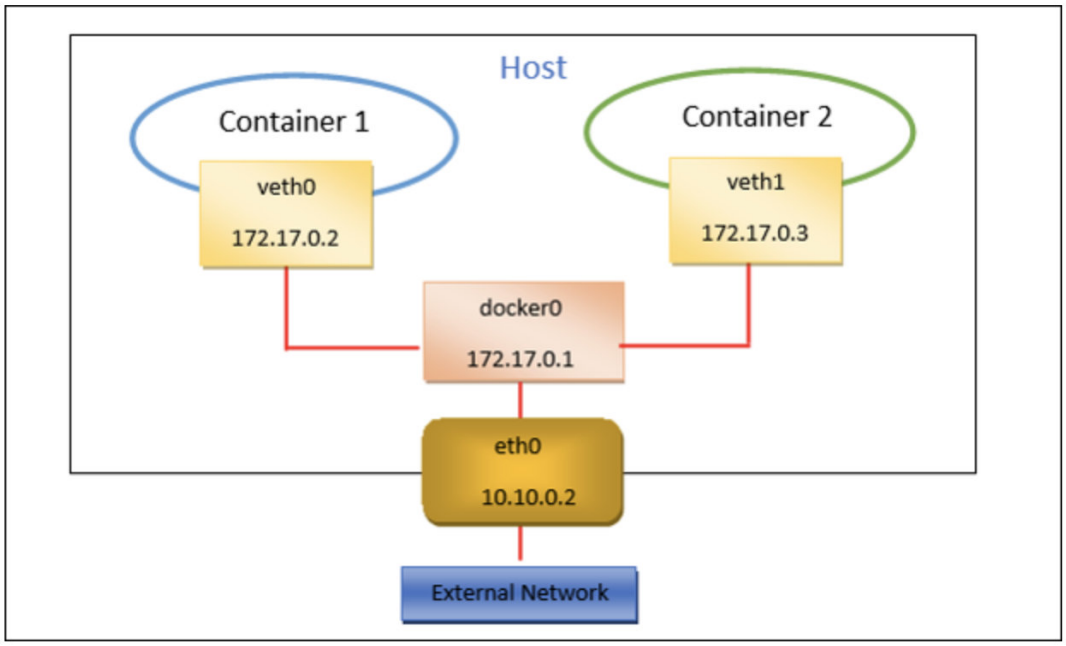
K8S Network

K8S network is divided into 3 levels. From inside to outside, there are 3 types: 1) Container Network. 2) Intra-Cluster Network. 3) External Cluster Network. Below is the details,

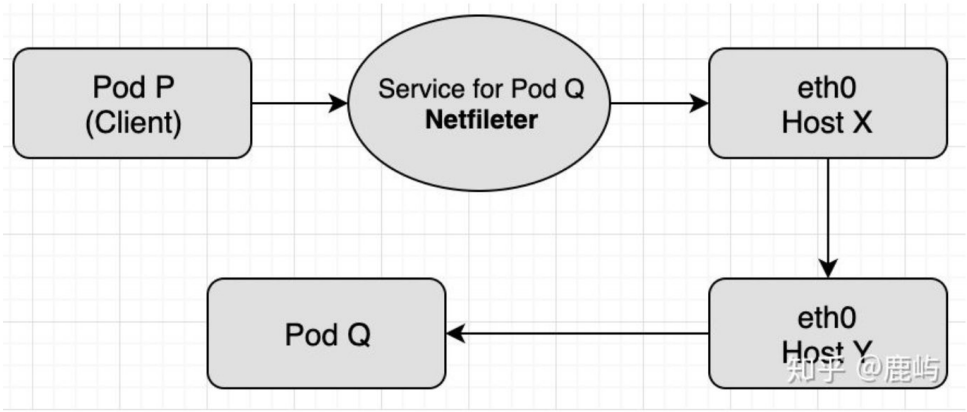
1. Container Network

Docker networking is limited within the host itself. By default, it creates a virtual bridge named docker0 and for each container it allocates a virtual ethane which get attached to the bridge docker0. So, in Docker two containers can communicate with each other if they resident on the same host.

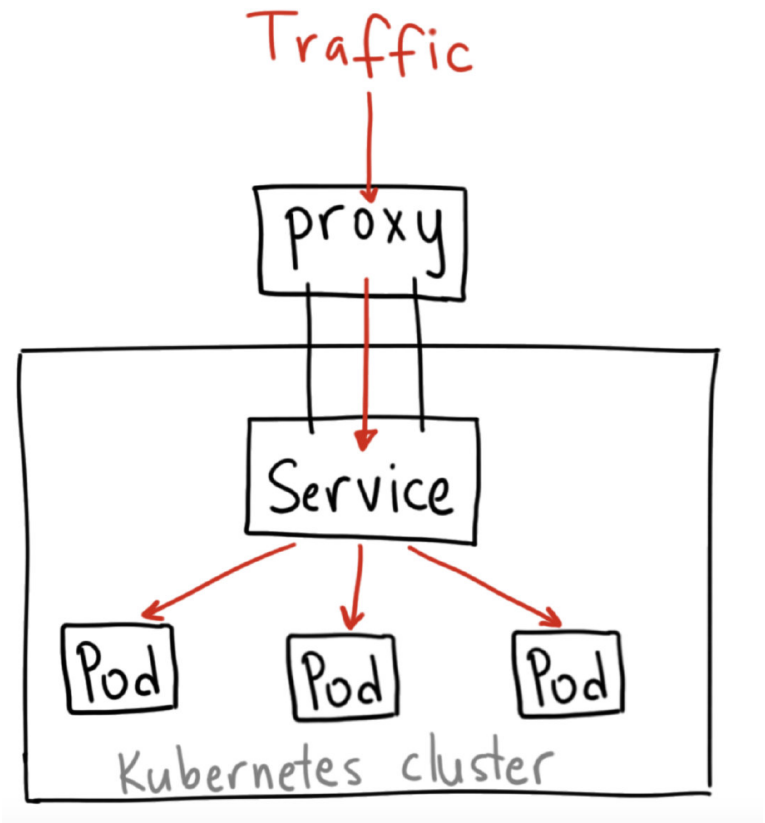


2. Intra-Cluster Network

This means Pod to Pod communicate. In the cluster, all Pods share the same network segment. So the Pods can communicate directly.



Also you can expose the Pod to a service, then in the cluster, other applications can access the Pod service. Like this,

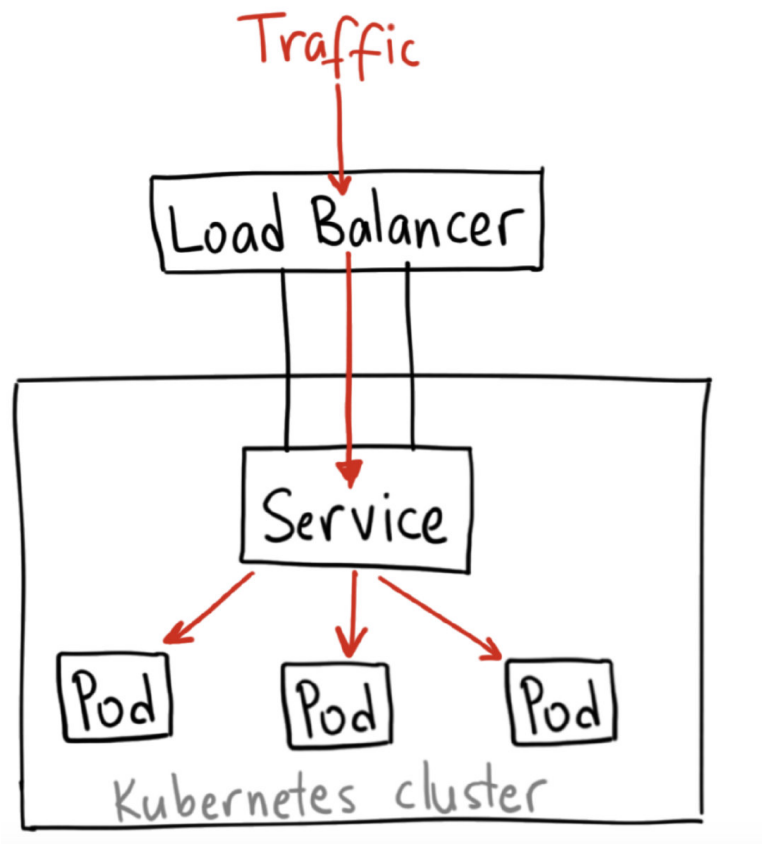


3. External Cluster Network

How to access the k8s cluster network from outside? There are 3 ways to build the network model.

3.1 LoadBalacer

A LoadBalacer Service is the standard way to expose a service to the internet. But only on the AWS or Google Cloud can use this function. On GKE, this will spin up a network load balancer that will give you a single IP address that will forward all traffic to your service.

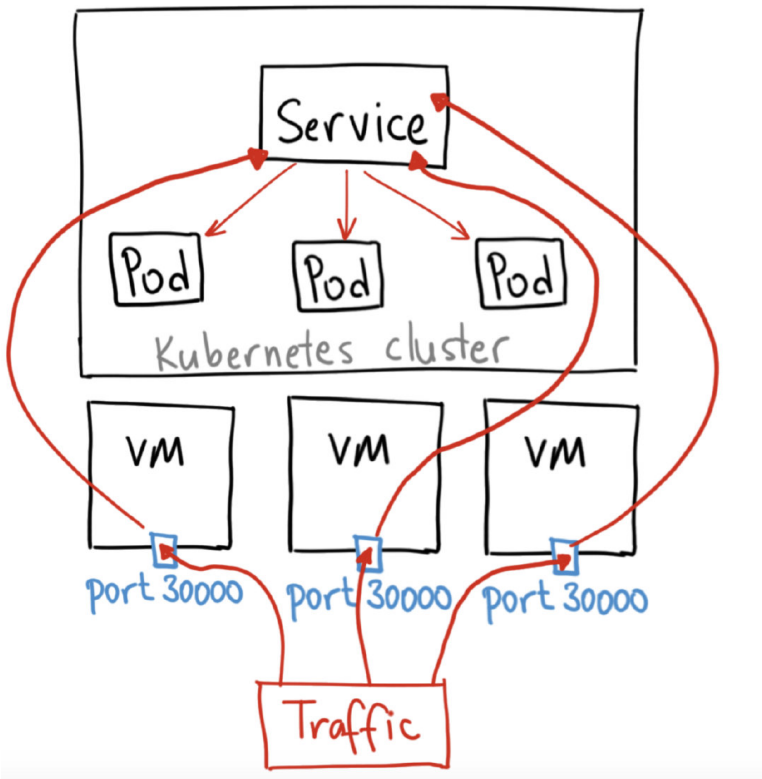


If you want to want to directly expose a service, this is the default method. All traffic on the port you specify will be forwarded to the service.There is no filtering, no routing, etc. This means you can send almost any kind of traffic to it, like HTTP, TCP, UDP. Websocket, gRPC, or whatever.

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.

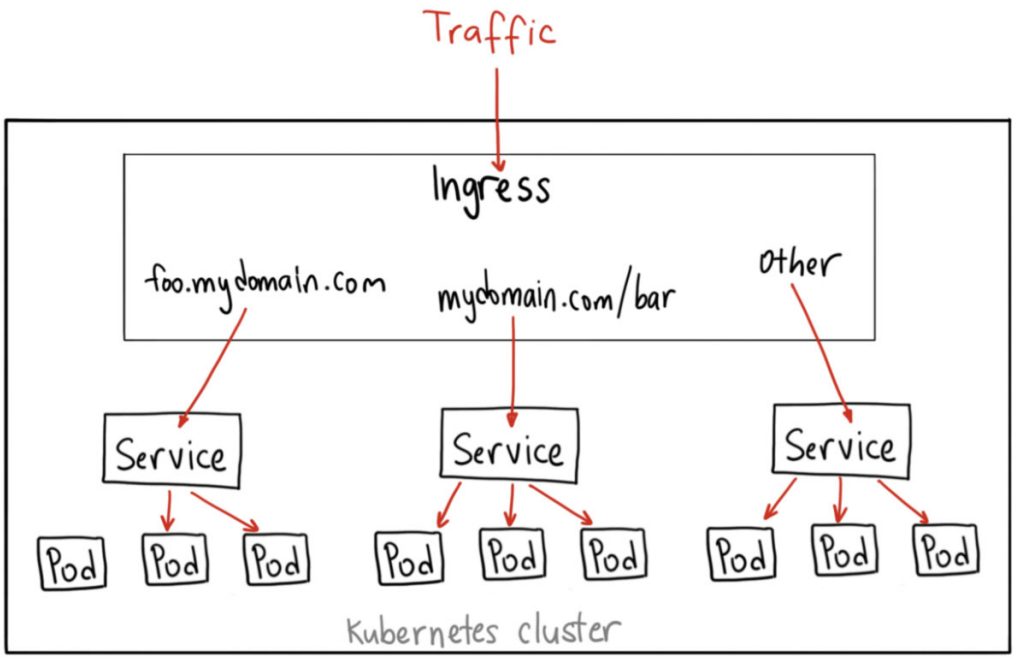
3.2 NodePort

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 Node, and any traffic that is sent to this port is forwarded to the service.



3.3 Ingress

Unlike all the above examples, Ingress is actually not a type of service. Instead, it sits in front of multiple services and act as a "smart router" or entry point into your cluster.You can do a lot of different things with an Ingress, and there are many types of Ingress controllers that have different capabilities.




```
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
```