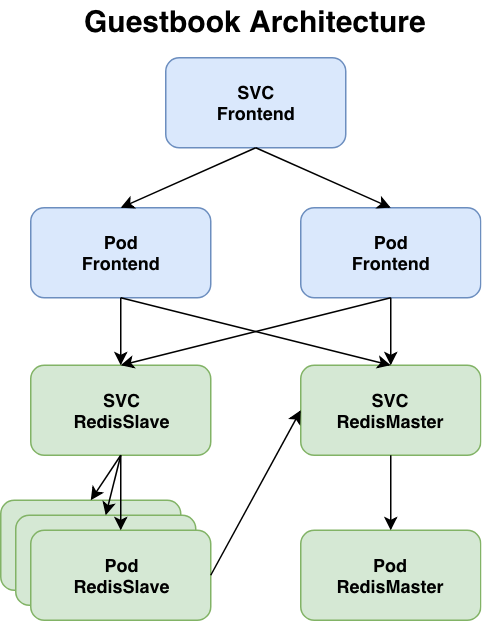
# Service



wget<https://raw.githubusercontent.com/sparkmbt/sparkmbt/main/kubesample.yaml>

k explain svc

k delete -f kubesample.yaml

k apply -f kubesample.yaml

Redis-Master

k get deploy redis-master  
k get pods -l role=master -o wide

k get svc redis

k describe svc redis

Redis-slave

k get deploy redis-slave

k get svc redis-slave

k get pods -l role=slave -o wide

k get ep redis-slave

k describe svc redis-slave

Frontend - Node-port

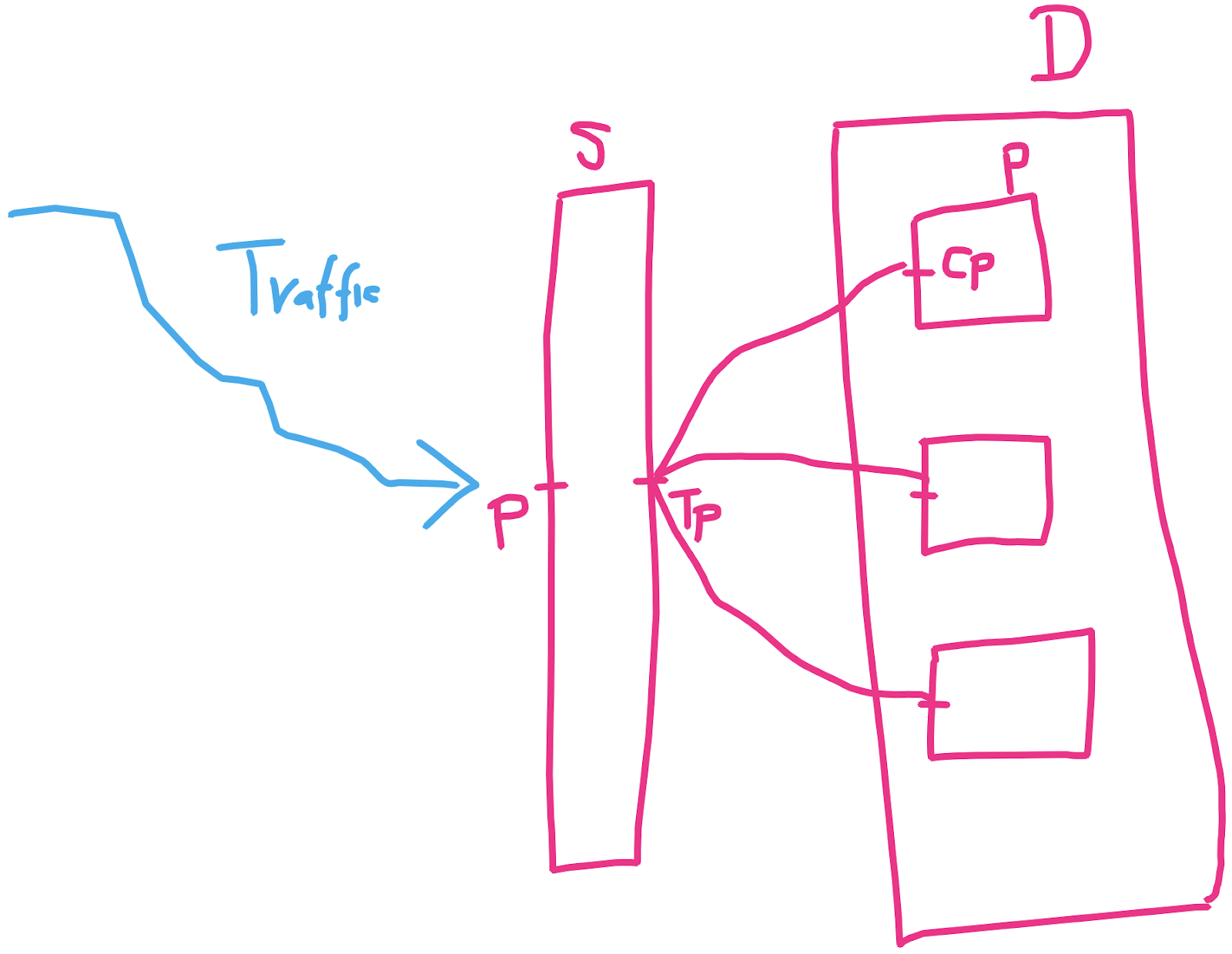
k get deploy frontend

k get pods -l app=kubesample -o wide

k get svc frontend

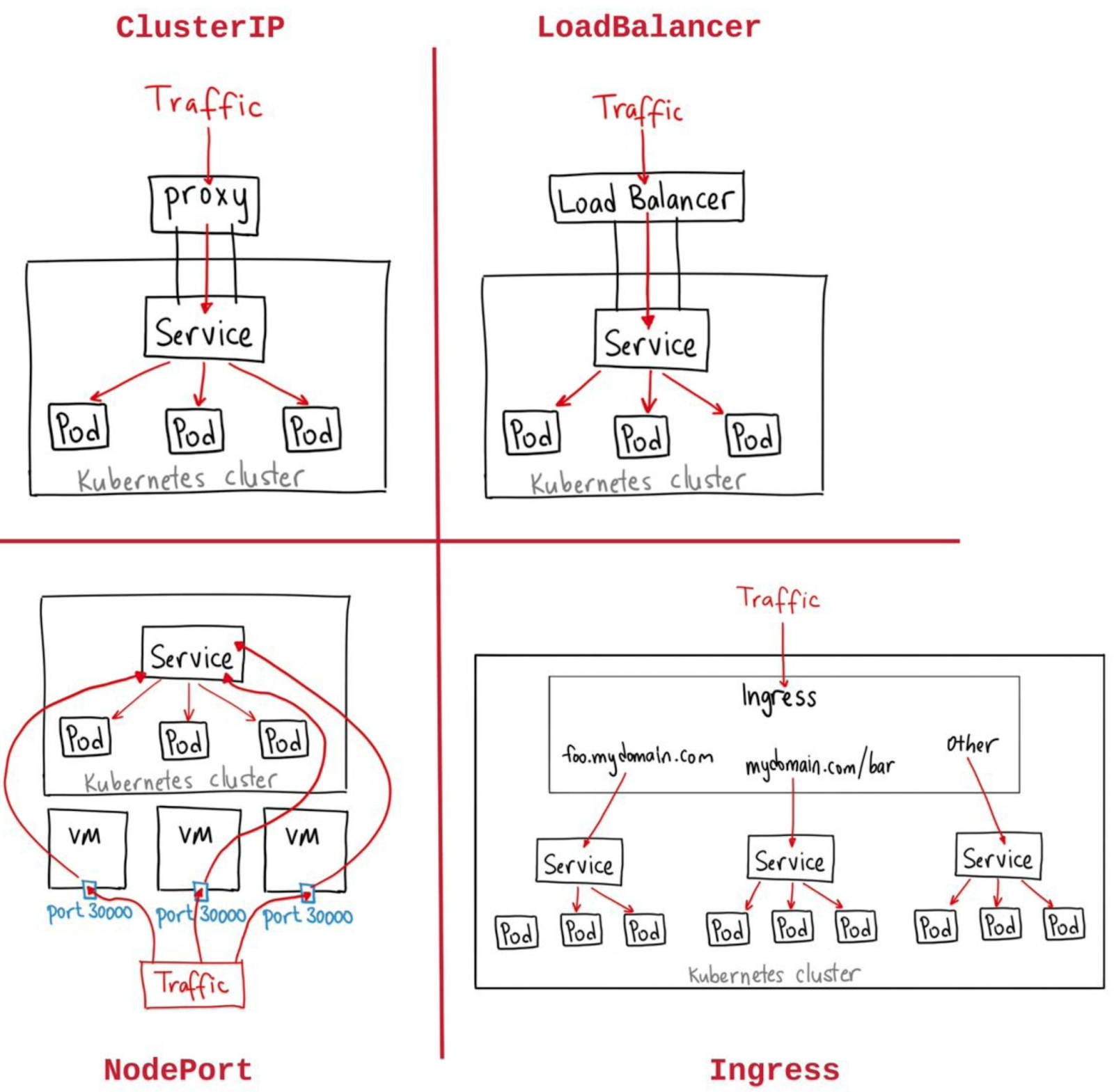
k get ep frontend

k describe svc frontend



# Service Types

* ClusterIP(default): Exposes the Service on a cluster-internal IP. Choosing this value makes the Service only reachable from within the cluster. This is the default ServiceType.
* NodePort: Exposes the Service on each Node's IP at a static port (the NodePort). A ClusterIP Service, to which the NodePort Service routes, is automatically created. You'll be able to contact the NodePort Service, from outside the cluster, by requesting <NodeIP>:<NodePort>.
* LoadBalancer: Exposes the Service externally using a cloud provider's load balancer. NodePort and ClusterIP Services, to which the external load balancer routes, are automatically created



# DNS in the Cluster

By default, Kubernetes has core DNS,

k get deploy -n kube-system

NAME READY UP-TO-DATE AVAILABLE AGE

**coredns** 2/2 2 2 13d

k get svc -n kube-system

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

**kube-dns** ClusterIP **10.96.0.10** <none> 53/UDP,53/TCP,9153/TCP 13d

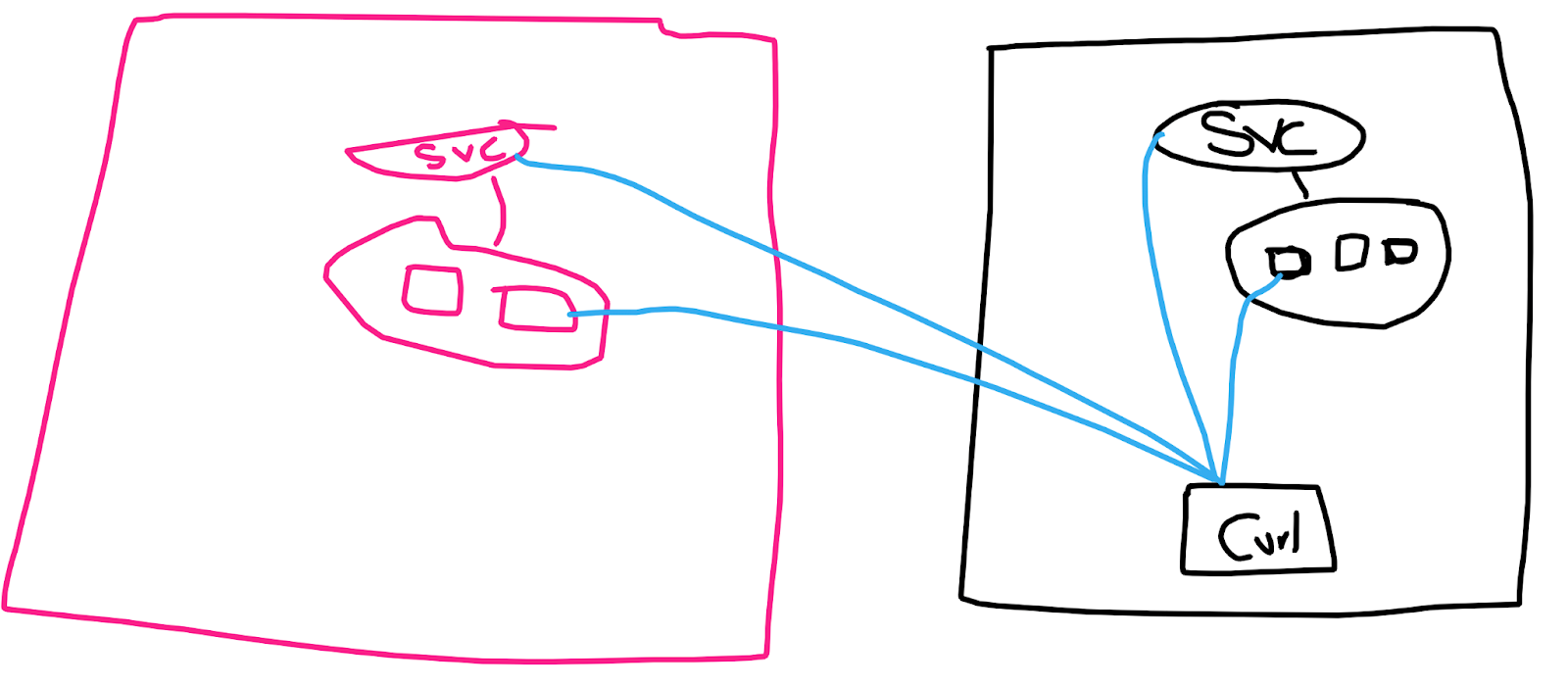
# DNS & Network communication

Pod -> Service in Same namespace

Pod -> Service in different namespace

Pod -> Pod in same namespace

Pod -> Pod in different namespace



#test ns - Pod and Service deployment

k create ns test

vi nginx.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-nginx

spec:

selector:

matchLabels:

run: my-nginx

replicas: 2

template:

metadata:

labels:

run: my-nginx

spec:

containers:

- name: my-nginx

image: nginx

ports:

- containerPort: 80

k apply -f nginx.yaml -n test

vi nginx-service.yaml

apiVersion: v1

kind: Service

metadata:

name: my-nginx

namespace: test

spec:

type: NodePort

ports:

- port: 80

selector:

run: my-nginx

k apply -f nginx-service.yaml

Alternate way to create a service,  
k expose deploy my-nginx --port 80 --type NodePort -n test

#IP and Networking

k get pods -o wide | grep frontend

#Check the List of Pod IPs

k get ep frontend

#Check service IP

k get svc frontend

#lets use NetworkingImage - radial/busyboxplus:curl

kubectl run curl --image=radial/busyboxplus:curl -i --tty

[ root@curl:/ ]$ nslookup google.com

[ root@curl:/ ]$ nslookup frontend  
Server: 10.96.0.10

Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: frontend

Address 1: 10.100.244.95 frontend.default.svc.cluster.local

[ root@curl:/ ]$ curl frontend

[ root@curl:/ ]$ curl frontend.default.svc.cluster.local

#Diff ns - test

[ root@curl:/ ]$ nslookup my-nginx.test

[ root@curl:/ ]$ curl my-nginx.test.svc.cluster.local

# Accessing Pod using Pod IP

DONT COPY PASTE COMMANDs, Find Pod IP and replace them here

[ root@curl:/ ]$ curl 192.168.232.217:80

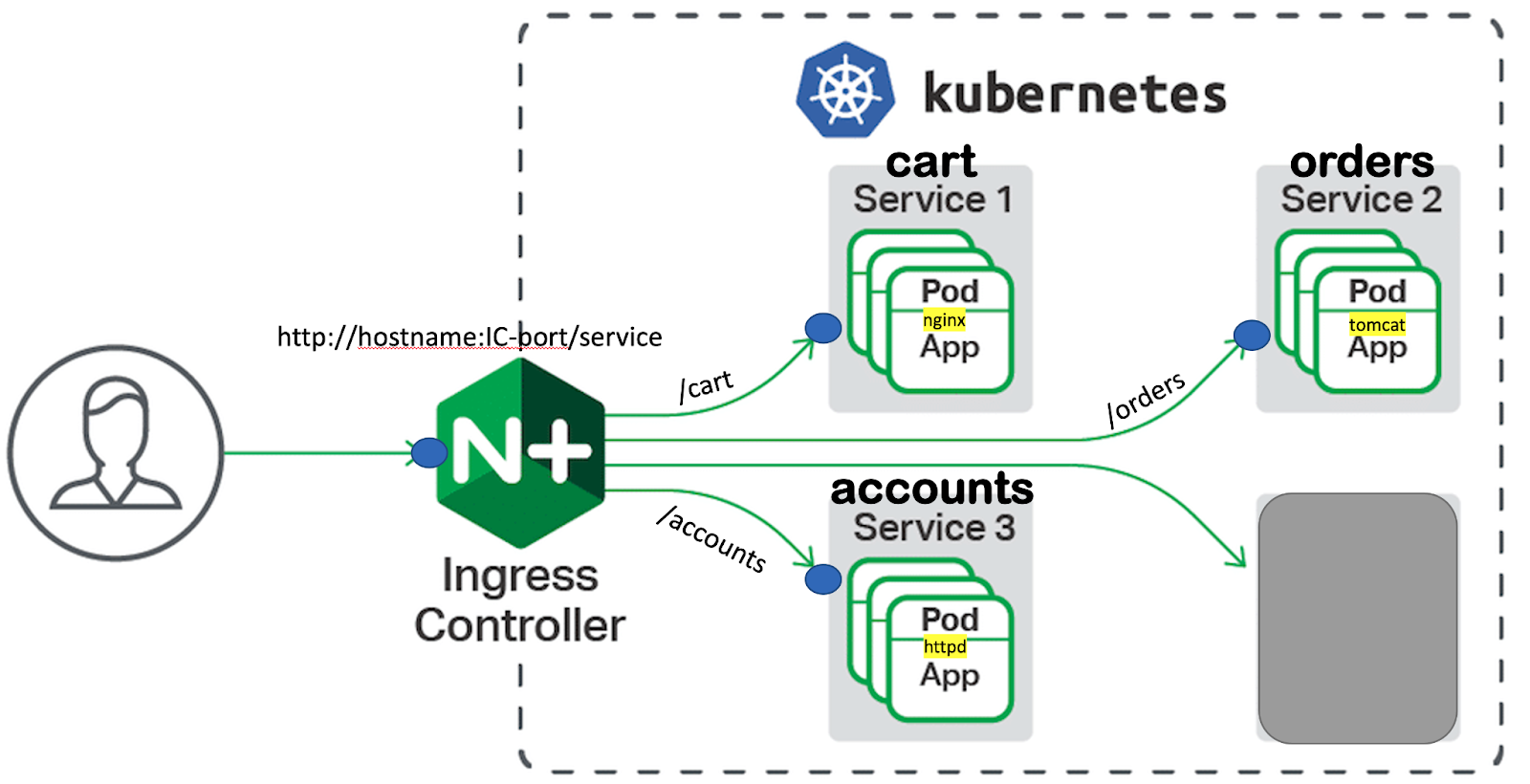
[ root@curl:/ ]$ exit

k delete -f kubesample.yaml

# Ingress

Kubernetes Ingress controllers:

* Accept traffic from outside the Kubernetes platform, and load balance it to pods (containers) running inside the platform
* Are configured using the Kubernetes API to deploy objects called “Ingress Resources”
* Monitor the pods running in Kubernetes and automatically update the load‑balancing rules when pods are added or removed from a service
* **Content-based routing**:
  + Routing requests with the URI that starts with /serviceA to service A and requests with the URI that starts with /serviceB to service B.



kubectl apply -f<https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/baremetal/deploy.yaml>

k get all -n ingress-nginx

Get nginx controller endpoint

k get svc -n ingress-nginx ingress-nginx-controller

# Note down the Port number 80:**32184**/TCP,443:32649/TCP

ingress-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

app: cart

name: cart

spec:

replicas: 1

selector:

matchLabels:

app: ing1

template:

metadata:

labels:

app: ing1

spec:

containers:

- image: nginx

name: nginx

ports:

- containerPort: 80

---

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

app: orders

name: orders

spec:

replicas: 1

selector:

matchLabels:

app: ing3

template:

metadata:

creationTimestamp: null

labels:

app: ing3

spec:

containers:

- env:

- name: TOMCAT\_PASSWORD

value: rootroot

image: bitnami/tomcat

name: tomcat

ports:

- containerPort: 8080

---

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

app: accounts

name: accounts

spec:

replicas: 1

selector:

matchLabels:

app: ing2

template:

metadata:

labels:

app: ing2

spec:

containers:

- image: httpd

name: httpd

ports:

- containerPort: 80

---

k apply -f ingress-deployment.yaml

kubectl expose deployment cart --name cart --port 80

kubectl expose deployment orders --name orders --port 80 --target-port 8080

kubectl expose deployment accounts --name accounts --port 80

Ingress.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-demo

annotations:

kubernetes.io/ingress.class: "nginx"

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- http:

paths:

- path: /cart

pathType: Prefix

backend:

service:

name: cart

port:

number: 80

- path: /account

pathType: Prefix

backend:

service:

name: accounts

port:

number: 80

- path: /orders

pathType: Prefix

backend:

service:

name: orders

port:

number: 80

k apply -f ingress.yaml

k describe ingress ingress-demo

How to get IngressPort?

kubectl get svc -n ingress-nginx ingress-nginx-controller

# Note down the Port number 80:**32184**/TCP,443:32649/TCP

curl http://localhost:**<node-port>**/cart

curl http://localhost:**<node-port>**/orders

curl http://localhost:**<node-port>**/accounts

For Reference:

Service yaml files for reference

apiVersion: v1

kind: Service

metadata:

name: orders

spec:

ports:

- port: 80

**targetPort: 8080**

selector:

app: ing3

type: ClusterIP

# Load Balancer

apiVersion: v1

kind: Service

metadata:

name: frontend

labels:

app: kubesample

tier: frontend

spec:

# comment or delete the following line if you want to use a LoadBalancer

#type: NodePort

# if your cluster supports it, uncomment the following to automatically create

# an external load-balanced IP for the frontend service.

**type: LoadBalancer**

ports:

- port: 80

selector:

app: kubesample

tier: frontend