Container Networking

Petr Horáček

What is this about?

What is this about?

- Cluster networking **evolution**

What is this about?

- Cluster networking **evolution**
- Container networking **guts**

What is this about?

- Cluster networking **evolution**
- Container networking **guts**
- **Wonderful** world of Kubernetes **networking**

What is this about?

- Cluster networking **evolution**
- Container networking **guts**
- **Wonderful** world of Kubernetes **networking**
- **Using** Kubernetes networking

Why should I care?

- Overview why is it different, how it works and how to use it

Networking Evolution

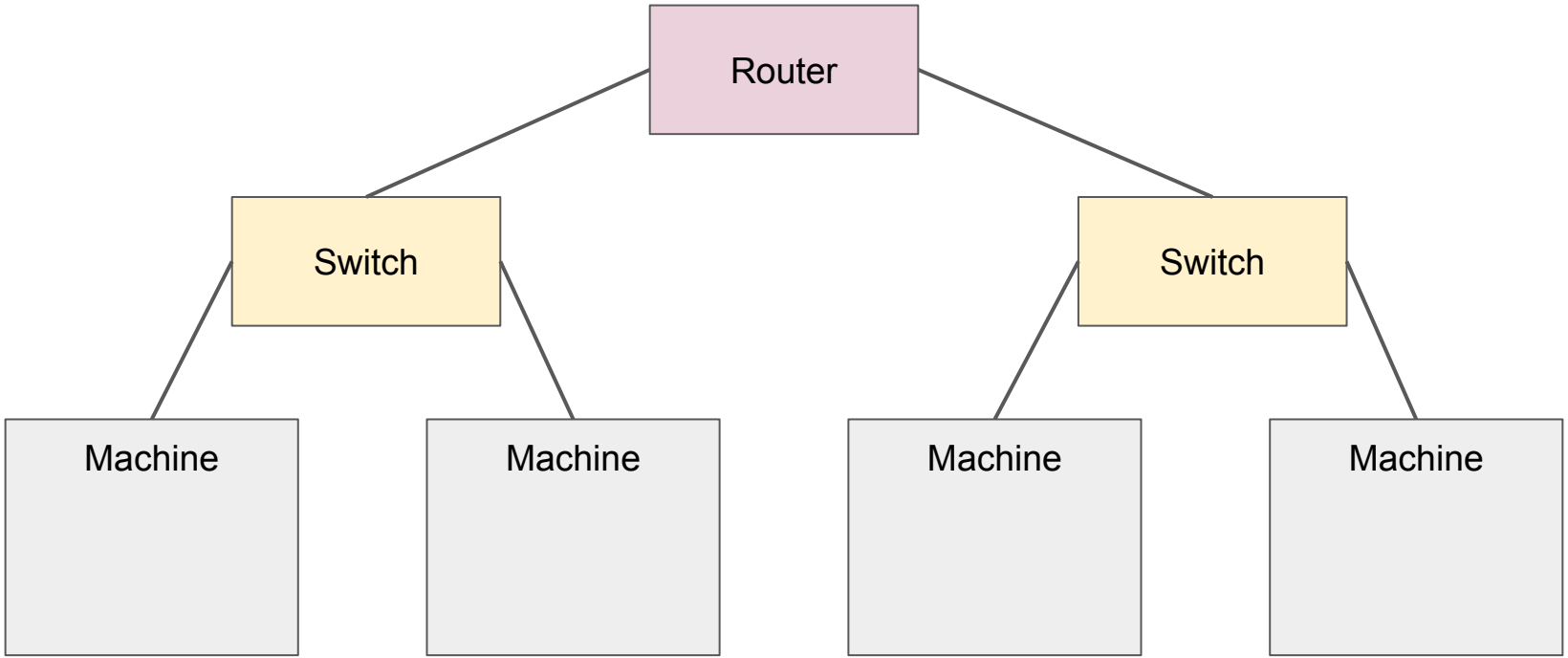
Networking Evolution

What changed since 80s?

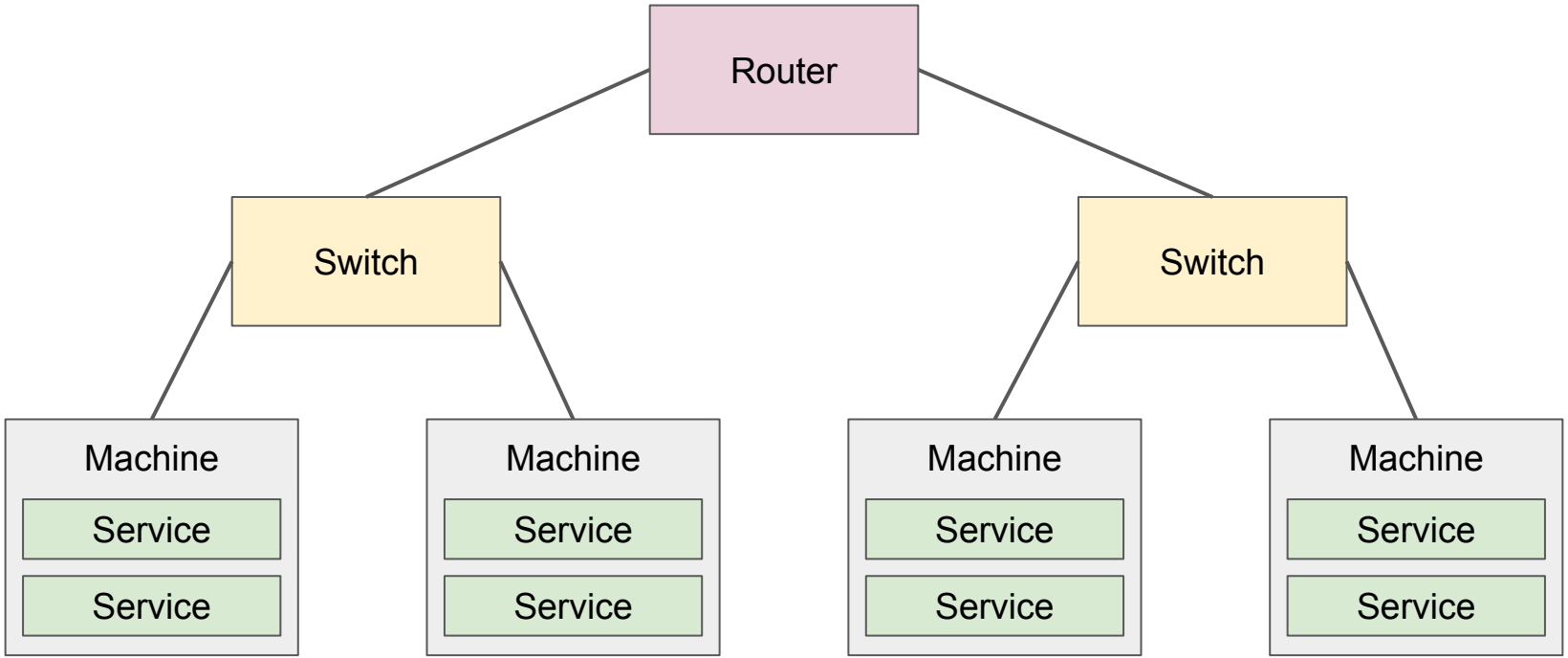
Evolution

physical ages

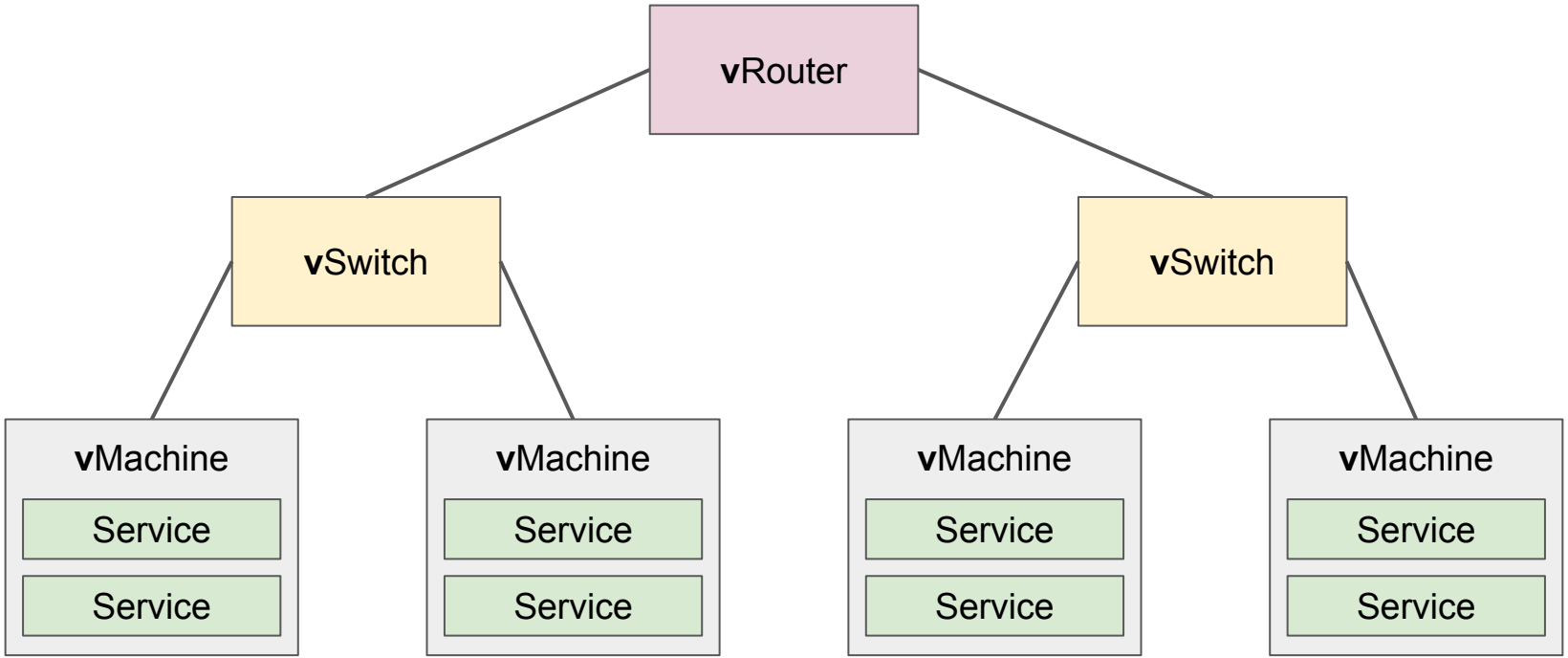
Evolution **physical ages**



Evolution **physical ages**



Evolution **virtual ages**



Evolution

cloud ages

Evolution

cloud ages

Evolution

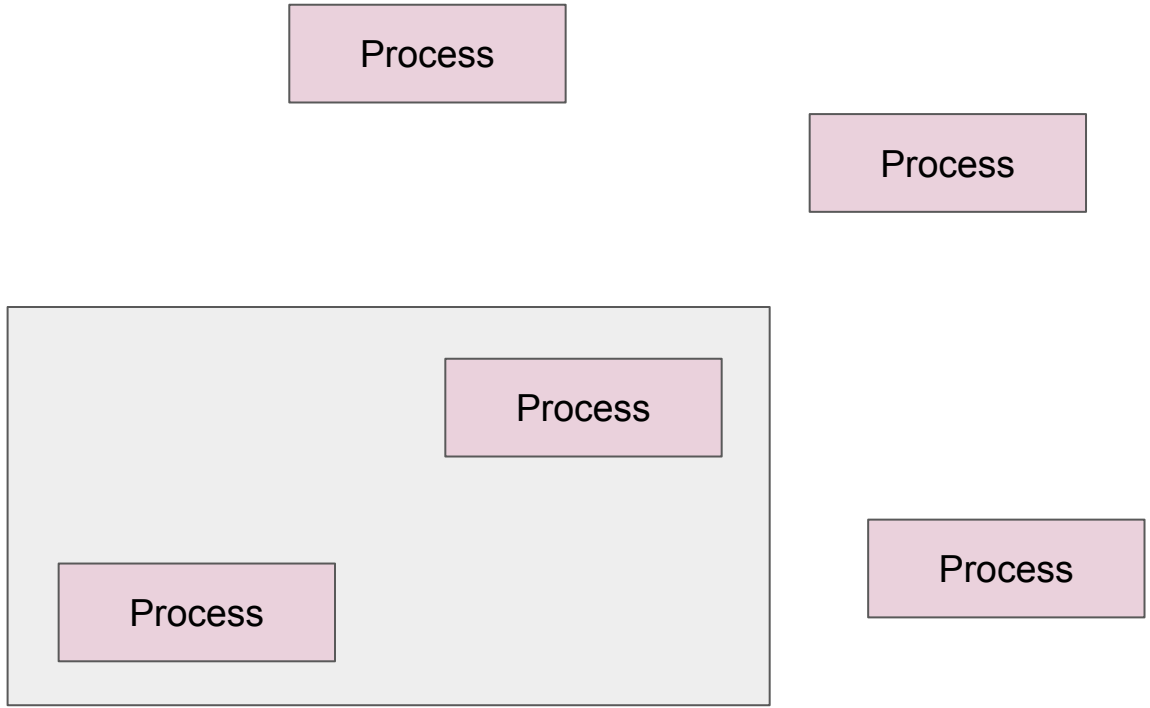
- So that's why we need different networking concept for containers

Connecting Containers

Connecting Containers

Little bit of (necessary) internals, what are containers and how do we connect them?

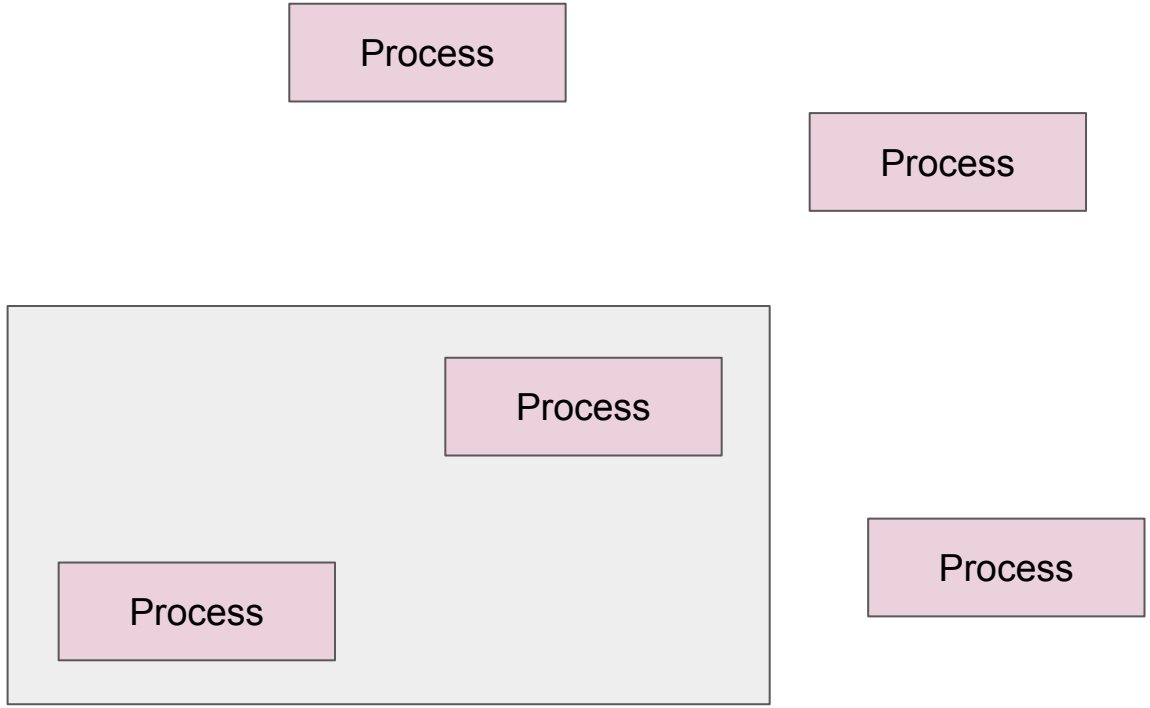
Container?!



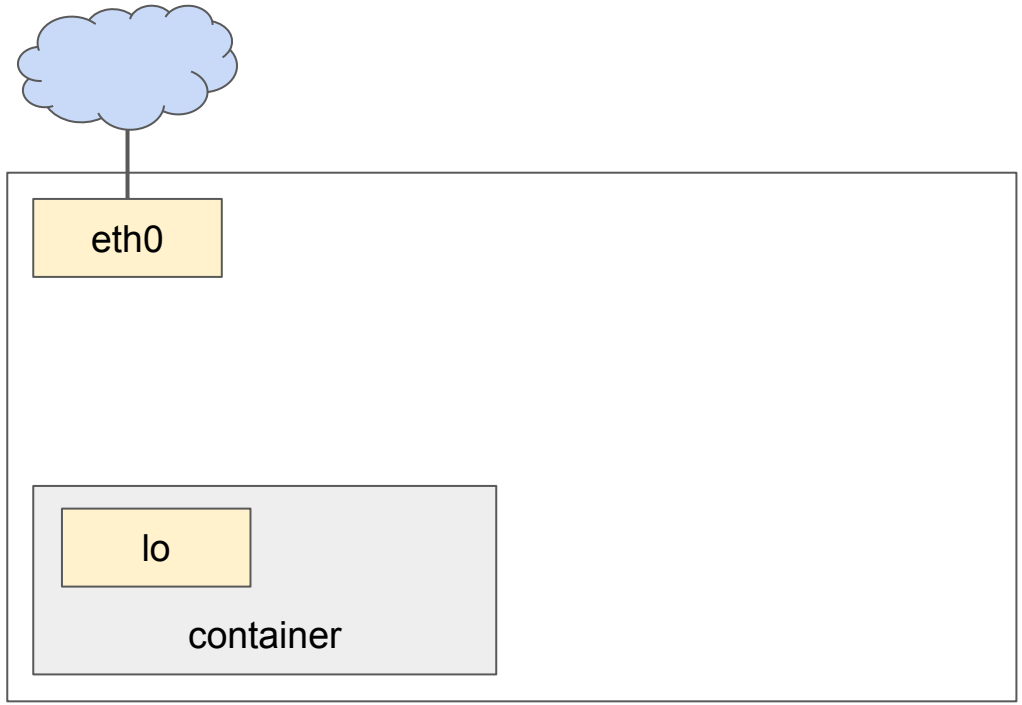
Container?!

Namespaces

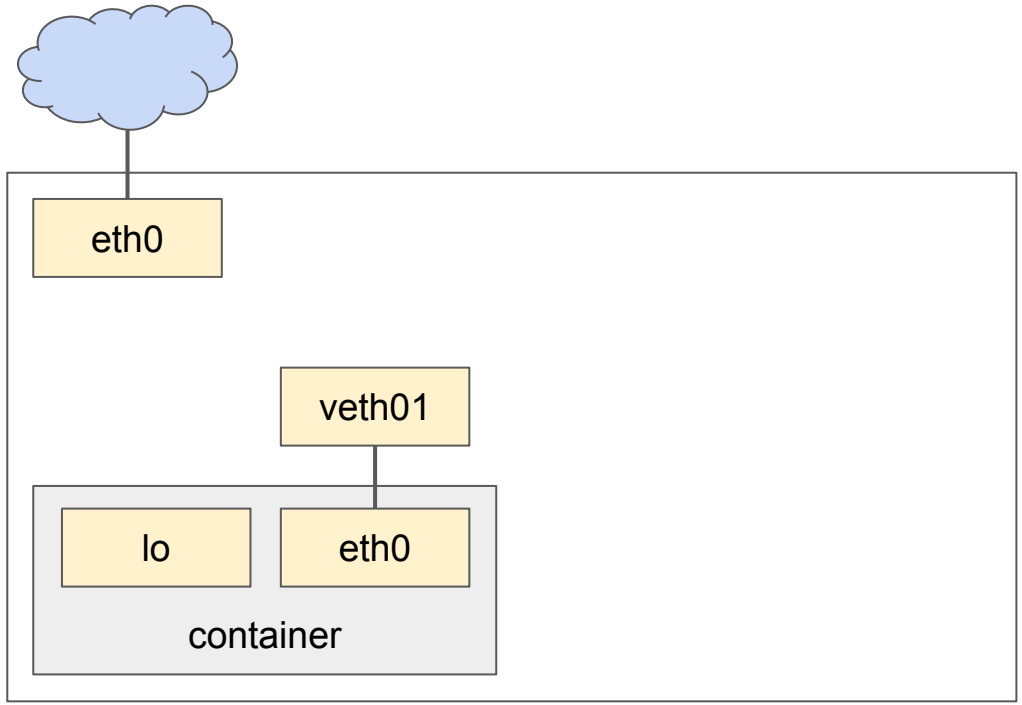
- resources
- filesystem
- processes
- network



Connecting **to host**

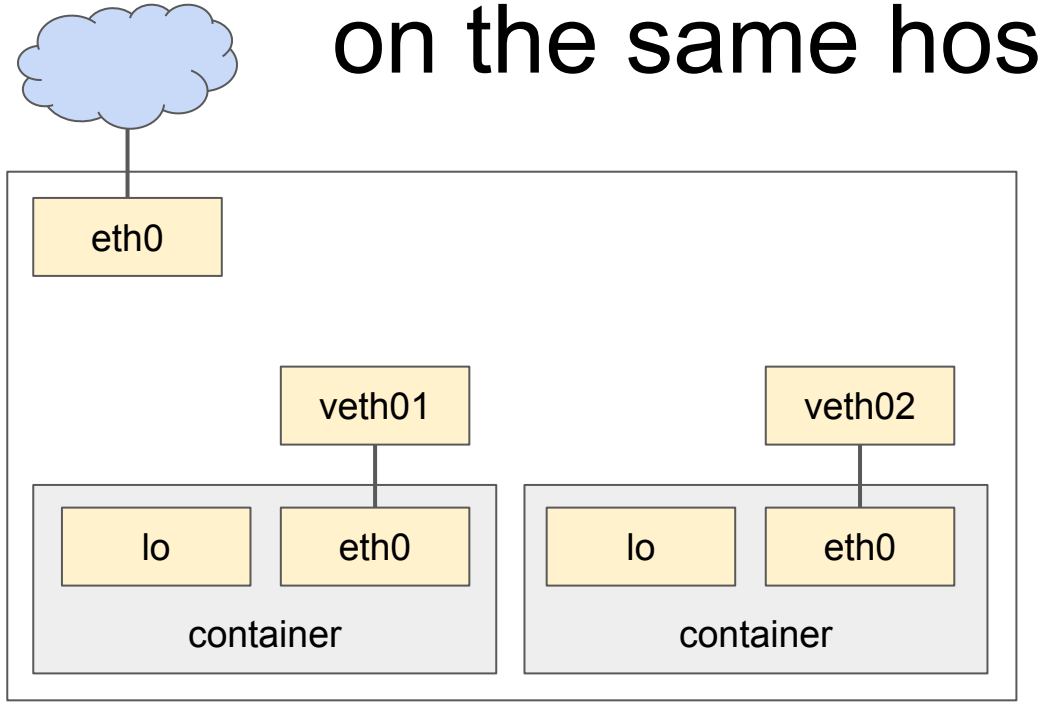


Connecting **to host**



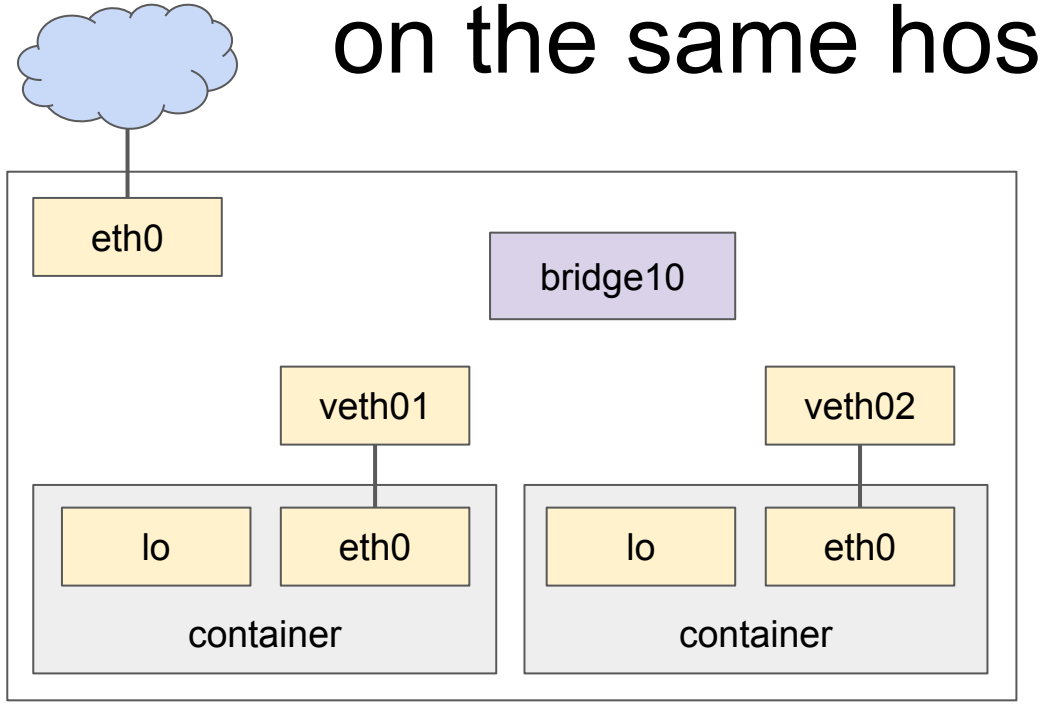
Connecting

between containers on the same host



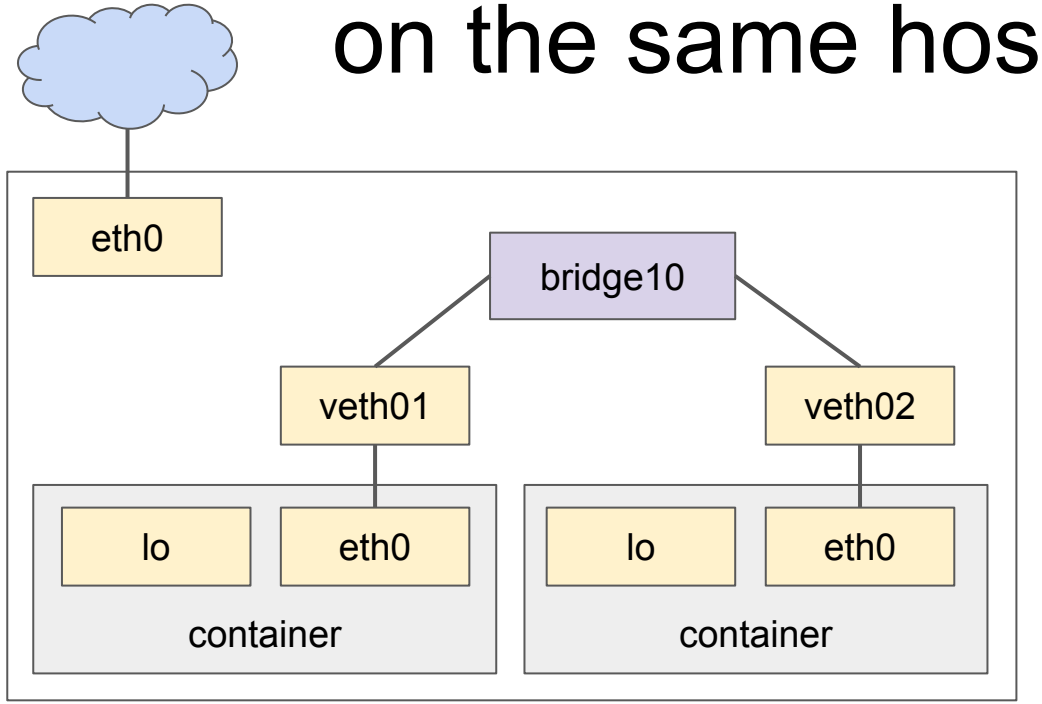
Connecting

between containers on the same host

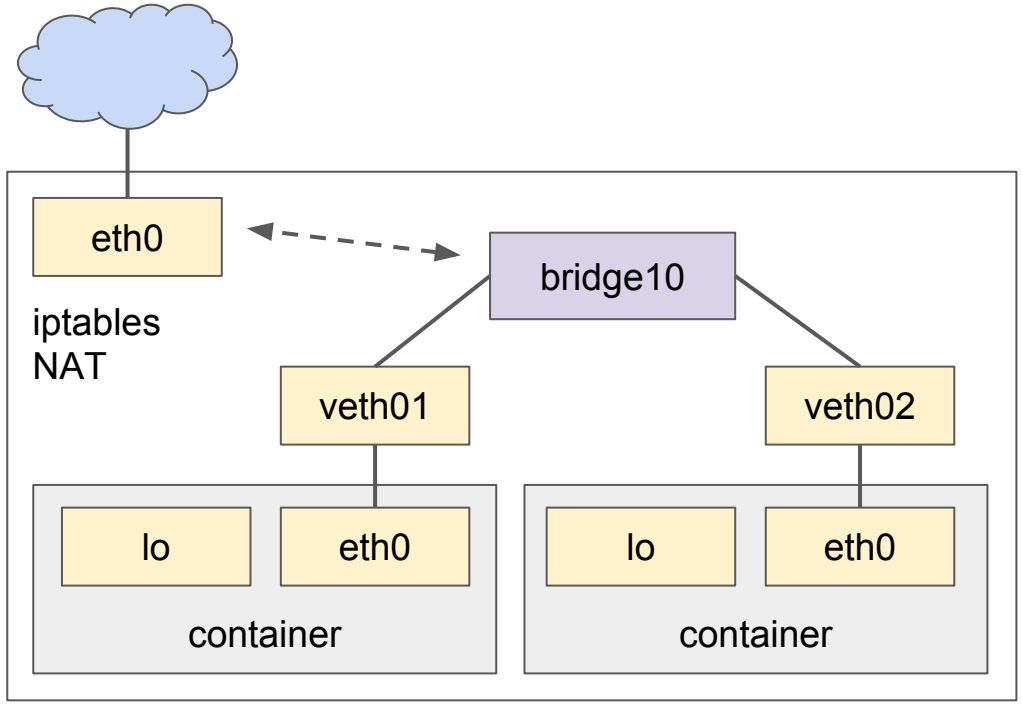


Connecting

between containers on the same host



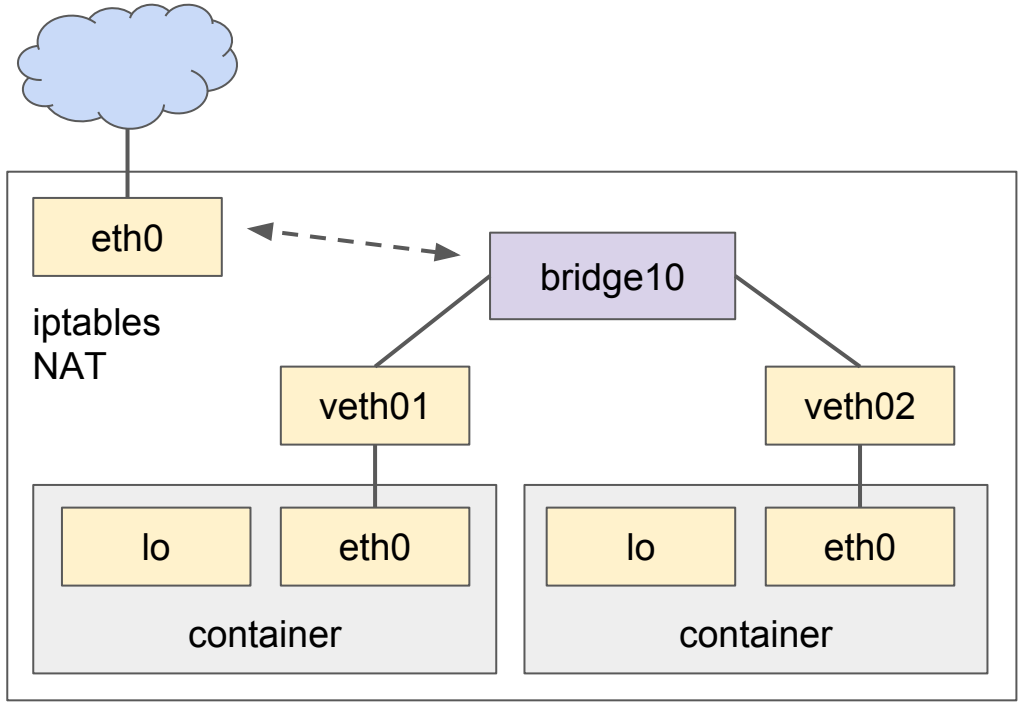
Connecting **to internet**



Connecting

to internet

- Lies...



Connecting containers

- Using virtual interfaces to connect namespaces

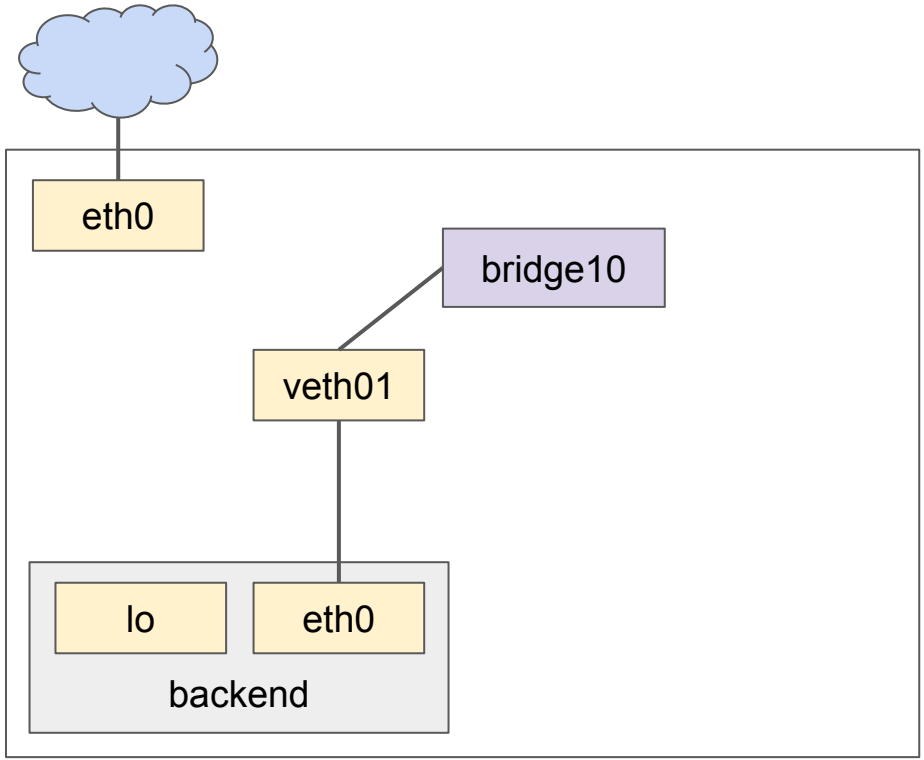
Connecting Containers

Connecting Containers In Kubernetes

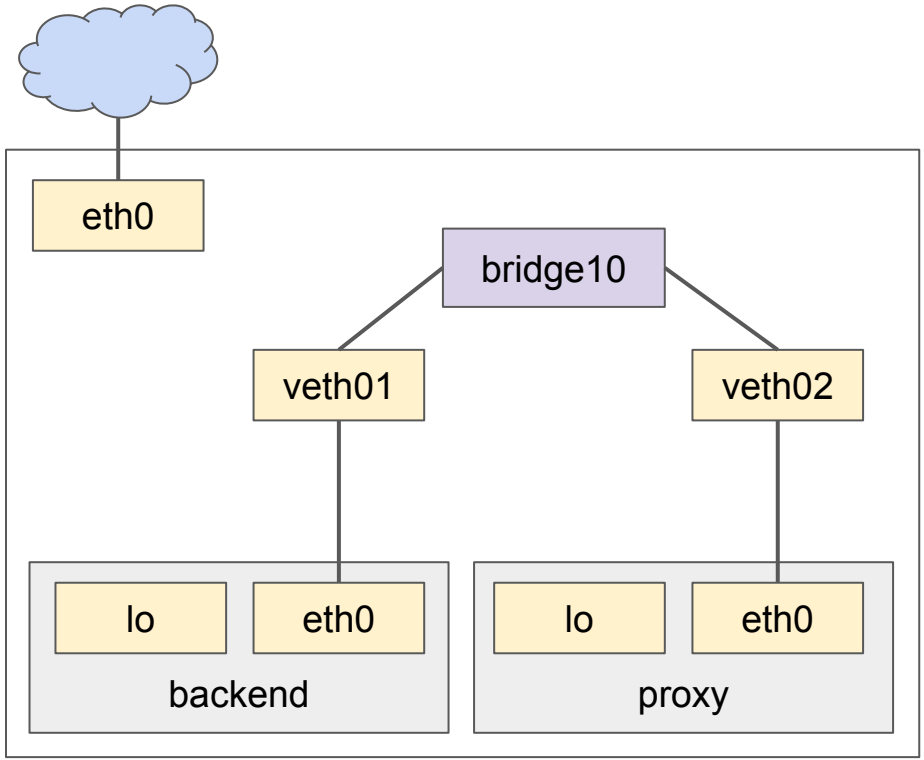
Connecting Containers In Kubernetes

There is a lot of networking going on in Kubernetes! It is exciting and charming in its simplicity… But it may get a little overwhelming at first.

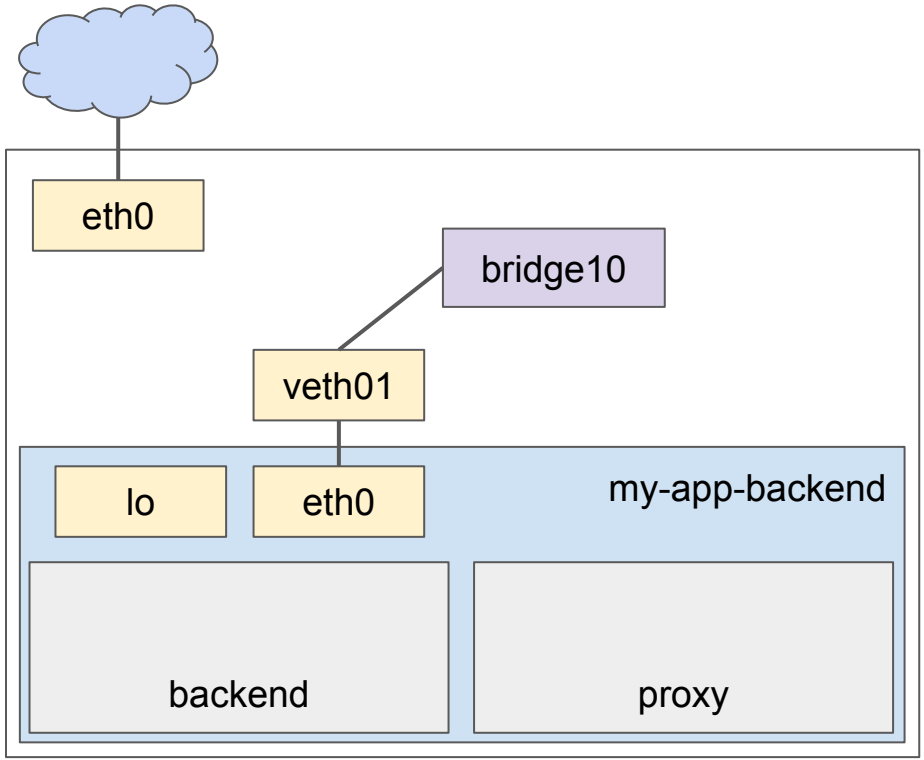
Actually, connecting Pods, not containers



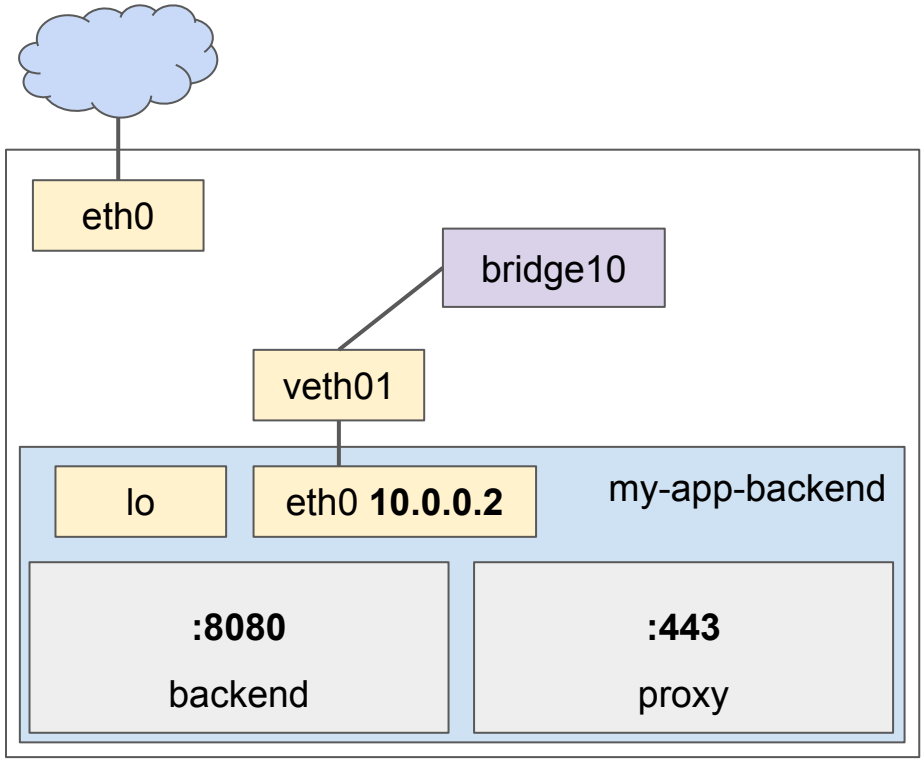
Actually, connecting Pods, not containers



Actually, connecting Pods, not containers

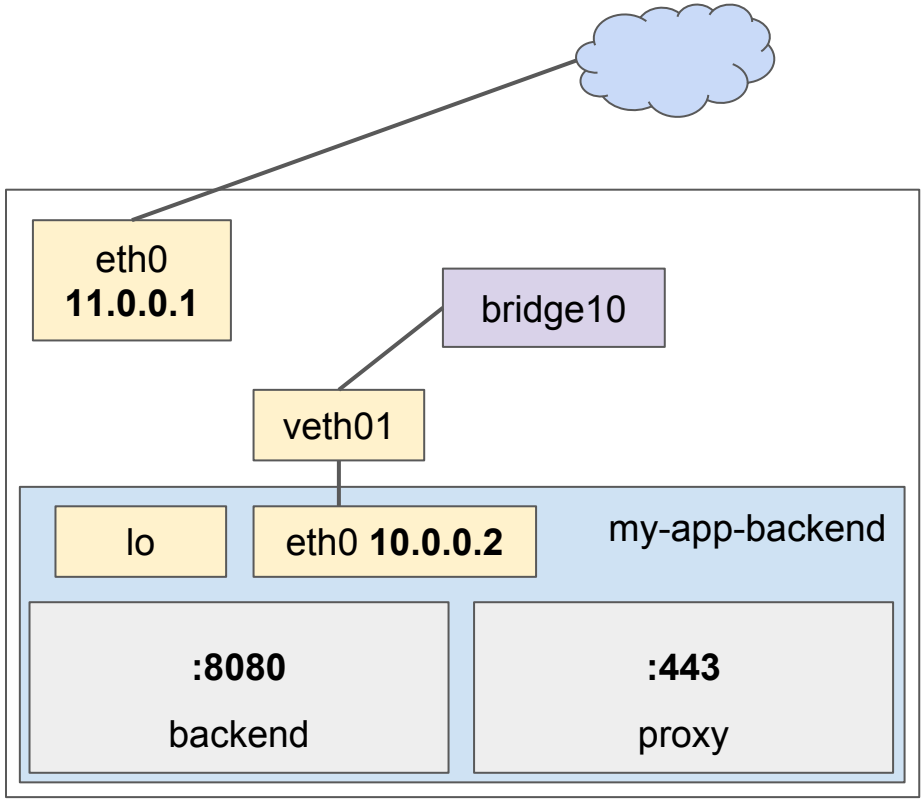


Actually, connecting Pods, not containers



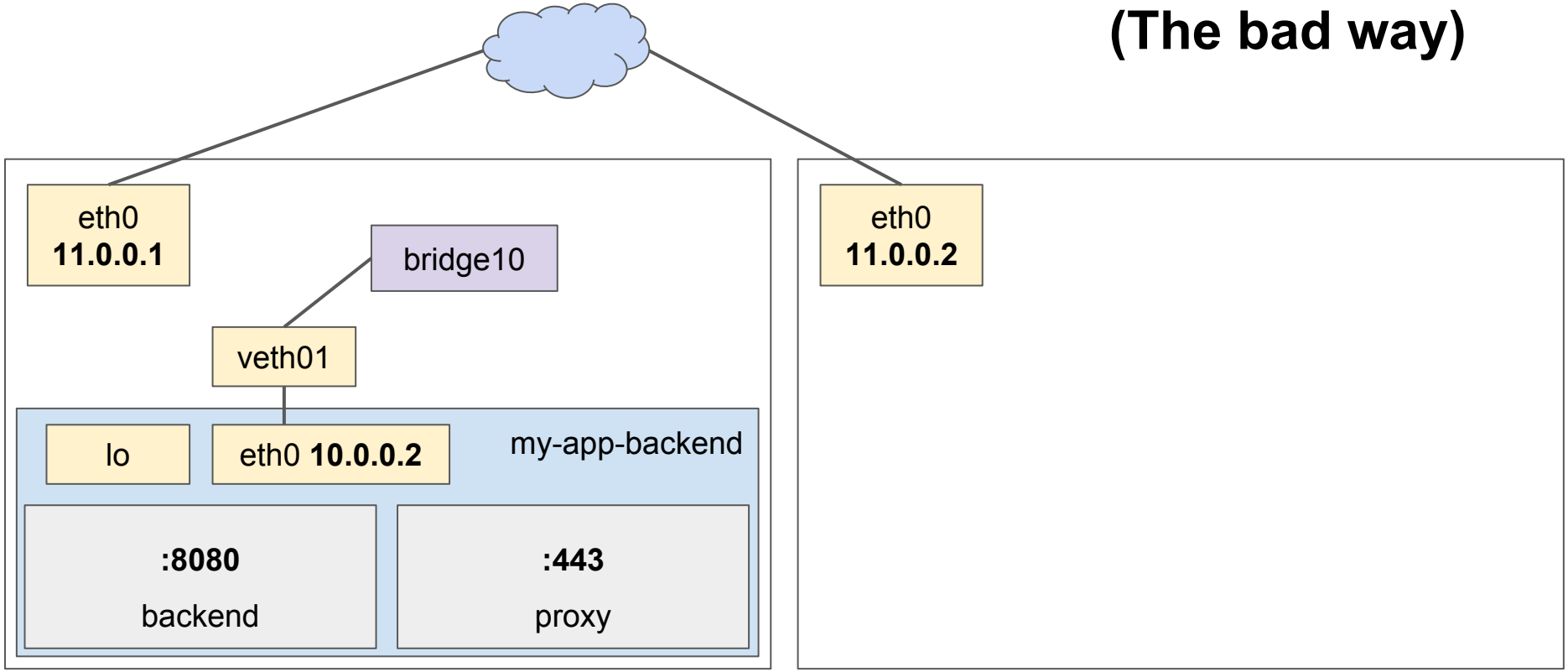
How do I access Pods on other Nodes?

(The bad way)



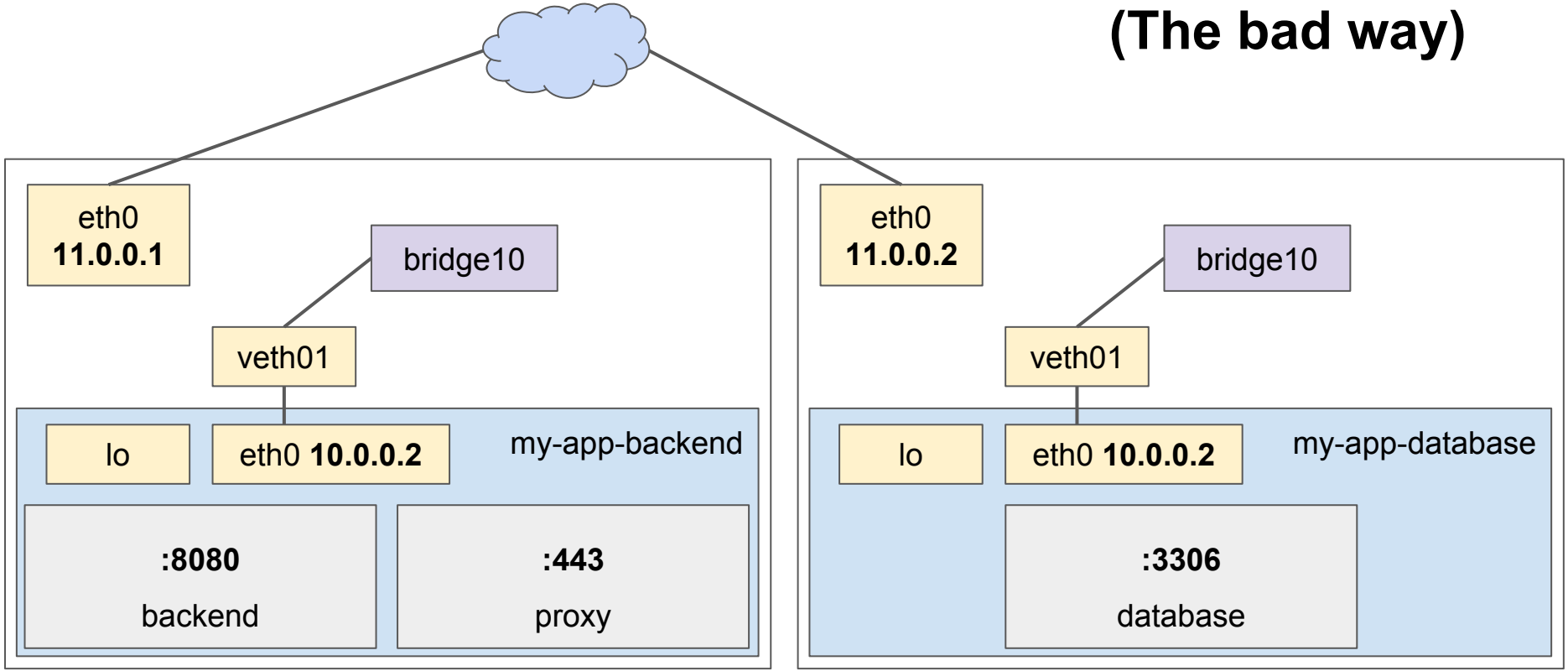
How do I access Pods on other Nodes?

(The bad way)



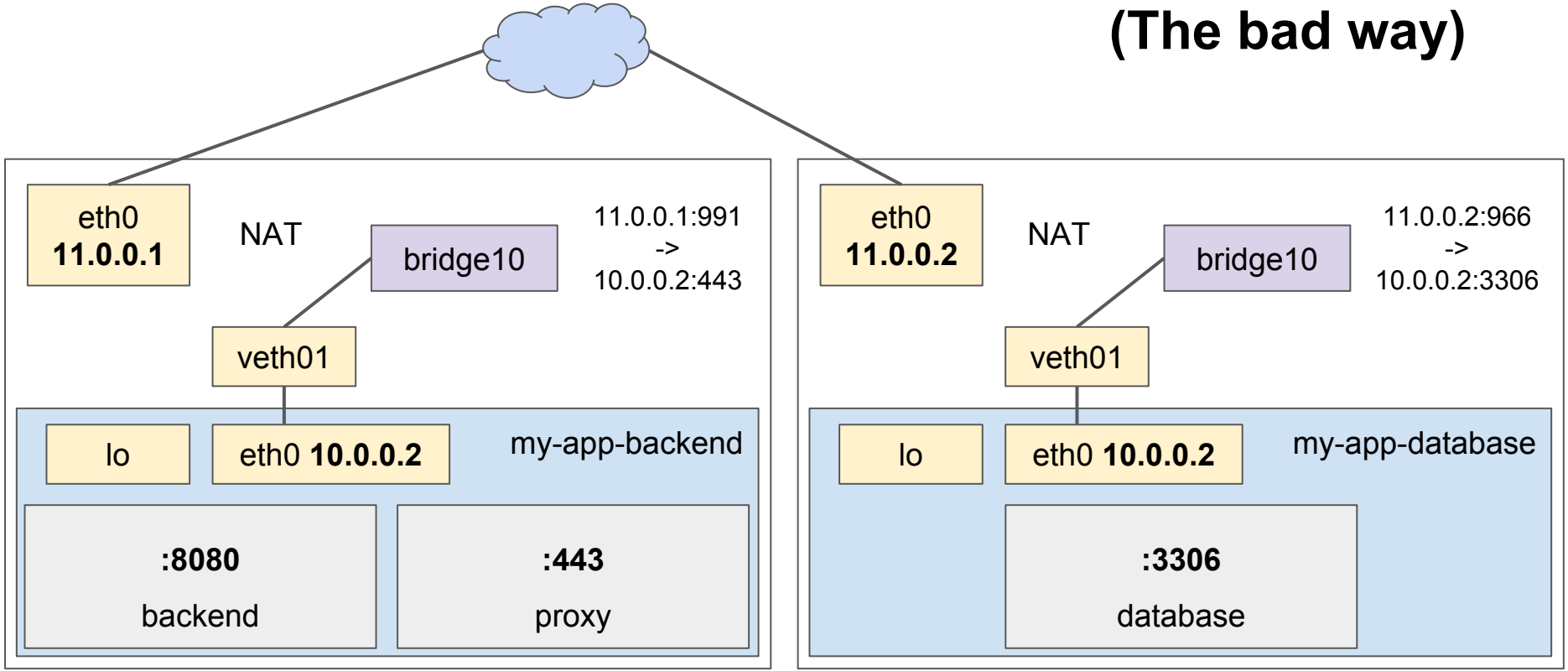
How do I access Pods on other Nodes?

(The bad way)

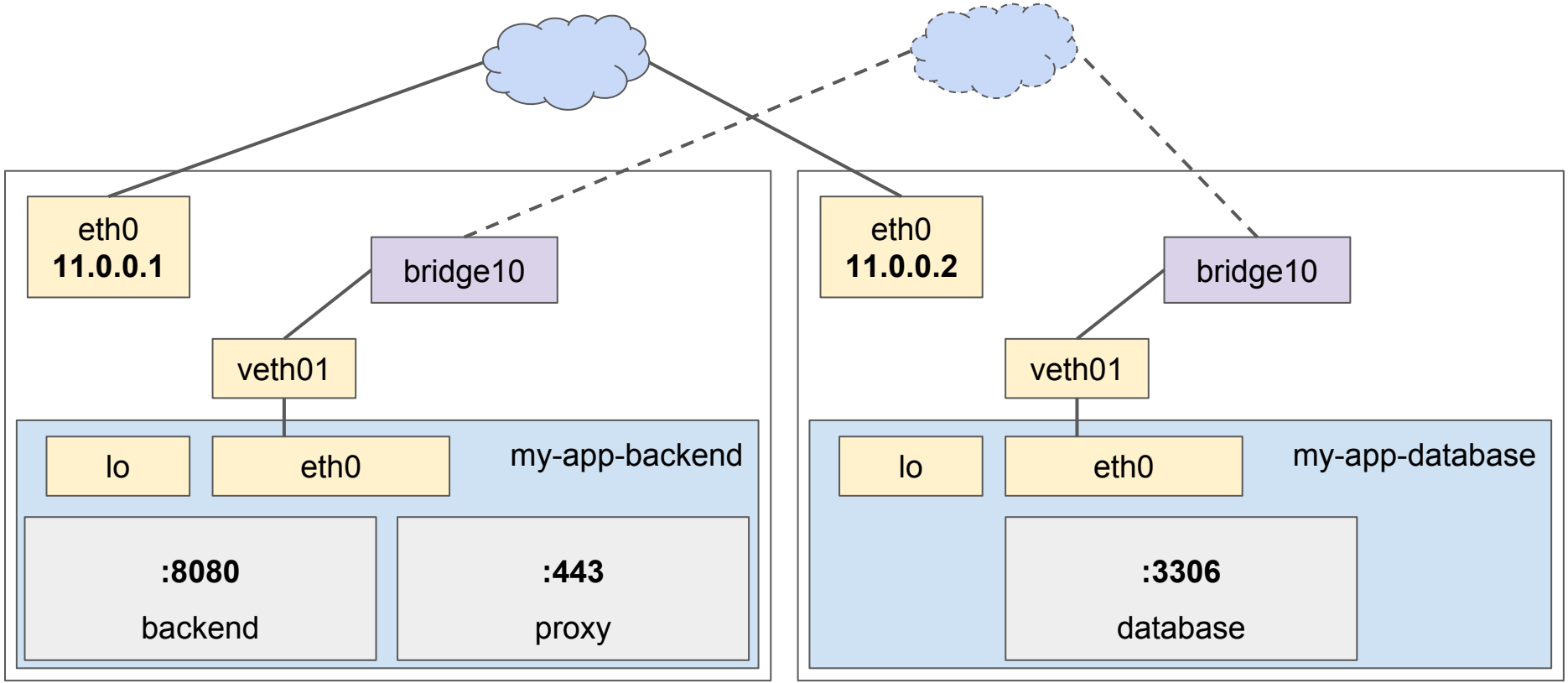


How do I access Pods on other Nodes?

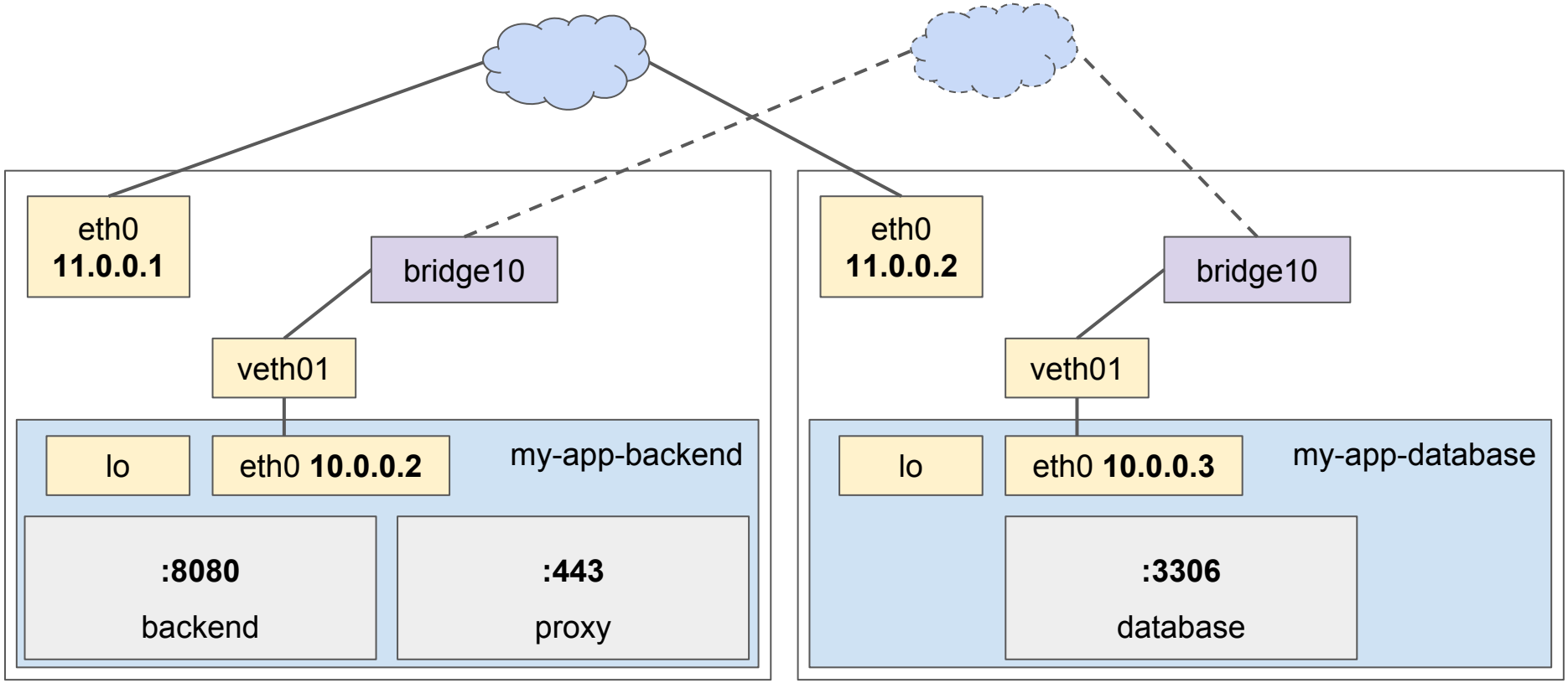
(The bad way)



Pods can reach one another



One IP per Pod, always



Kubernetes doesn't handle networks

Kubernetes doesn't handle networks

Connecting containers in Kubernetes

- How life of a packet in Kubernetes looks like

Hands on

Hands on

Pods, Services, Endpoints, Ingress, NetworkPolicies, …

Let's use them (and take a closer look).

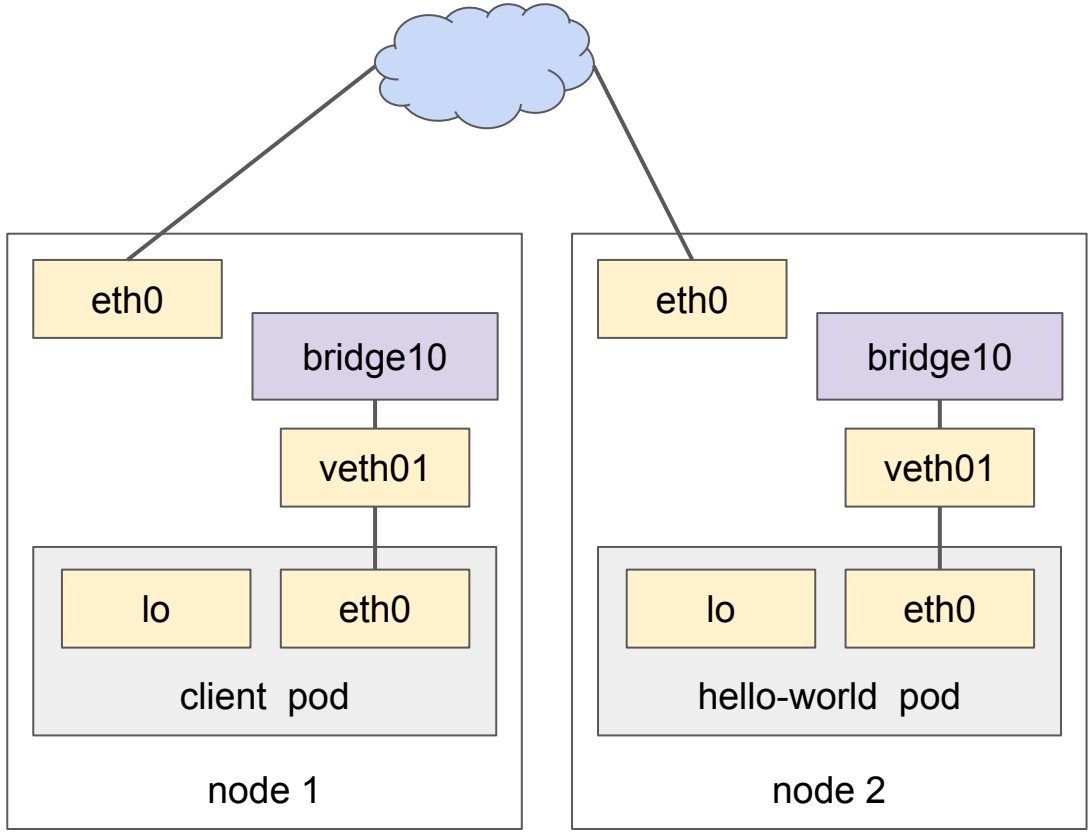
Pod again: Spec

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-world
spec:
  containers:
  - name: hello-world
    image: hello-world:1.0
    ports:
    - containerPort: 80
```

Pod again: State

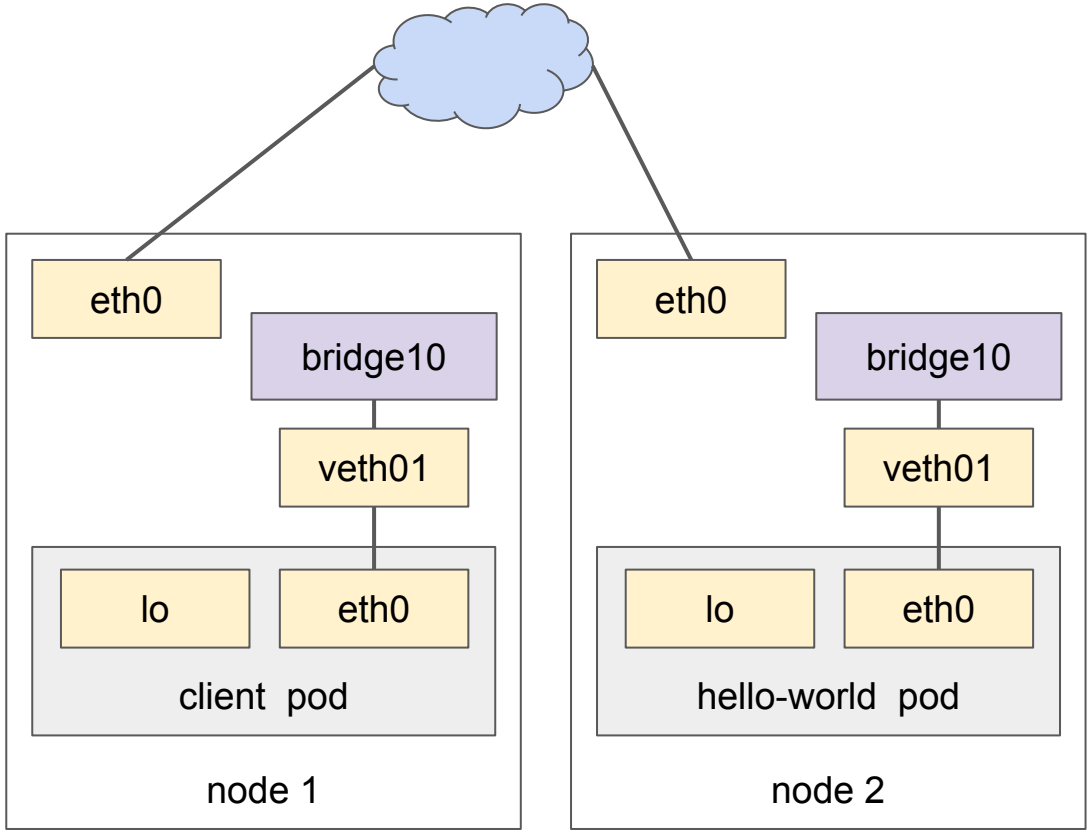
```
apiVersion: v1
kind: Pod
metadata:
  ...
  name: hello-world
  ...
spec:
  ...
status:
  ...
  podIP: 10.244.0.7
  ...
```

Pod again: Connecting



Pod again: Connecting

```
client $ curl 10.244.0.7
"Hello World"
```



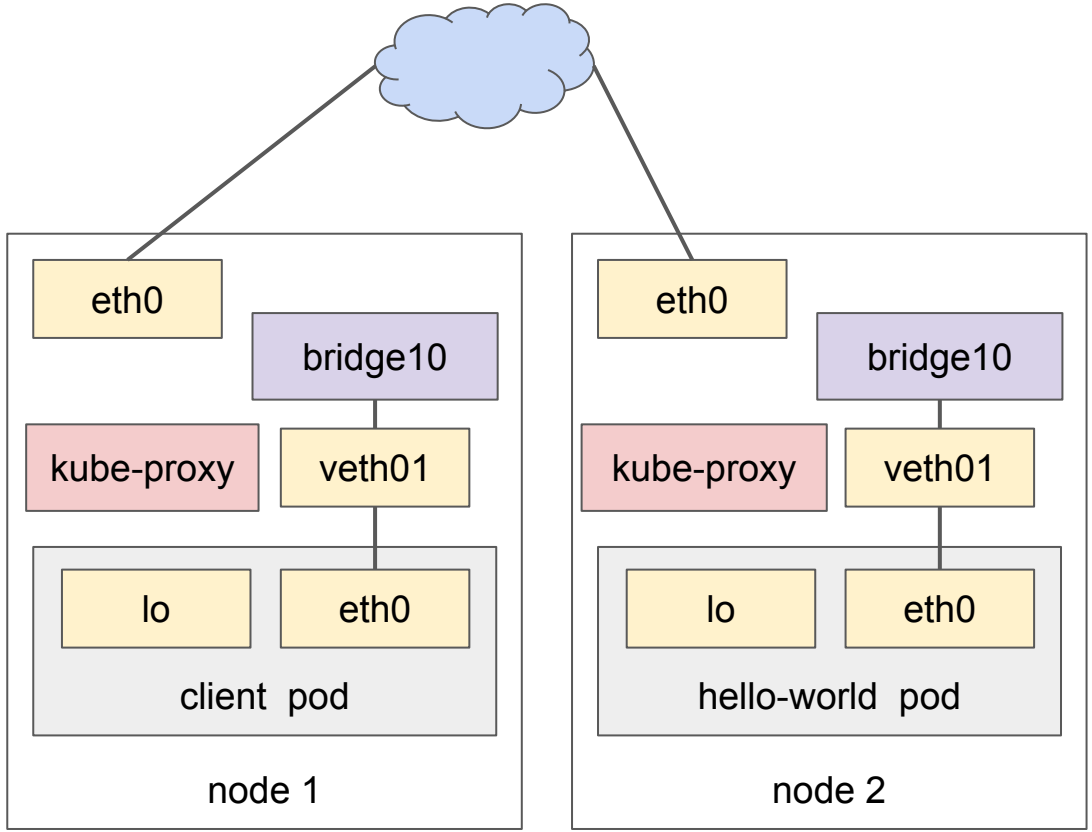
Exposing a Service: Spec

```
apiVersion: v1
kind: Service
metadata:
  name: hello-world-service
spec:
  type: ClusterIP
  ports:
  - port: 8080
    targetPort: 80
    protocol: TCP
  selector:
    app: hello-world
```

Exposing a Service: State

```
apiVersion: v1
kind: Service
metadata:
  ...
  name: hello-world-service
  ...
spec:
  clusterIP: 10.103.8.175
  ...
status:
  ...
```

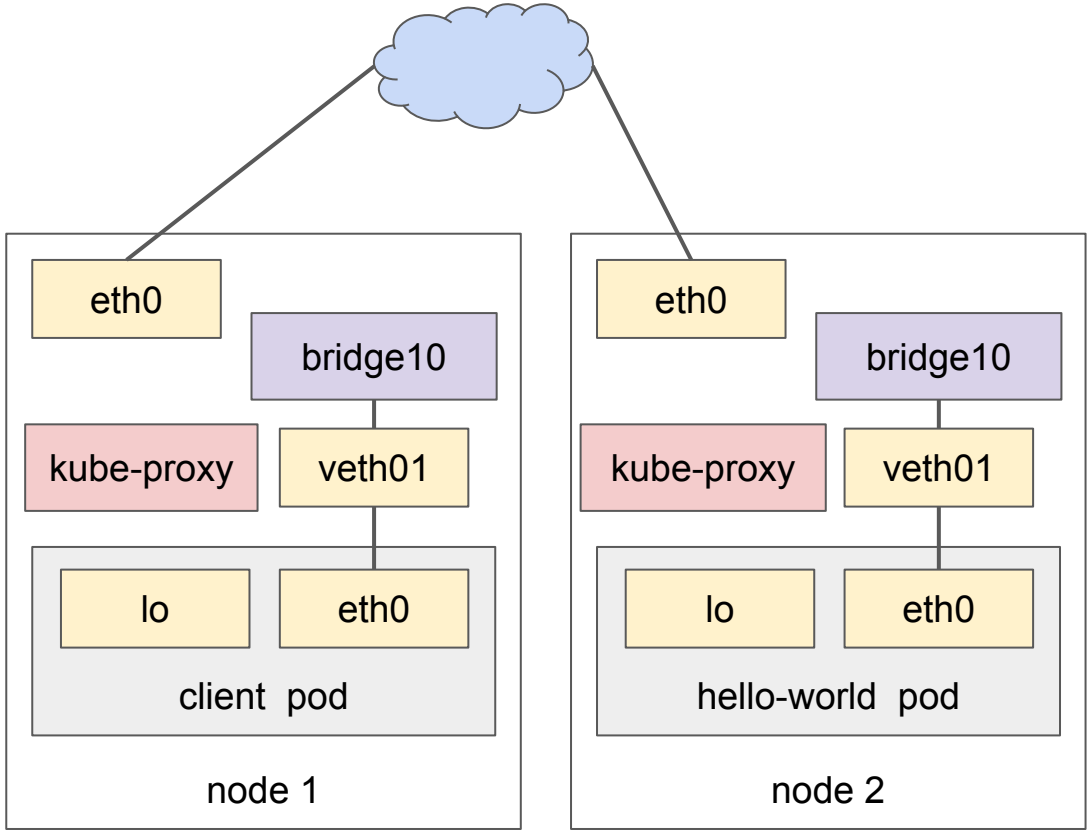
Exposing a Service: Connecting



Exposing a Service: Connecting

```
client $ curl 10.103.8.175
"Hello World"

client $ curl
hello-world-service
"Hello World"
```

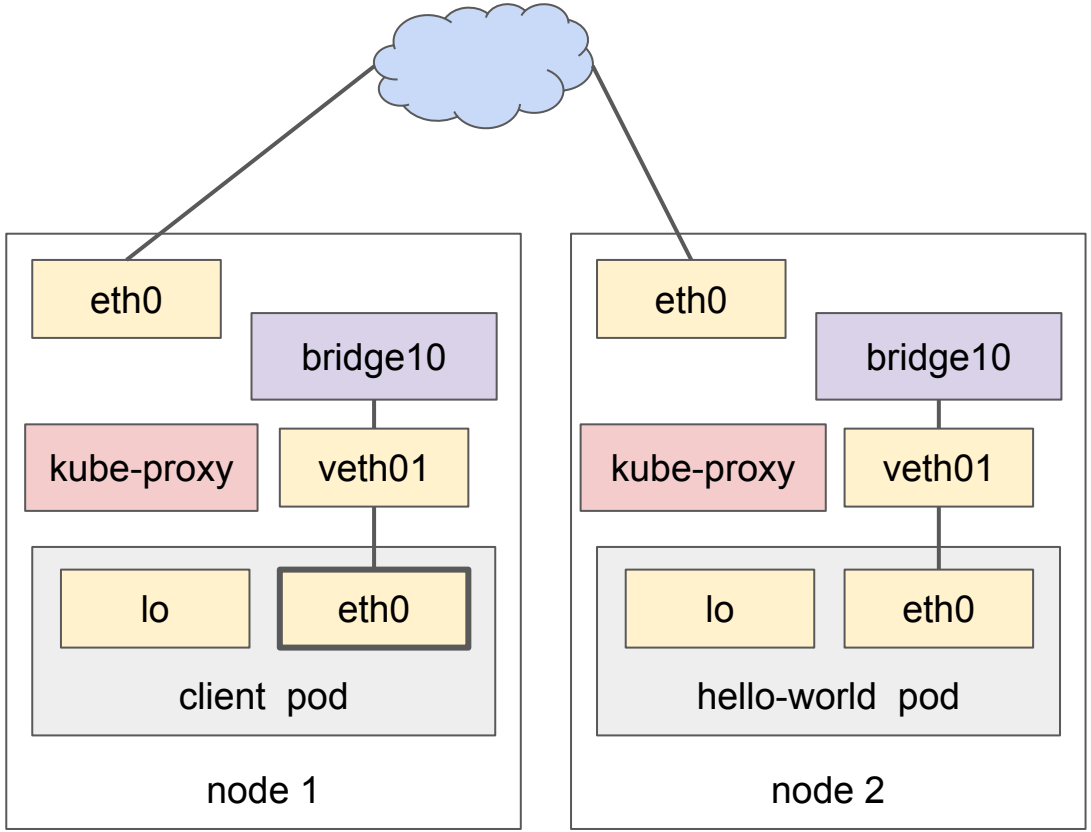


Exposing a Service: Connecting

```
client $ curl 10.103.8.175
"Hello World"

client $ curl
hello-world-service
"Hello World"
```

From: <eth0 IP>
To: <hello-world service IP>
Data: ...

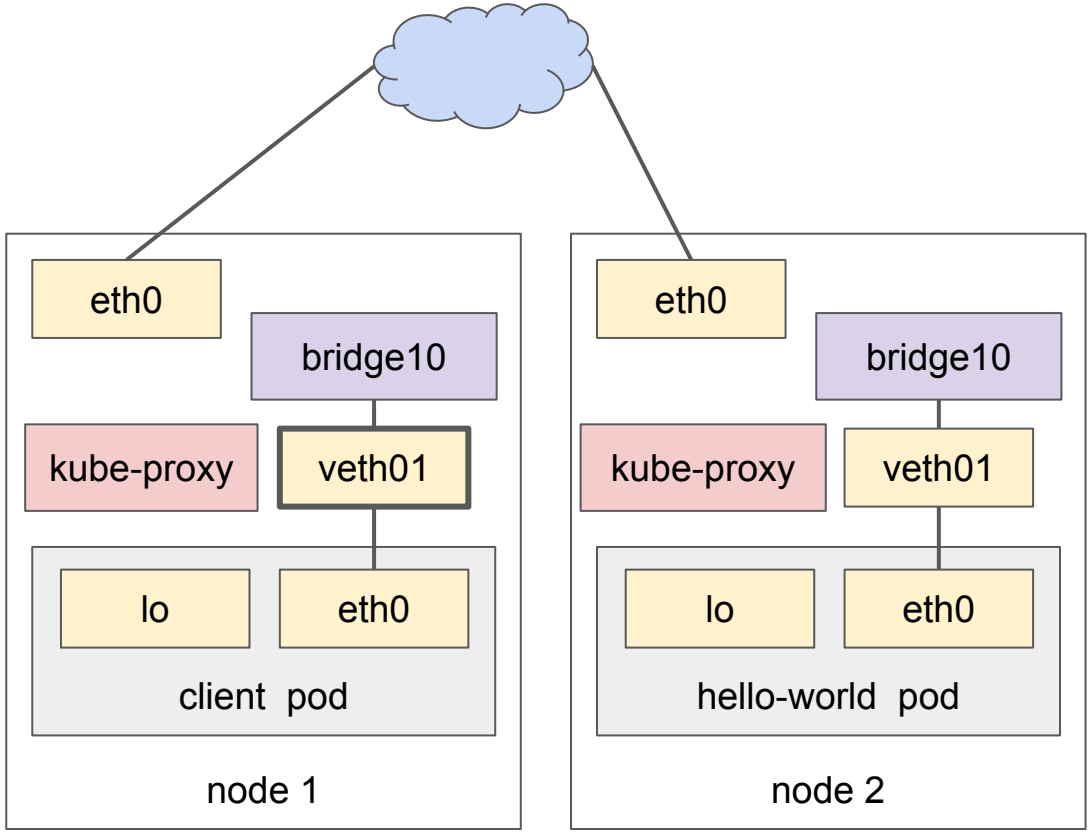


Exposing a Service: Connecting

```
client $ curl 10.103.8.175
"Hello World"

client $ curl
hello-world-service
"Hello World"
```

From: <eth0 IP>
To: <hello-world service IP>
Data: ...

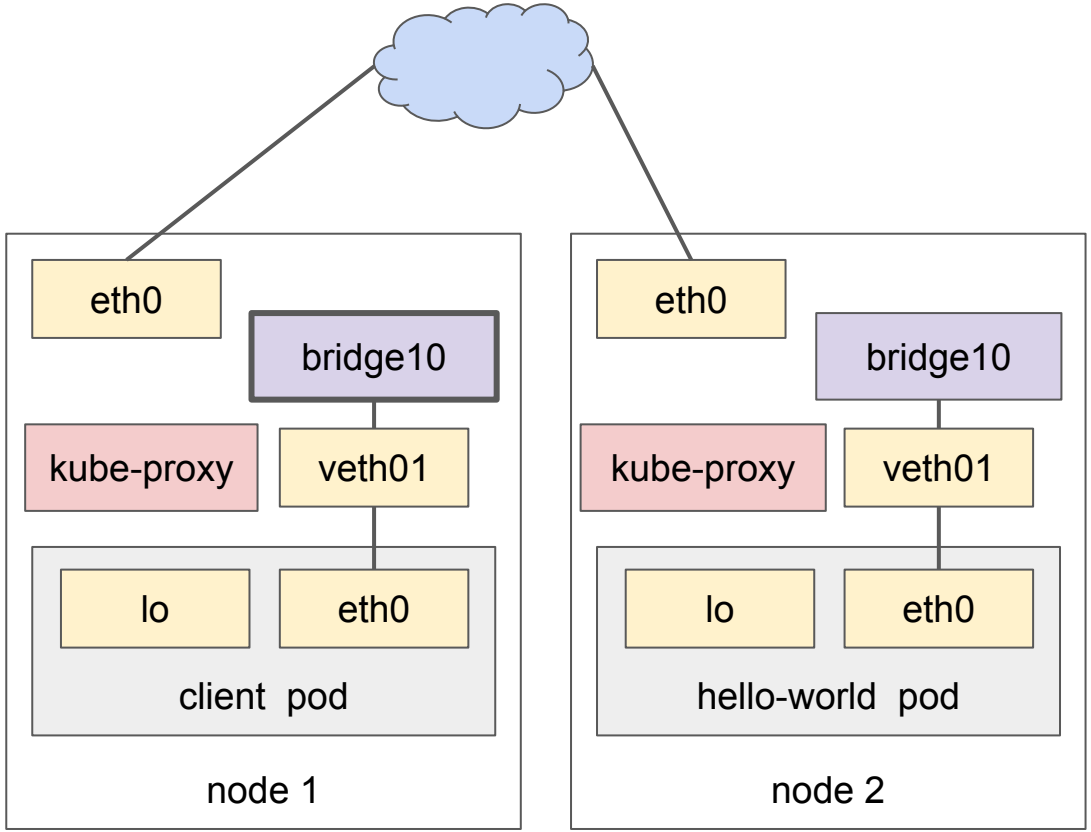


Exposing a Service: Connecting

```
client $ curl 10.103.8.175
"Hello World"

client $ curl
hello-world-service
"Hello World"
```

From: <eth0 IP>
To: <hello-world service IP>
Data: ...

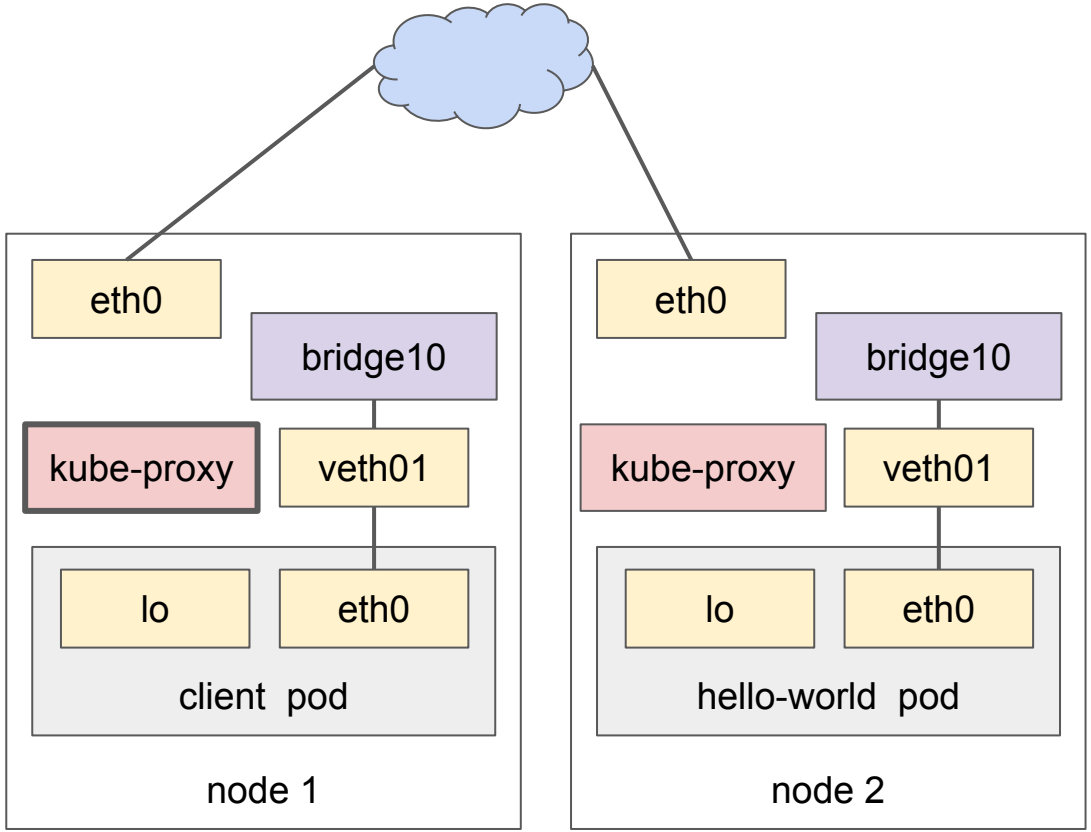


Exposing a Service: Connecting

```
client $ curl 10.103.8.175
"Hello World"

client $ curl
hello-world-service
"Hello World"
```

From: <eth0 IP>
To: **<hello-world service IP>**
Data: ...

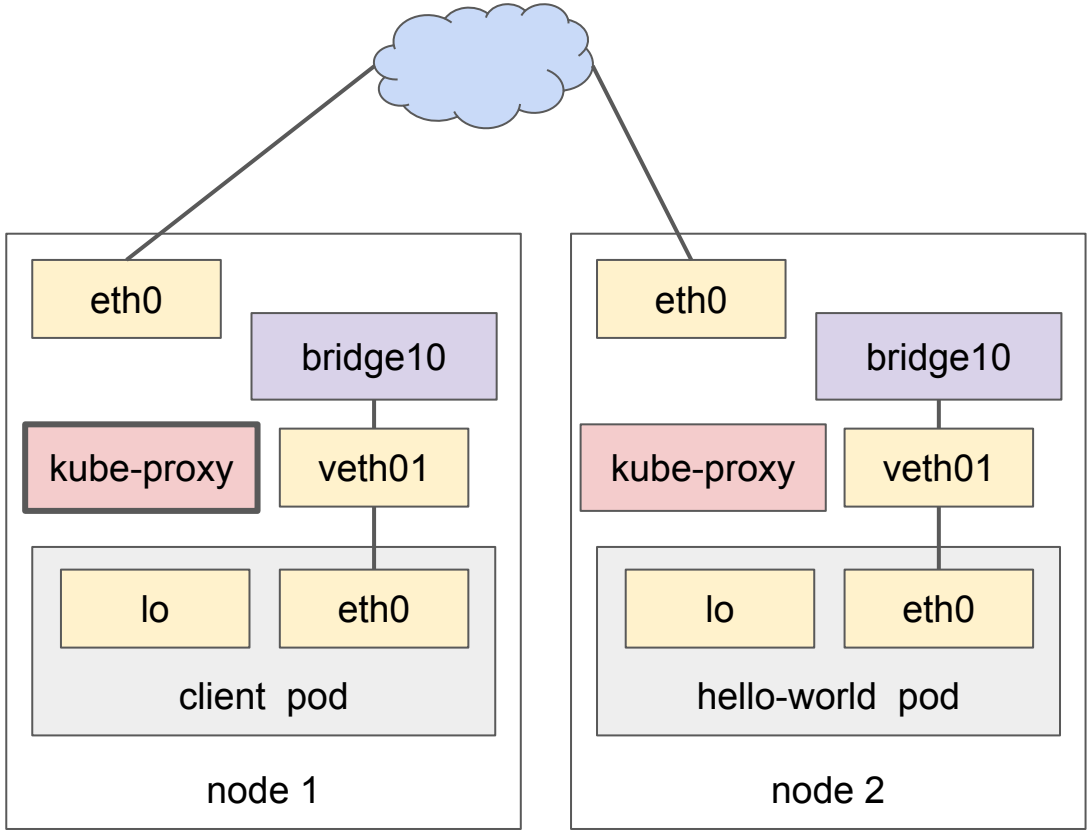


Exposing a Service: Connecting

```
client $ curl 10.103.8.175
"Hello World"

client $ curl
hello-world-service
"Hello World"
```

From: <eth0 IP>
To: **<random hello-world pod IP>**
Data: ...



Getting outside

- iptables NAT on each host

Getting inside

- Network load balancing (L4)
- HTTP load balancing (L7)

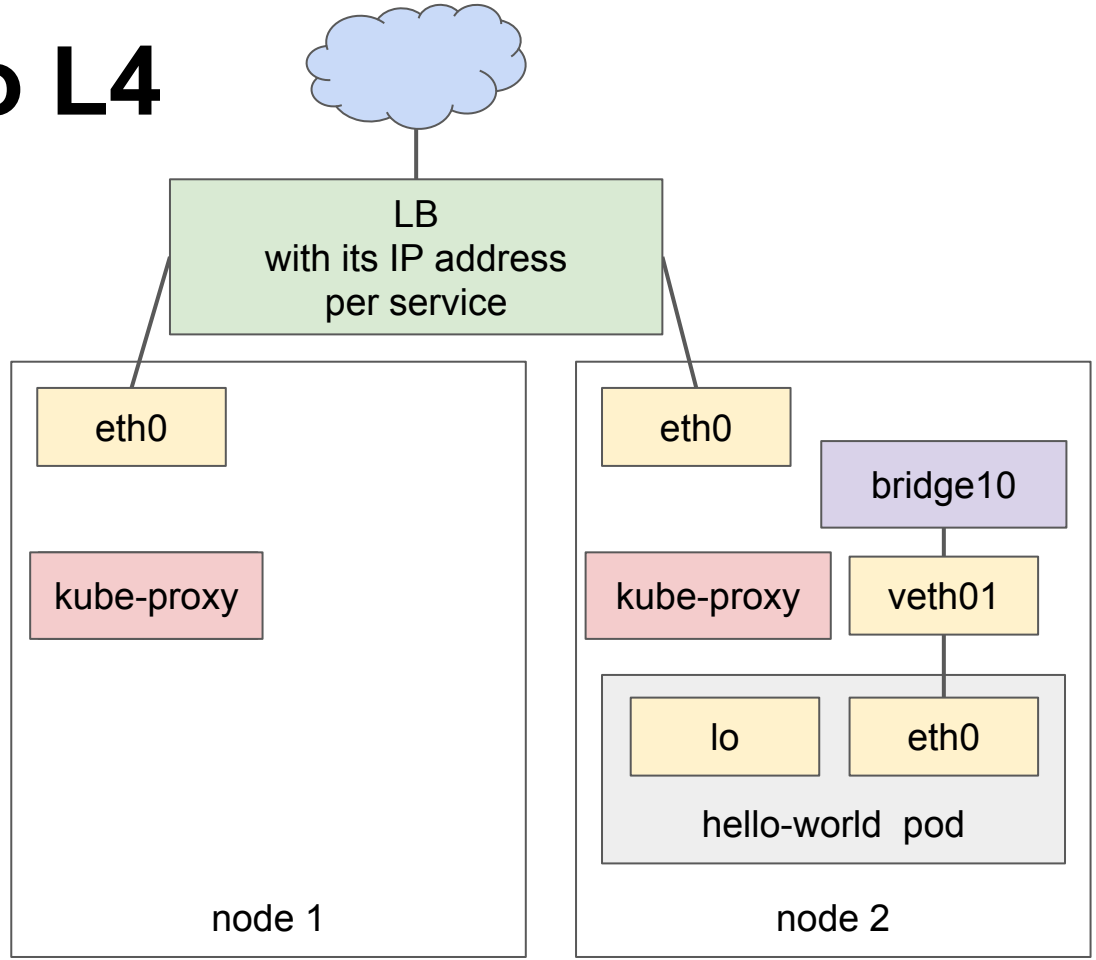
Getting inside **to L4**
Service: Spec

```
apiVersion: v1
kind: Service
metadata:
  name: hello-world
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 80
  selector:
    app: hello-world
```

Getting inside **to L4**
Service: State

```
apiVersion: v1
kind: Service
metadata:
  name: hello-world
spec:
  clusterIP: 10.103.8.176
  ...
state:
  loadBalancer:
    ingress:
    - ip: 86.105.20.11
```

Getting inside **to L4**

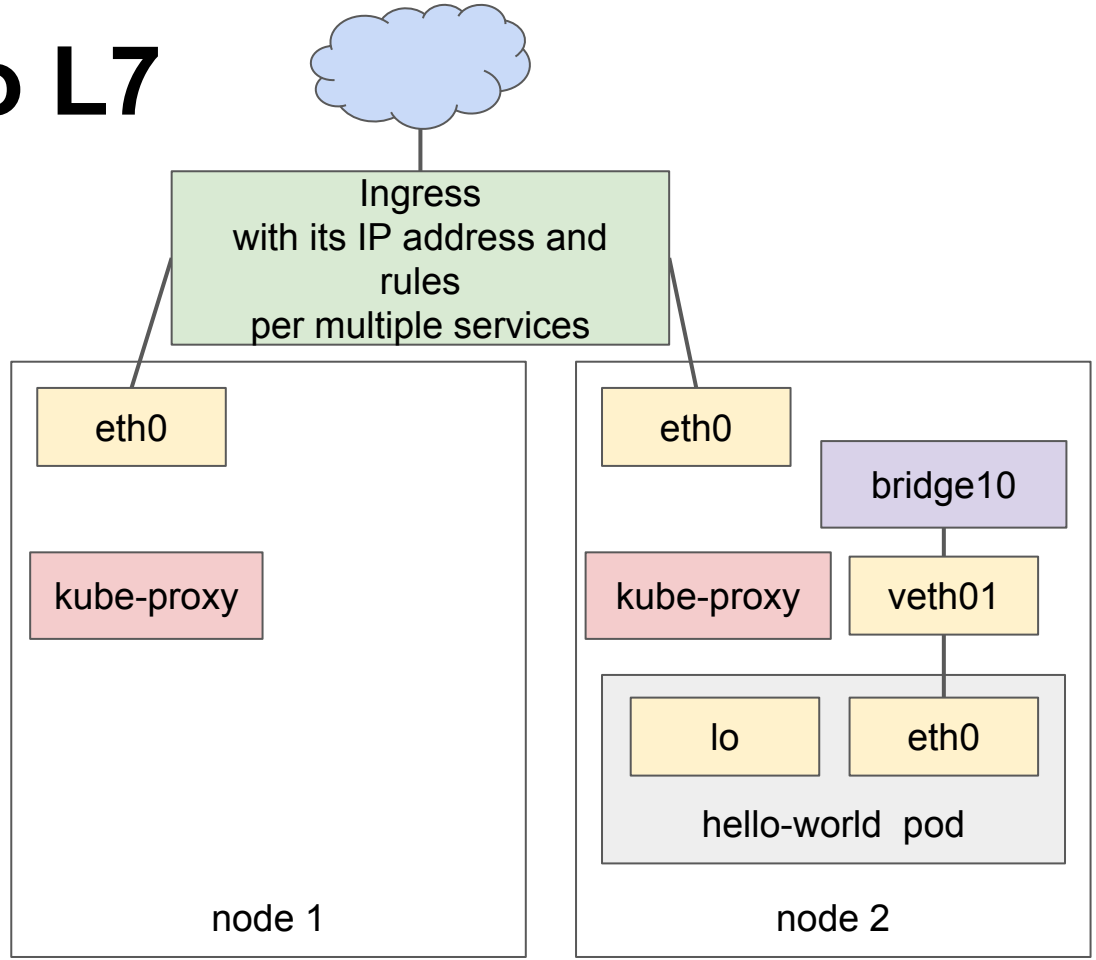


Getting inside **to L7**

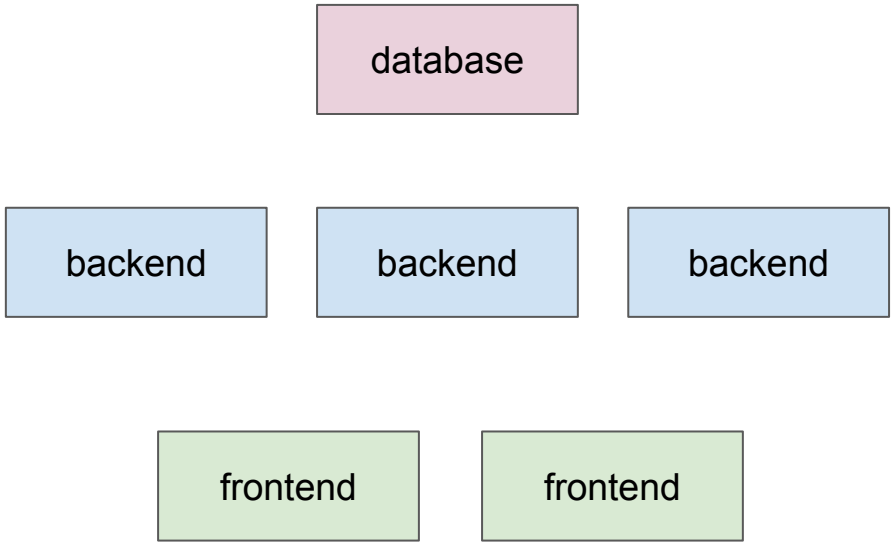
```
apiVersion:
extensions/v1beta1
kind: Ingress
metadata:
  name: hello-ingress
```

```
spec:
 rules:
 - host: world.hello.org
   http:
     paths:
     - backend:
         serviceName: hello-world
         servicePort: 8080
 - host: darkness.hello.org
   http:
     paths:
     - path: /void/*
       backend:
         serviceName: hello-darkness
         servicePort: 8080
```

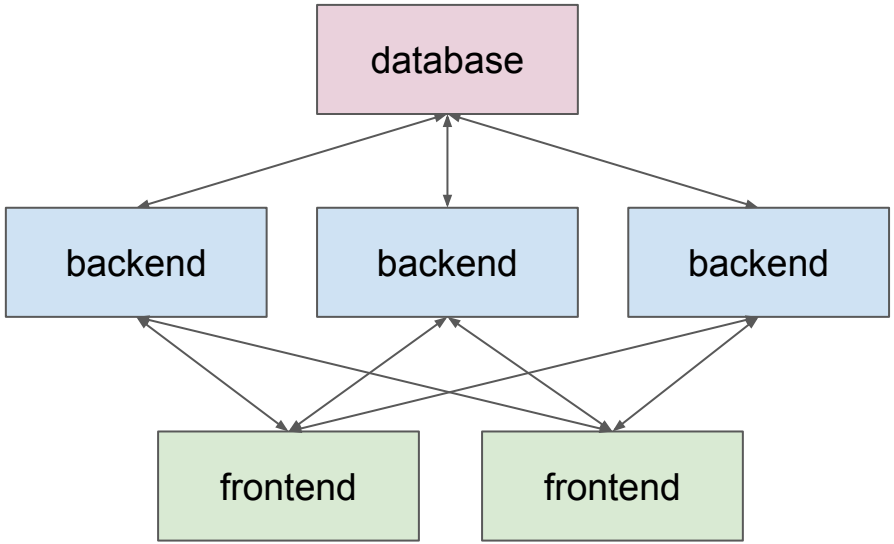
Getting inside **to L7**



Limiting Access



Limiting Access



Limiting Access: Spec

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
spec:
  podSelector:
    matchLabels:
      role: backend
  policyTypes:
  - Ingress
  - Egress
```

```
ingress:
- from:
  - podSelector:
      matchLabels:
        role: frontend
  ports:
  - protocol: TCP
    port: 6379
egress:
- to:
  - podSelector:
      matchLabels:
        role: database
  ports:
  - protocol: TCP
    port: 5978
```

Hands on

- Available components and how can we use them

Thanks for listening!
Bye

Check out https://skillsmatter.com/skillscasts/10466-deep-dive-on-kubernetes-networking