prioritize them using the weightage rules. NO filtering of NODES done here. Zone anti-affinity.

4. Select the NODE with highest points.




Singapore




Malaysia

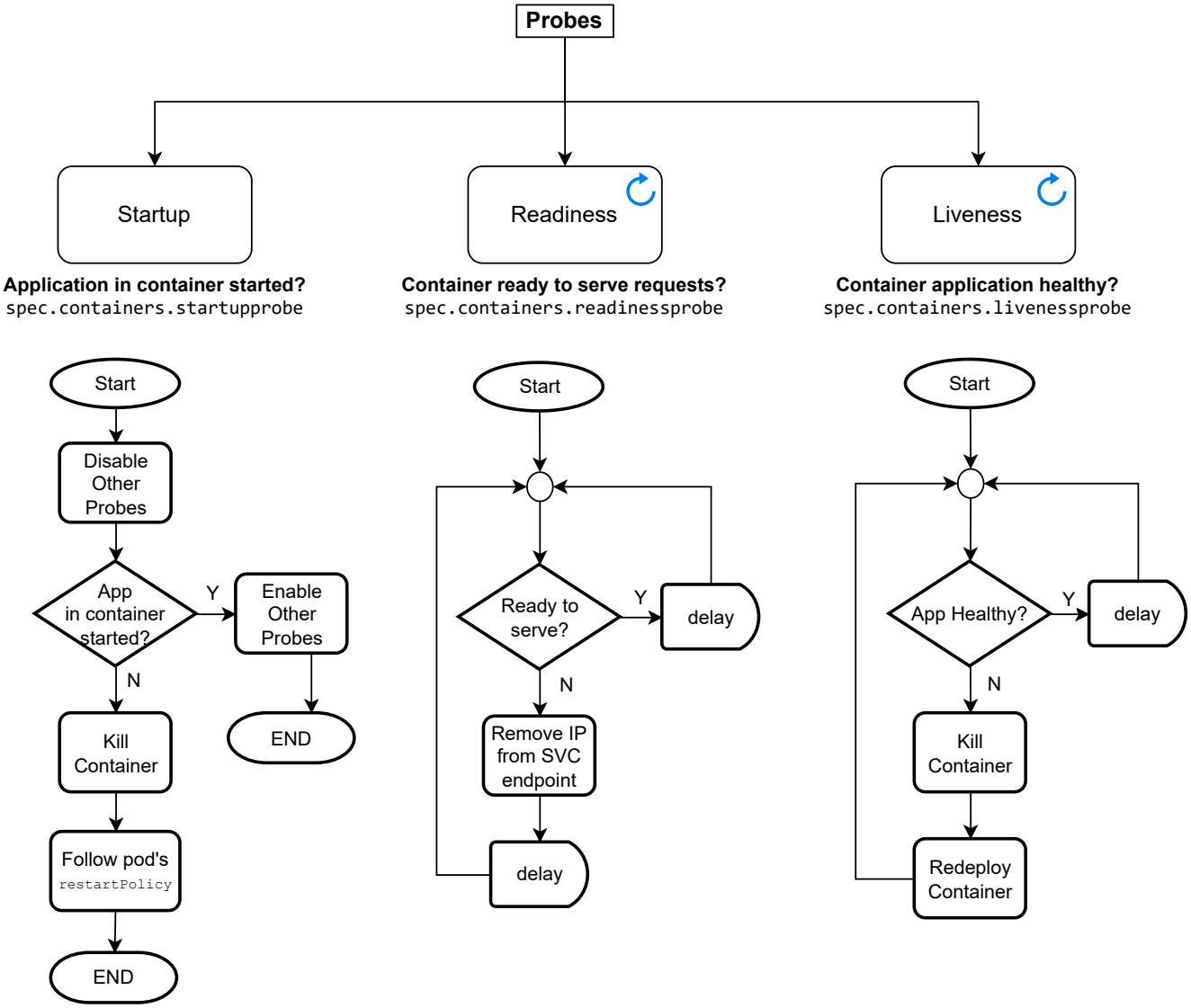
server a = 0 + 0
server b = 1 + 0
server c = 1 + 1
~~server d~~
~~server e~~

`oc label node <NODE> <KEY>=<VALUE>`

Region
<KEY> = failure-domain.beta.kubernetes.io/region
A set of hosts in closed geographical area. High speed connectivity.

Zone (availability zone)
<KEY> = failure-domain.beta.kubernetes.io/zone
A set of hosts that share common critical infra components (ups, switch, storage)

# Health Monitoring




successThreshold = 1
failureThreshold = 3