Kubernetes Deep Dive

O QUE É CLOUD COMPUTING?

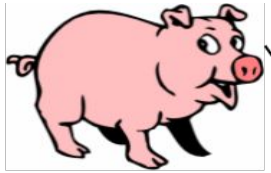
O que é Cloud afinal?

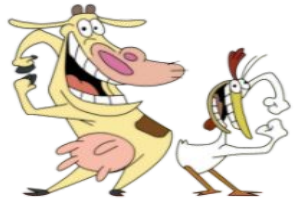
- Eu não sei onde meu recurso está
- Eu não sei o que roda por baixo para o meu recurso estar lá
- Eu quero pagar por uso
- Alguém orquestrou aquele recurso

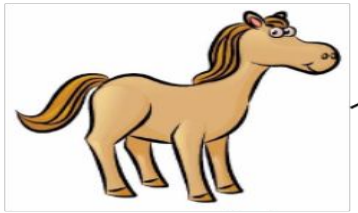
O que é Cloud afinal?

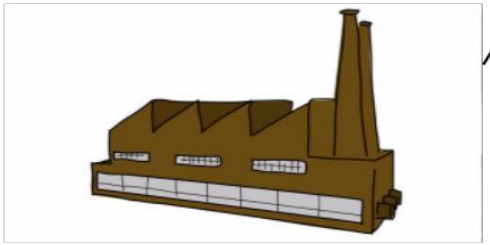
- Eu não sei onde meu recurso está
- Eu não sei o que roda por baixo para o meu recurso estar lá
- Eu quero pagar por uso
- **Alguém orquestrou aquele recurso**

Cloud é uma fábrica de salsicha









E O QUE É O KUBERNETES?

Kubernetes is an open-source system for **automating** deployment, **scaling**, and management of **containerized** applications

https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Tim Hockin
@thockin

Em resposta a @JK_Dynamic_D e @kelseyhightower

Anyone who calls k8s a PaaS is not paying attention.

Joe Beda
@jbeda

First off: Kubernetes *is* a complex system. It does a lot and brings new abstractions. Those abstractions aren't always justified for all problems. I'm sure that there are plenty of people using Kubernetes that could get by with something simpler. /3

Componentes e objetos do Kubernetes

API Server

Scheduler

Controller Manager

Kubelet

Kube Proxy

POD

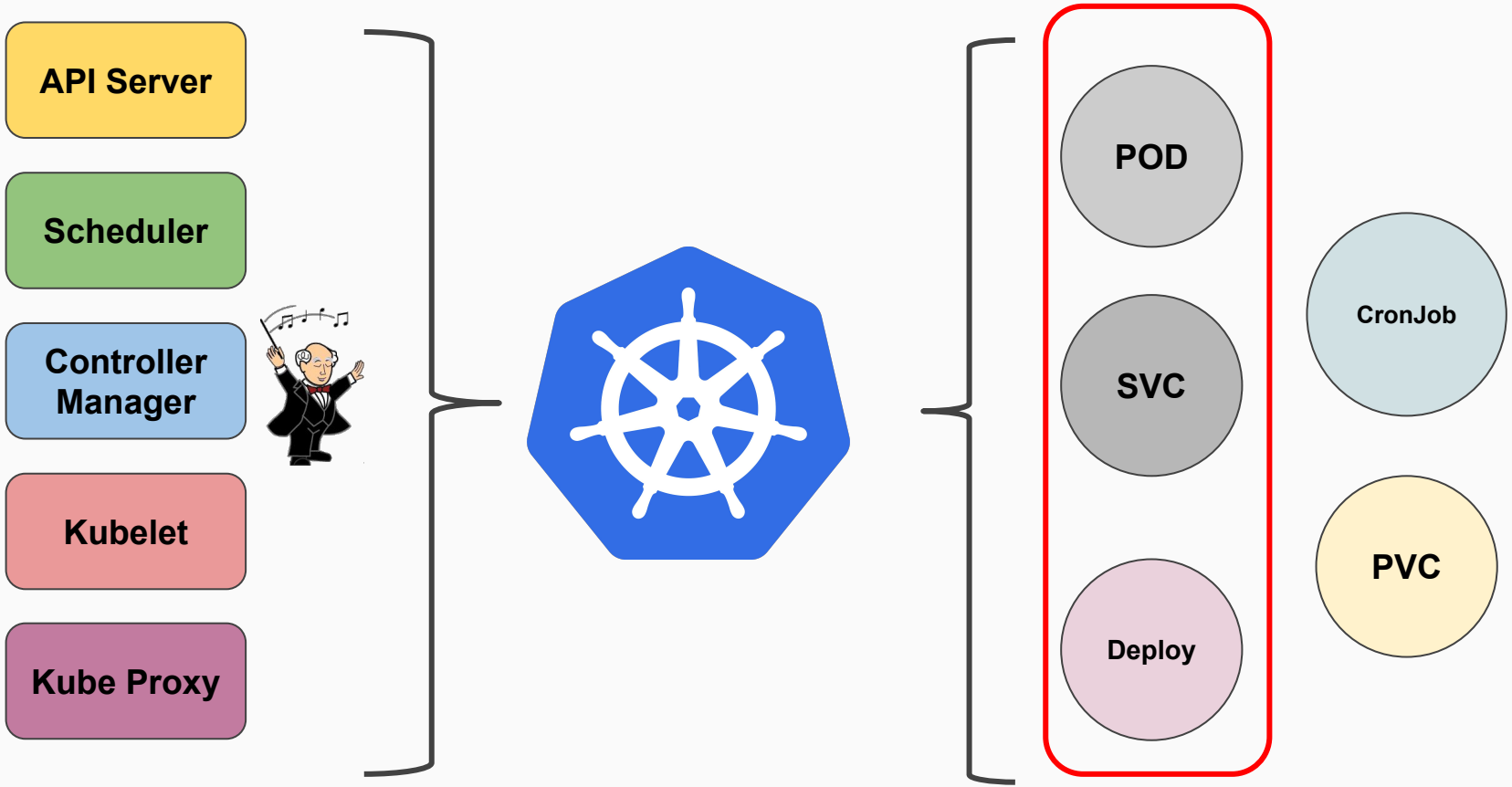
SVC

Deploy

CronJob

PVC

Objetos do Kubernetes

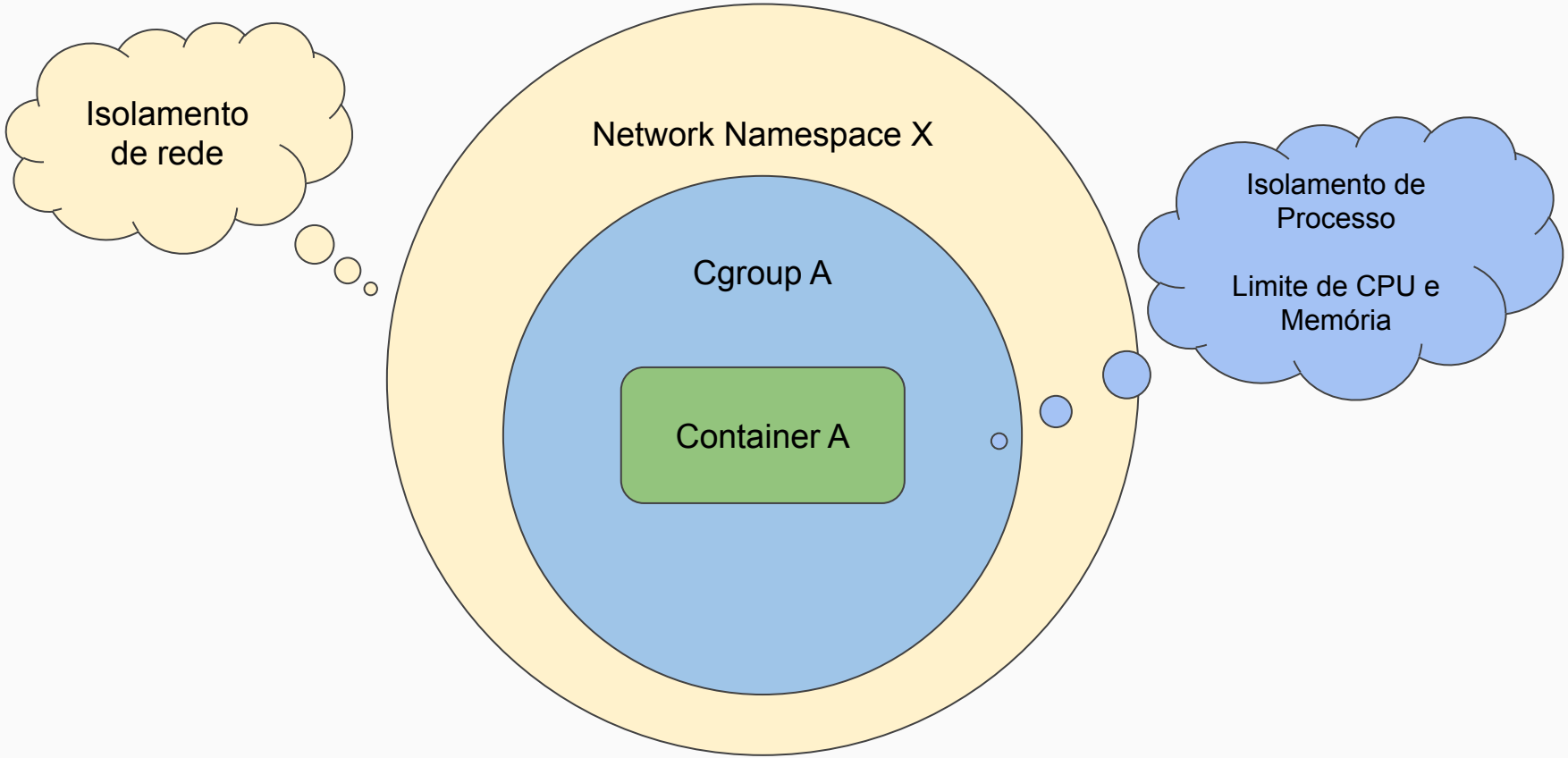


ENTÃO KUBERNETES É SÓ PARA CONTAINERS?

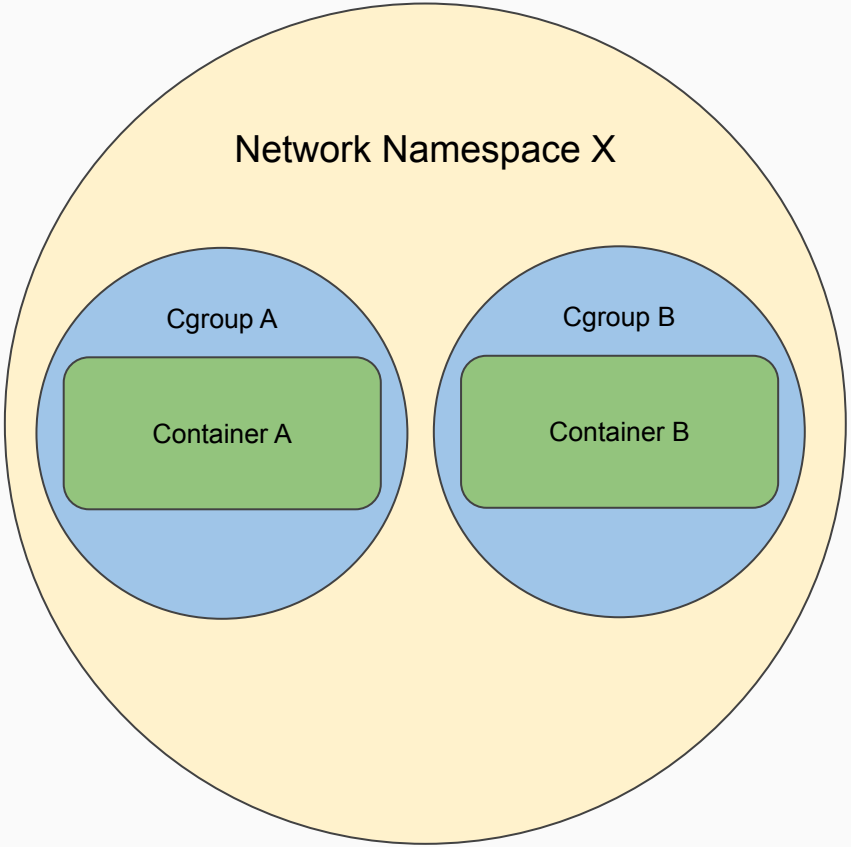
DEMO KUBE PLAYER

VOLTANDO AO TEMA ORIGINAL :P

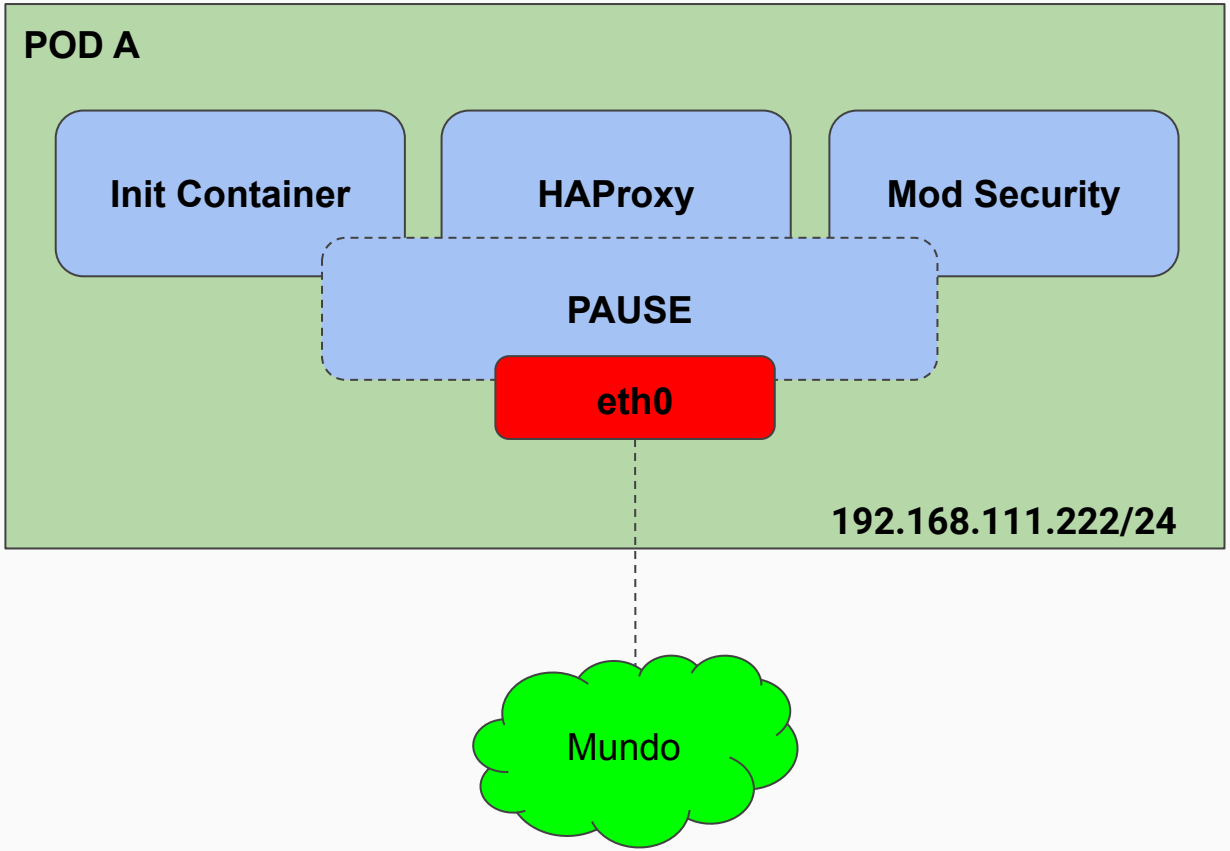
Anatomia de um Container



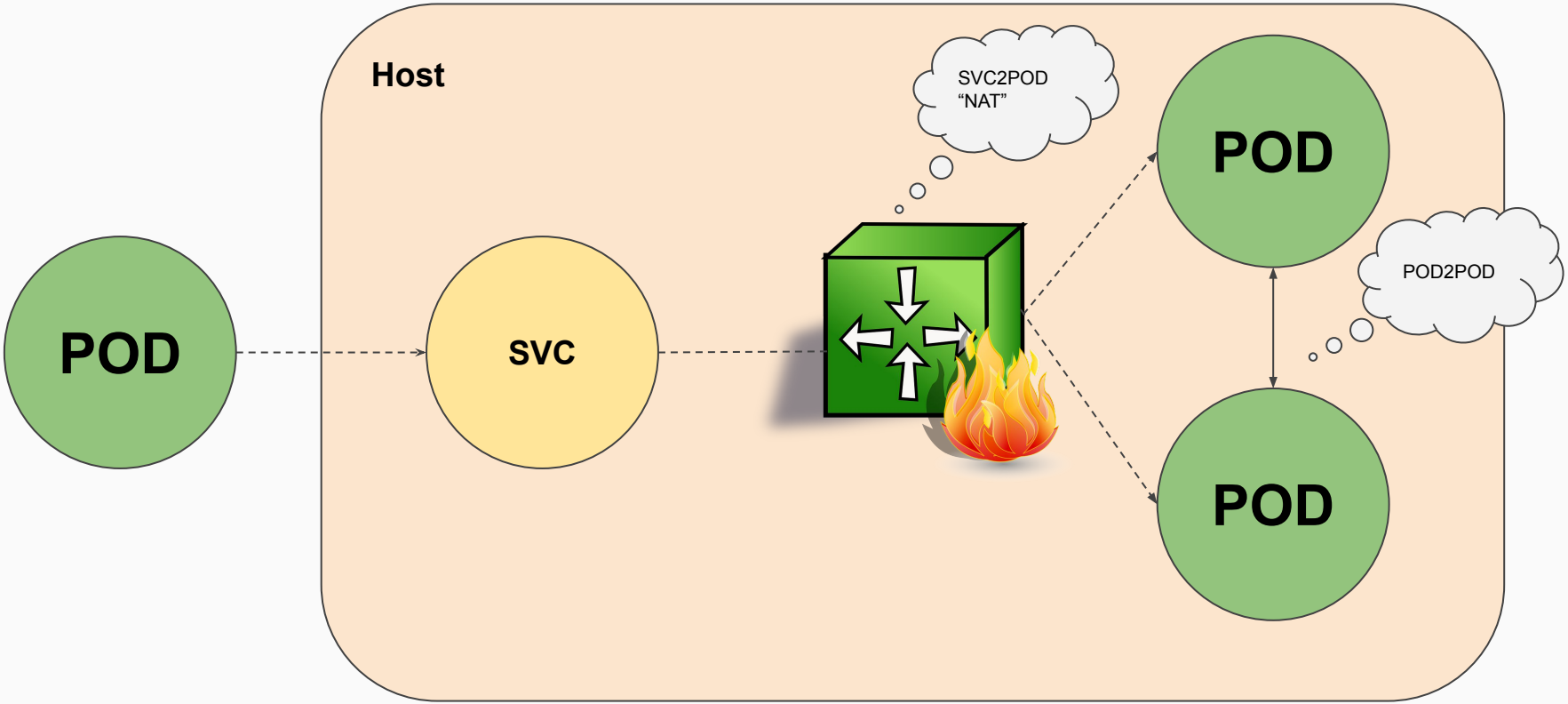
Anatomia de um POD



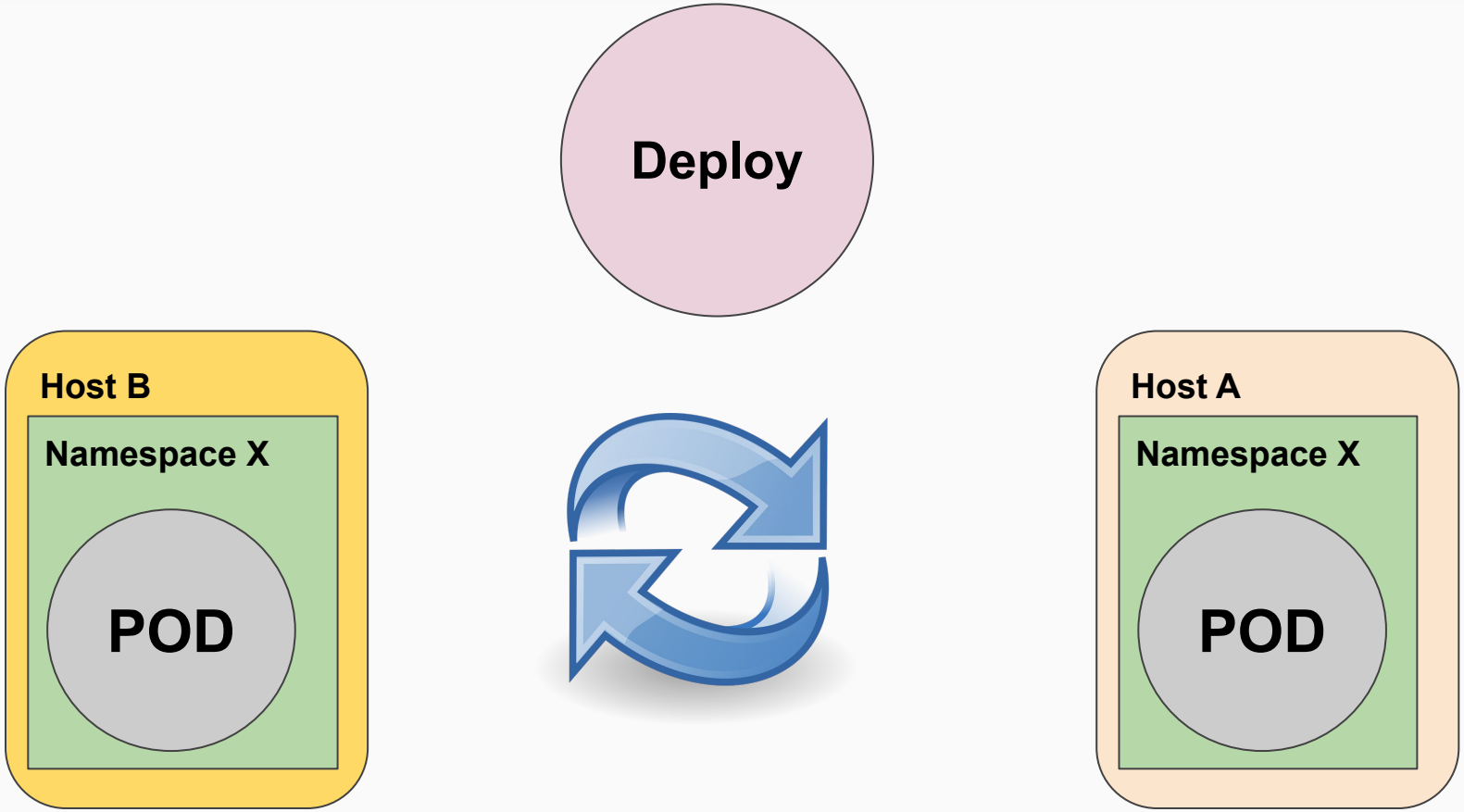
Anatomia de um POD



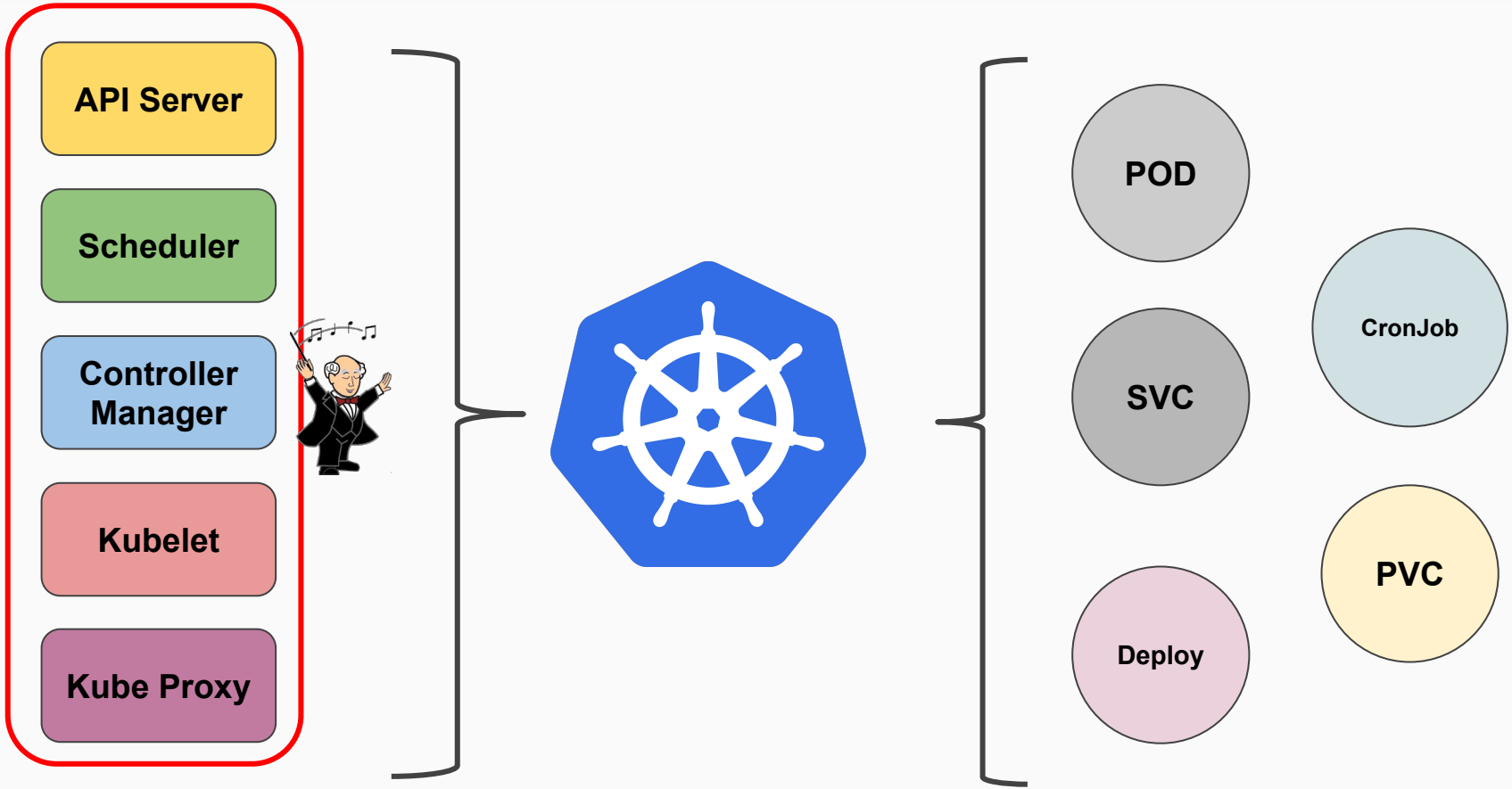
Anatomia de um Service



Anatomia de um Deployment



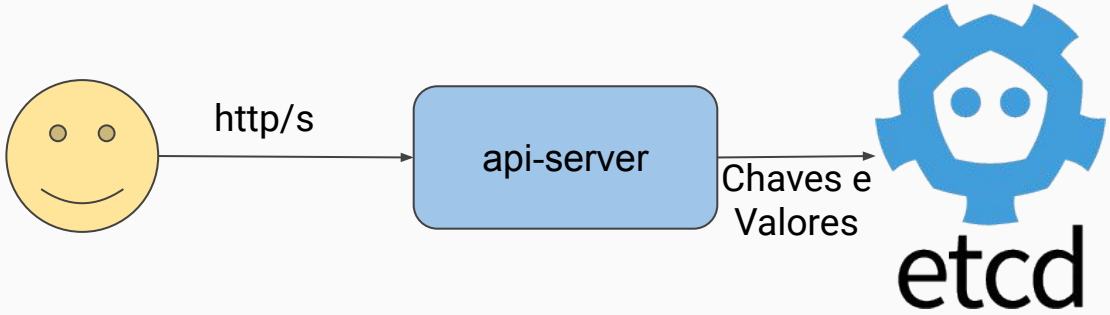
Componentes do Kubernetes



COMO ENTENDER PARA QUE SERVE CADA UM? TIRANDO ELES DO AR!!!

APIServer + ETCD

- ETCD: Store K/V - Persistência de objetos do Kubernetes
- APIServer - Apenas um repositório de objetos!
  - Autenticação
  - RBAC
  - Watch
  - REST
  - Extensível



API

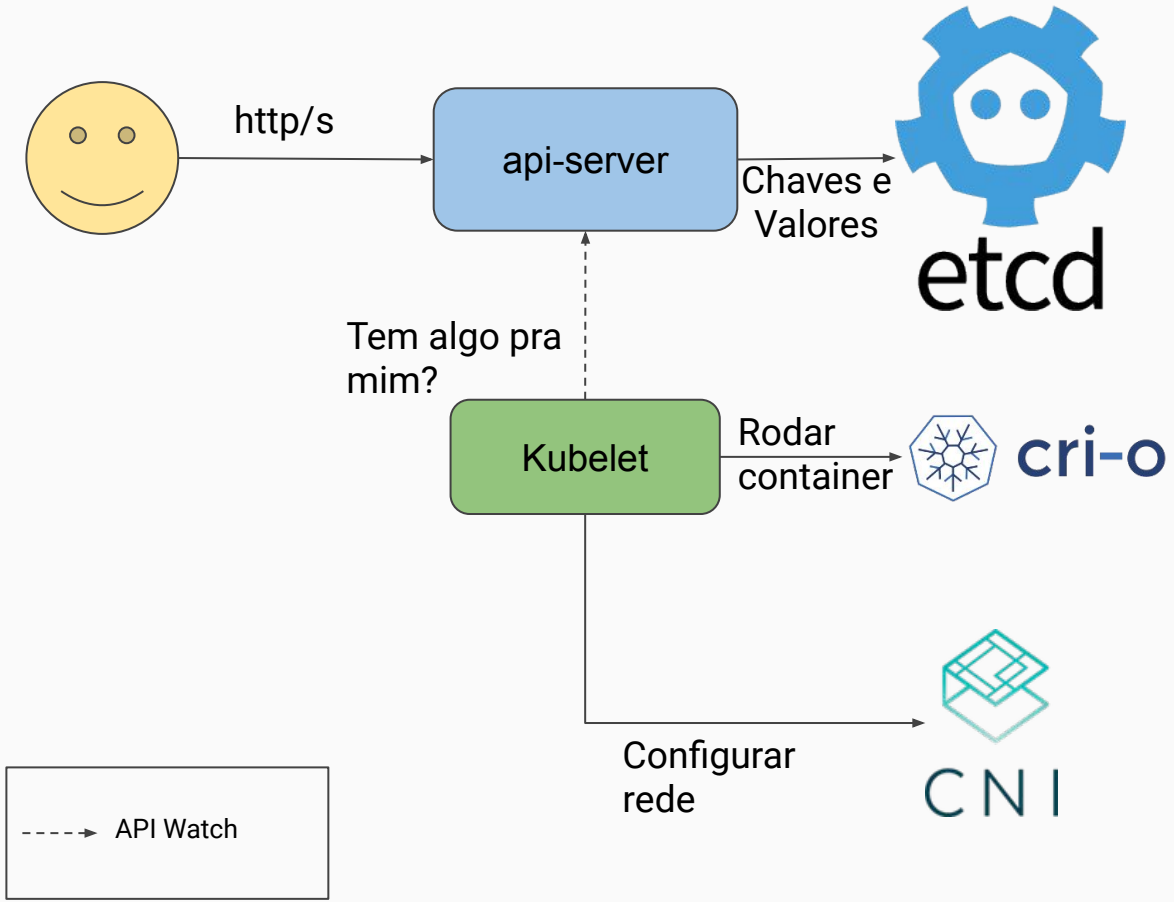
```
apiVersion: estaleiro.io/v1
kind: Music
metadata:
    namespace: default
    name: mario
spec:
    music: cm-mario
```



```
$ curl http://API:8080/apis/estaleiro.io/v1/namespaces/default/musics/mario

{
    "apiVersion":"estaleiro.io/v1",
    "kind":"Music",
    "metadata":{
        "creationTimestamp":"2019-04-25T21:21:39Z",
        "generation":2,
        "name":"mario",
        "namespace":"default",
        "resourceVersion":"469",
        "selfLink":"/apis/estaleiro.io/v1/namespaces/default/musics/mario",
        "uid":"1d852d55-67a0-11e9-b82a-3c970ea1b665"},
        "spec": {
            "music":"cm-mario",
            "status":"Tocando"
    }
}
```

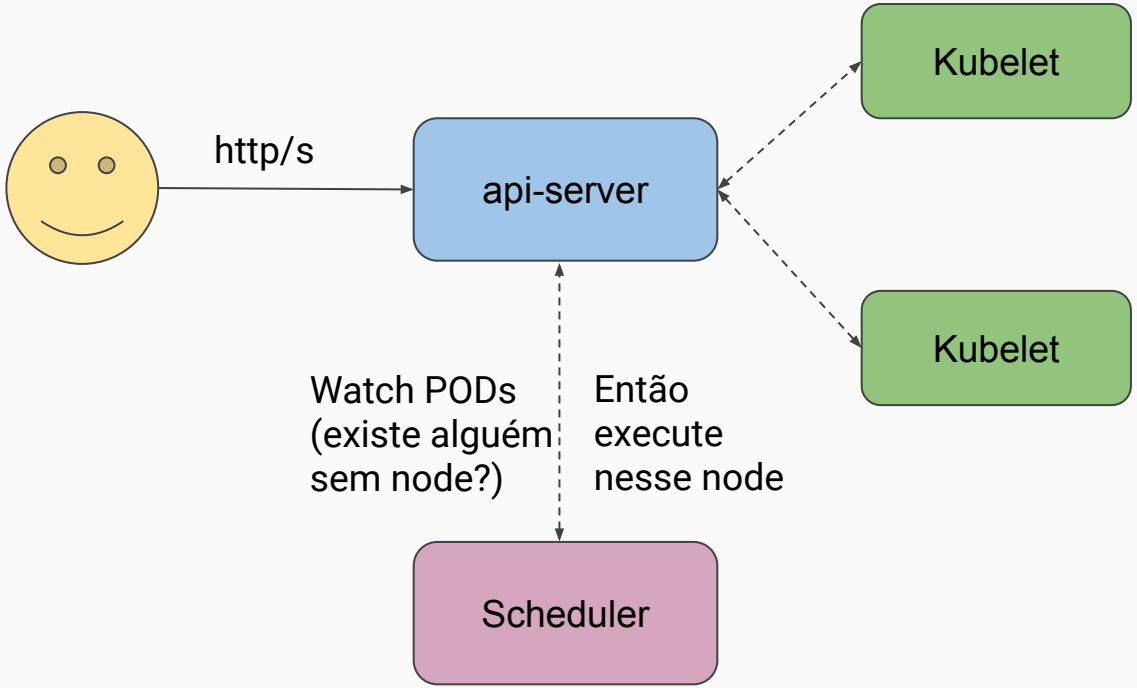
Kubelet

- Kubelet: “Existe algum workload para eu rodar?”
- Pega POD, obtém parâmetros de rede com o CNI, chama o Container Runtime e executa a carga



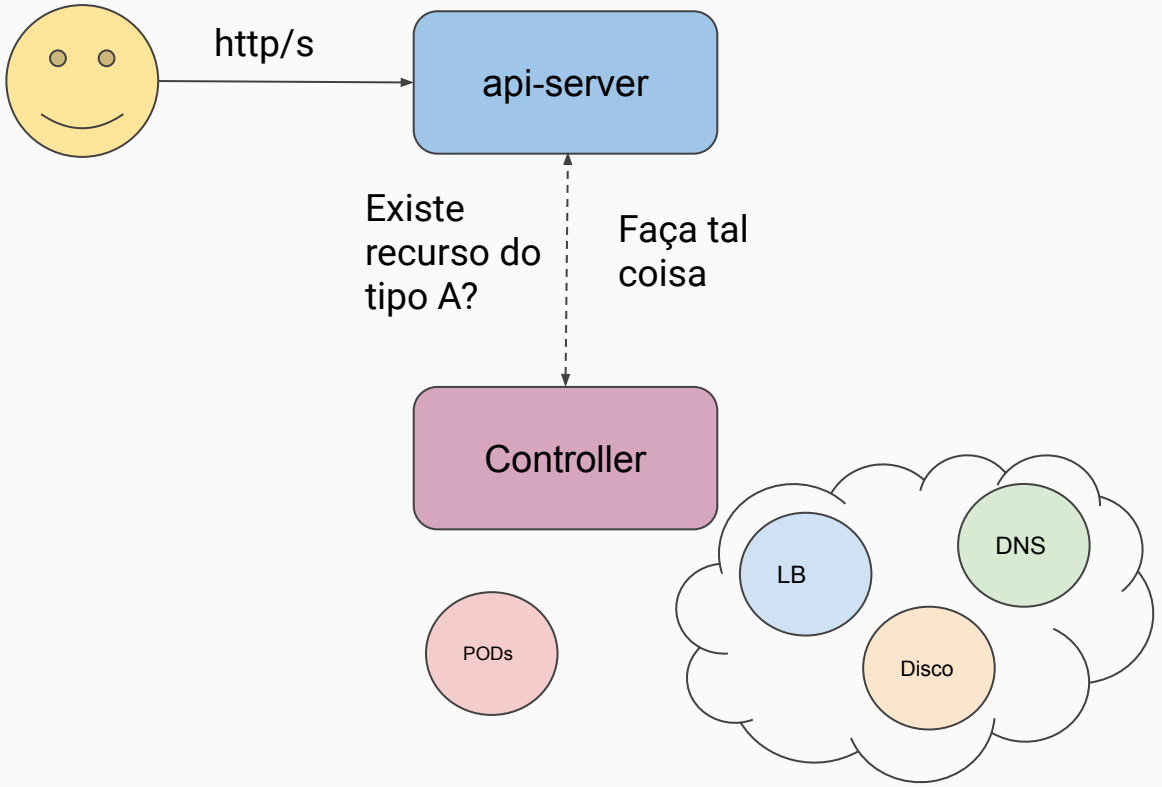
Scheduler

- Onde rodar esse Workload?
- Regras de afinidade / anti-afinidade
- Nó com disco local SSD
- Nó com GPU
- Labels, requisitos



Controllers

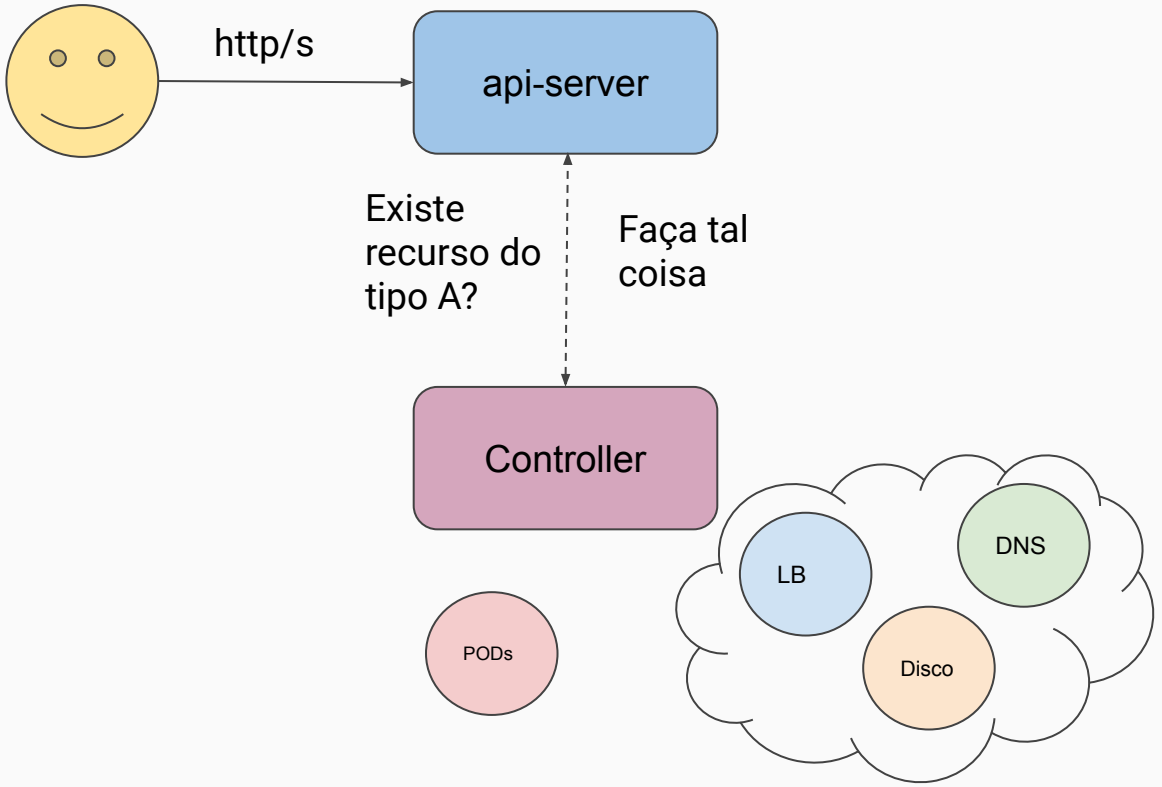
- Controllers controlam as coisas




Ora Ora, temos um
Xeroqui Homes aqui

Controllers

- Controllers controlam as coisas
- 1 Deploy, N PODs
- 1 Service com Type LoadBalancer, 1 novo ELB na Amazon
- 1 PVC, 1 novo volume no Cloud Provider



There is no cloud
It’s just someone else’s computer

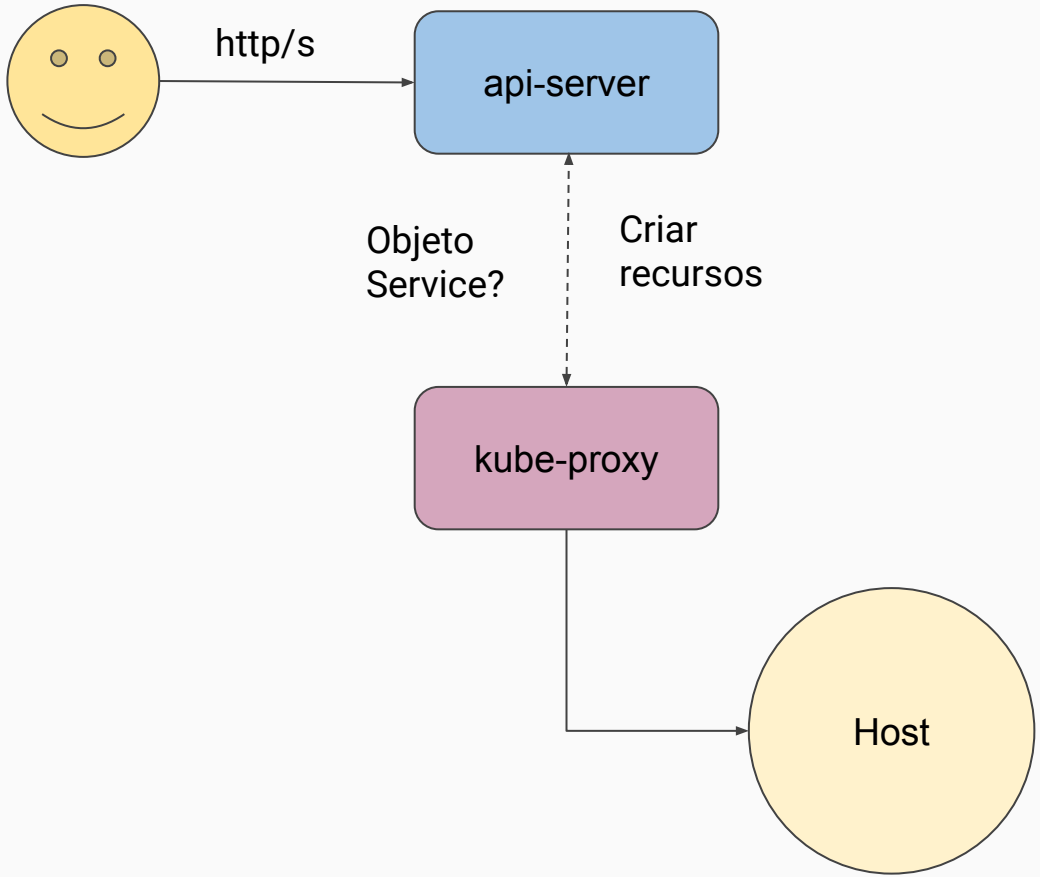
https://visualpharm.com/free-icons/sad%20cloud-595b40b85ba036ed117dc41c

Sem Controller

- Auto provisionamento do kubeadm não funciona
- Deployments não tem PODs
- Cronjobs não rodam
- E outros problemas a mais :D

Kube Proxy

- KubeProxy - Services e NodePort
  - Regras de iptables para NAT e NodePort
  - Entradas no IPVS para Service IP



COMUNICAÇÃO POD2POD

Comunicação POD2POD

$ ***kubectl*** *get pods -o wide*

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
echoserver-75685667dd-g9q8h	1/1	Running	0	9d	192.168.166.142	node1
echoserver-75685667dd-nbzwb	1/1	Running	0	9d	192.168.104.3	node2
echoserver-75685667dd-w2922	1/1	Running	0	9d	192.168.104.4	node2
echoserver-75685667dd-w4hzc	1/1	Running	0	9d	192.168.166.141	node1

==========

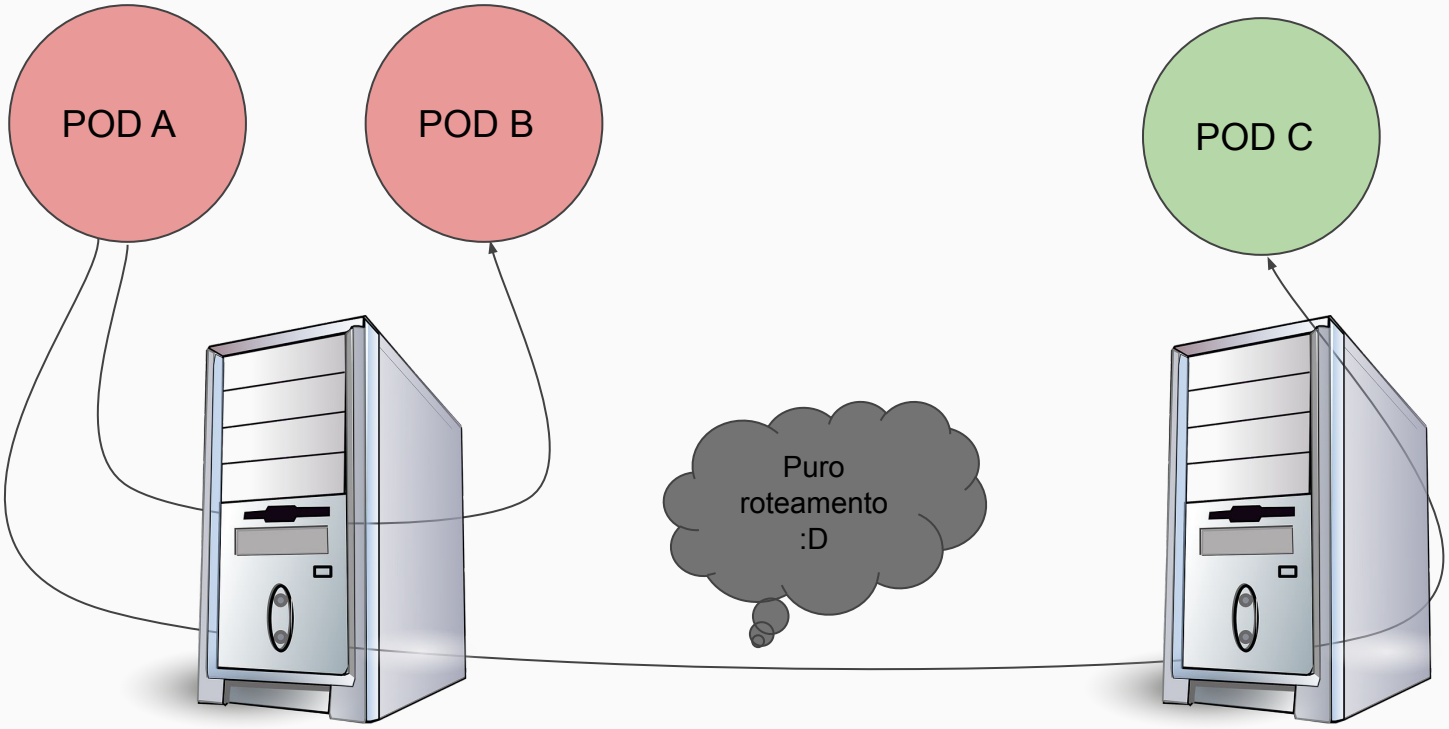
$ ***kubectl*** *exec* ***echoserver-75685667dd-g9q8h*** *curl 192.168.**104**.3:8080*
CLIENT VALUES:
client_address=**192.168.166.142**
request_uri=http://192.168.104.3:8080/

==========

$ *kubectl exec* ***echoserver-75685667dd-g9q8h*** *curl 192.168.**166**.141:8080*
CLIENT VALUES:
client_address=192.168.166.142
request_uri=http://192.168.166.141:8080/

O QUE ACONTECEU?

POD2POD



Comunicação POD2POD

node1 (172.16.224.**51**) $ ***ip route***

default	via 172.16.224.2	dev ens192	proto static
192.168.**104**.0/26	via 172.16.224.**52**	dev tunl0	proto bird onlink
blackhole	192.168.166.128/26		proto bird
192.168.**166**.141		dev cali91e73b6782f	scope link
192.168.**166**.142		dev cali55f17d7b353	scope link

node2 (172.16.224.**52**) $ ***ip route***

default	via 172.16.224.2	dev ens192	proto static
192.168.**166**.128/26	via 172.16.224.**51**	dev tunl0	proto bird onlink
blackhole	192.168.104.0/26		proto bird
192.168.**104**.3		dev calif16747aea13	scope link
192.168.**104**.4		dev calia8f2ba1d023	scope link

COMUNICAÇÃO SVC2POD

Comunicação SVC2POD

$ ***kubectl*** *get pods -o wide*

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
echoserver-75685667dd-g9q8h	1/1	Running	0	9d	192.168.166.142	node1
echoserver-75685667dd-nbzwb	1/1	Running	0	9d	192.168.104.3	node2
echoserver-75685667dd-w2922	1/1	Running	0	9d	192.168.104.4	node2
echoserver-75685667dd-w4hzc	1/1	Running	0	9d	192.168.166.141	node1

==========

$ ***kubectl*** *get svc*

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
echoserver	ClusterIP	**10.99.153.85**	<none>	8080/TCP	6s

==========

$ *kubectl exec -it* ***echoserver-75685667dd-g9q8h*** *curl -v* ***echoserver****:8080*
** Connected to* ***echoserver (10.99.153.85)*** *port 8080 (#0)*
[...]
CLIENT VALUES:
client_address=192.168.166.142
request_uri=http://echoserver:8080/

Comunicação SVC2POD

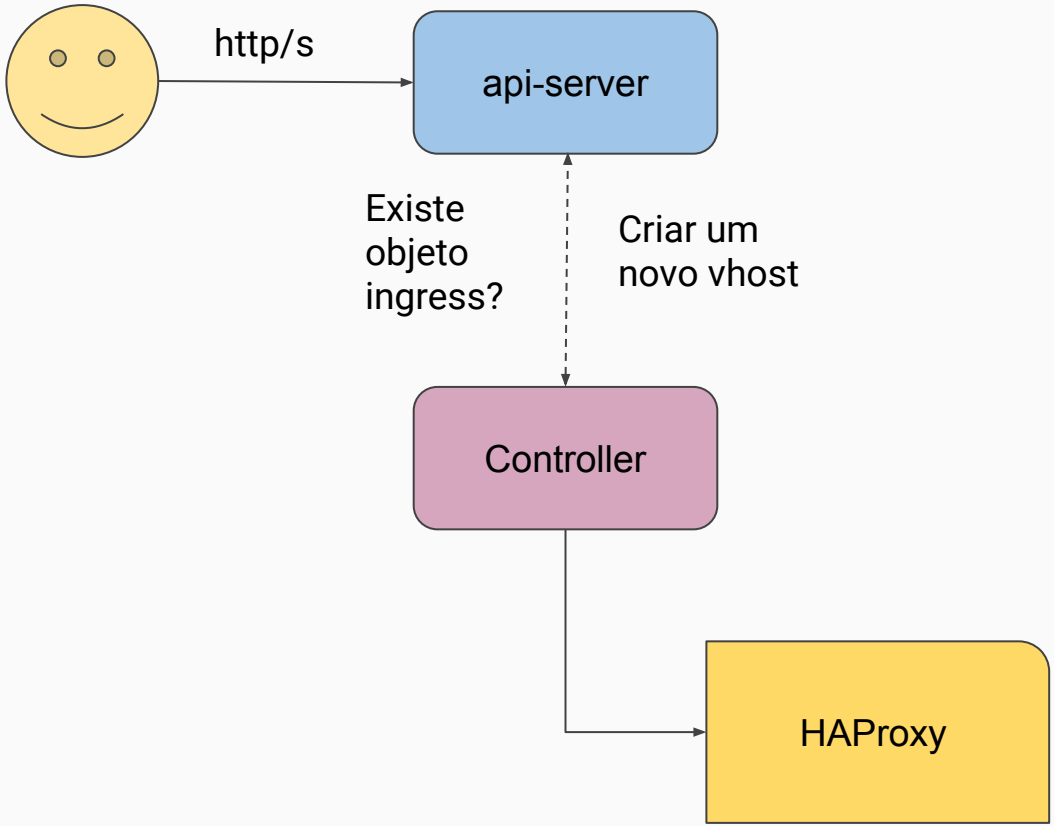
node2 (172.16.224.**52**) $ ***ipvsadm -L -n***

```
Prot        LocalAddress:Port      Scheduler   Flags
      ->    RemoteAddress:Port                 Forward     Weight      ActiveConn      InActConn

TCP         10.99.153.85:8080      rr
      ->    192.168.104.5:8080                 Masq        1           0               0
      ->    192.168.104.6:8080                 Masq        1           0               0
      ->    192.168.166.141:8080               Masq        1           0               0
      ->    192.168.166.142:8080               Masq        1           0               0
```

EU VIM AQUI SÓ PELO INGRESS!!!

Controllers

- Ingress CONTROLLER
- Entra objeto de Ingress, sai configuração de servidor Web
  - HAProxy
  - NGINX
  - Contour
  - Istio
  - Gloo
  - Etc etc etc
- Assim como meu Controller de musica :)



Ingress

$ ***kubectl*** *describe ingress echoserver*
Name: echoserver
Address:
Default backend: default-http-backend:80 (<none>)
Rules:

Host	Path	Backends
echoserver.test		
	/	echoserver:8080 (192.168.104.5:8080,192.168.104.6:8080,192.168.166.141:8080 + 1 more...)

==========

$ curl -H "Host: **echoserver.test**" http://172.16.224.51 ⇐ O ingress aqui está exposto com um NodePort na porta 80 do “node1”

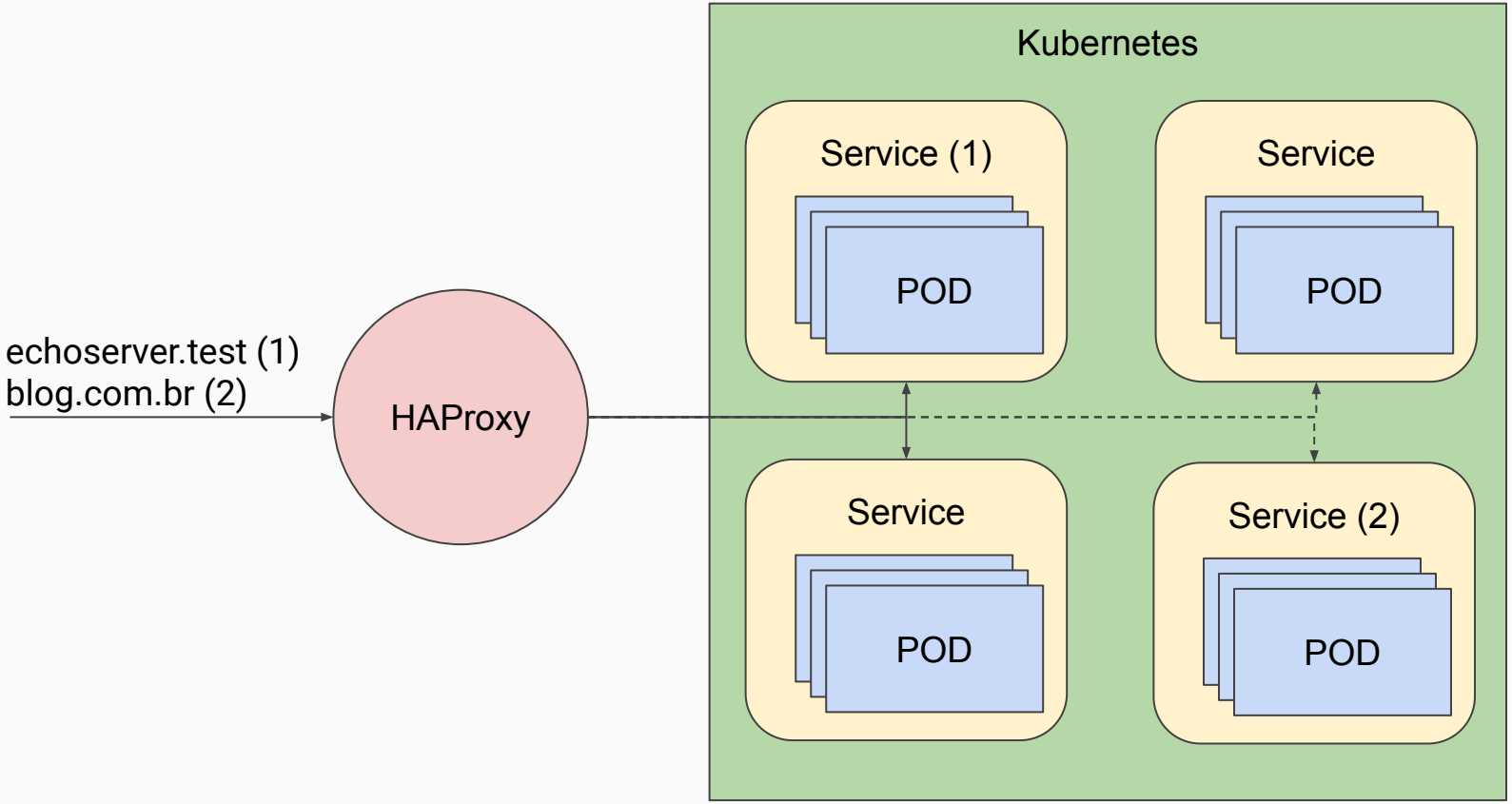
CLIENT VALUES:
client_address=192.168.166.128
request_uri=http://echoserver.test:8080/

O QUE ACONTECEU?

Enquanto isso, dentro do Container do HAProxy

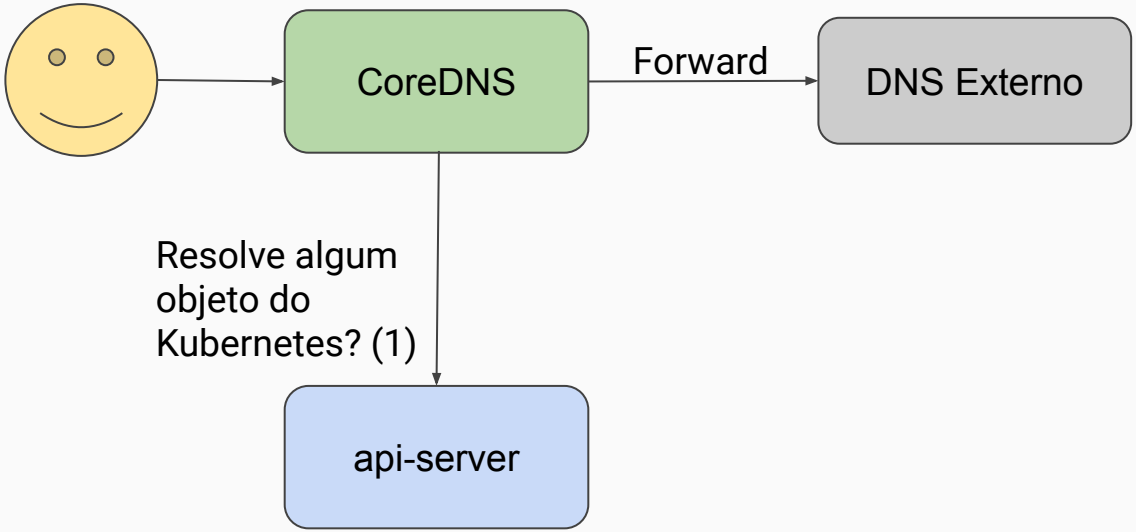
```
$ kubectl  exec -n ingress-controller -it haproxy-ingress-7cphn cat /etc/haproxy/haproxy.cfg

backend default-echoserver-8080
    mode http
    balance roundrobin
    server 192.168.104.5:8080 192.168.104.5:8080 weight 1 check inter 2s
    server 192.168.166.141:8080 192.168.166.141:8080 weight 1 check inter 2s

[...]
frontend httpfront-shared-frontend
    bind *:80
    mode http
    acl host-echoserver.test var(txn.hdr_host) -i echoserver.test echoserver.test:80 echoserver.test:443
    use_backend default-echoserver-8080 if host-echoserver.test

================================================================================

$ kubectl  exec -n ingress-controller -it haproxy-ingress-7cphn ps

    7              /haproxy-ingress-controller --default-backend-service=ingr
   32              haproxy -f /etc/haproxy/haproxy.cfg -p /var/run/haproxy.pi
```

Ingress Controller



CoreDNS

- Resolve nome para os PODs dentro do Cluster
- “svc.cluster.local” -> Qual o IP do Service para esse nome?
- Menos IPs, mais nomes (desacoplar aplicações)
- O sufixo “cluster.local” pode ser alterado
- CoreDNS não serve apenas para Kubernetes :)
- CoreDNS não precisa rodar dentro do cluster (mas é bom os PODs conseguirem chegar nele!)



CoreDNS

```
$ kubectl get svc -n default |grep echoserver

echoserver        ClusterIP          10.99.153.85     <none>              8080/TCP   5h23m

=============================================================================

$ dig +noall +answer @127.0.0.1 echoserver.default.svc.cluster.local
echoserver.default.svc.cluster.local. 5 IN A      10.99.153.85

=============================================================================

$ cat Corefile

.:53 {
    kubernetes cluster.local {
            upstream
      }
      forward . 8.8.8.8 # Para quem é feito o Forward em caso de esse DNS não ser autoridade
      cache 30
}
```

Referências

- https://github.com/jamiehannaford/what-happens-when-k8s
- https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/#additional-controllers
- https://jvns.ca/blog/2016/10/10/what-even-is-a-container/
- https://github.com/jcmoraisjr/haproxy-ingress

Obrigado!

Ricardo Pchevuzinske Katz
SERPRO

Emails:
ricardo.katz@serpro.gov.br
ricardo.katz@gmail.com

Twitter: @rpkatz

Telegram: t.me/kubernetesbr

estaleiro