



sfl[phone]

The free software  
enterprise-class  
softphone for GNU/Linux



debian



kubuntu



openSUSE



# Não profissional





# OPOVO



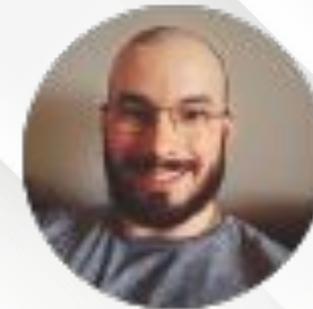
**Novell.**  
PartnerNet®  
DATA CENTER  
SPECIALIST



Certified  
LINUX  
ADMINISTRATOR  
**Novell.**



# Marcelo Cavalcante Rocha



 marcelo@marcelocavalcante.net

[www.marcelocavalcante.net](http://www.marcelocavalcante.net)

 [www.linkedin.com/in/marcelocrocha](http://www.linkedin.com/in/marcelocrocha)

 marcelokalib

 marcelocavalcanterocha

 [facebook.com/marcelocavalcante](https://facebook.com/marcelocavalcante)

**kalib @ freenode.org**





# Kubernetes

# Conceitos - Core

## Arquitetura do Cluster

Um pouco sobre a arquitetura de um cluster Kubernetes e seus componentes básicos e fundamentais

## Serviços e outras primitivas de Redes

Introdução a como todos esses componentes se comunicam bem como sobre a estrutura de redes do Kubernetes

## APIs Primitivas

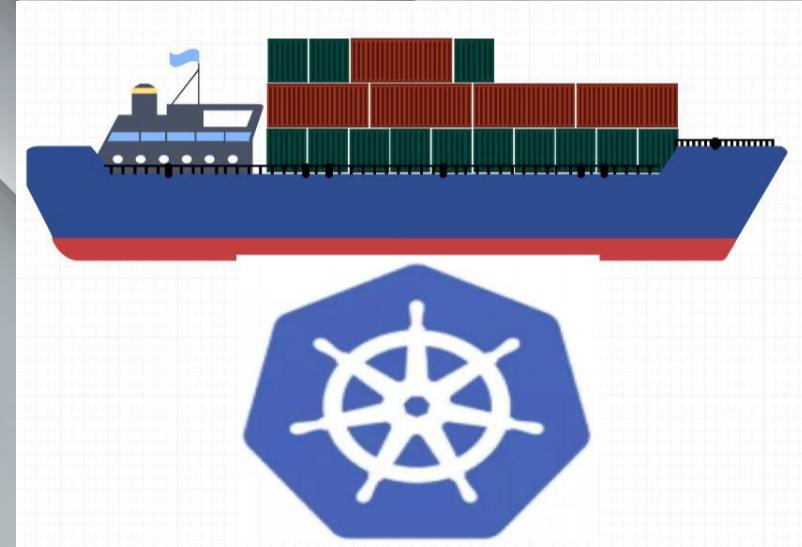
Introdução as APIs fundamentais do Kubernetes como pods, replicaset, deployments, services...

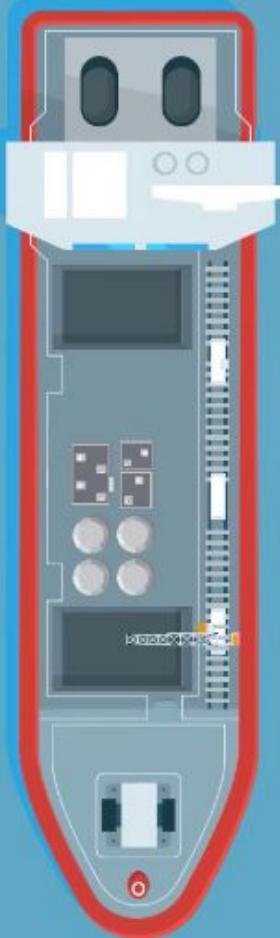




# Arquitetura do Cluster

- △ Arquitetura
- △ ETCD
- △ Kube-API Server
- △ Controller Managers
- △ Kube Scheduler
- △ Kubelet
- △ Kube Proxy





## MASTER NODE

Gerencia, Planeja, Agenda e Monitora os nodes

## WORKER NODES

Hospeda aplicações de containers

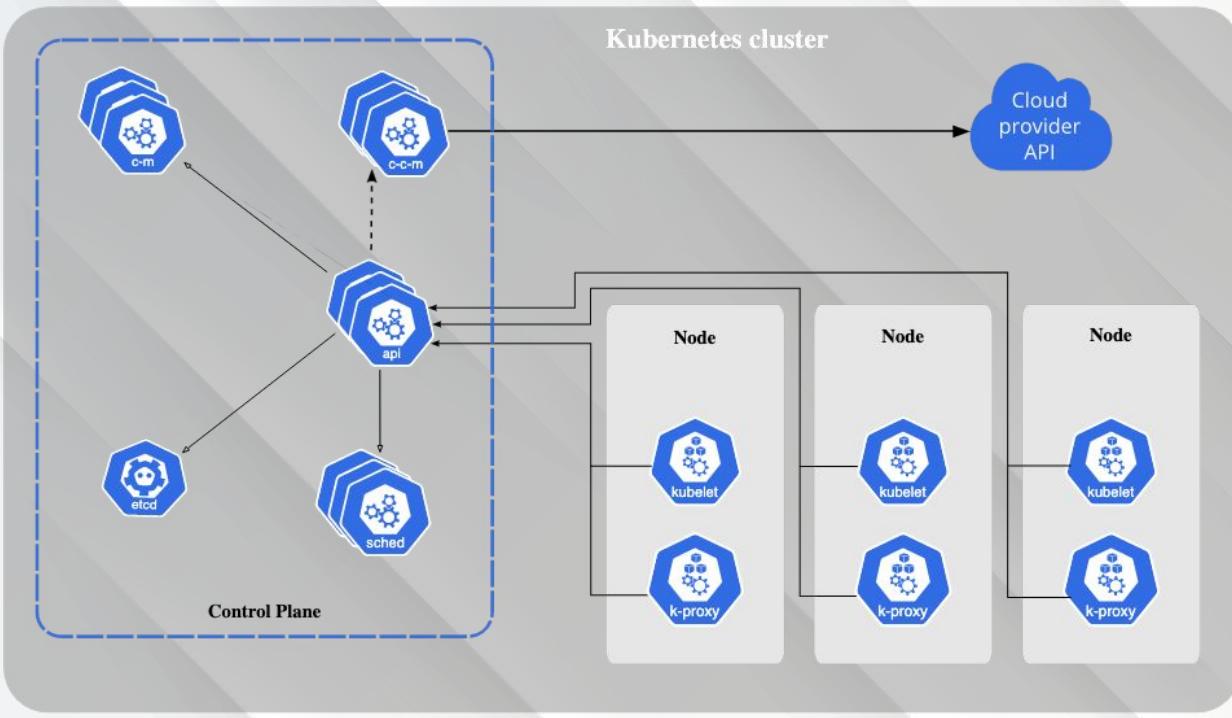
Um **cluster Kubernetes** é composto por **NODES** (ou nós), responsáveis por hospedar aplicações em formato de containers.

São como **navios de carga**, que apenas recebem os containers e os transportam. Na arquitetura de um cluster Kubernetes, chamamos estes nodes de **WORKER NODES**.

Os **navios de controle** são chamados de **MASTER NODE**. Eles são os responsáveis por monitorar e gerenciar o trabalho dos worker nodes: alocar containers, registrar informações, escolher em qual worker node vai cada container, etc.

<https://kubernetes.io/docs/concepts/overview/components/>

# Control Plane Components





**Master**  
Gerencia, Planeja, Agenda e  
Monitora os nodes

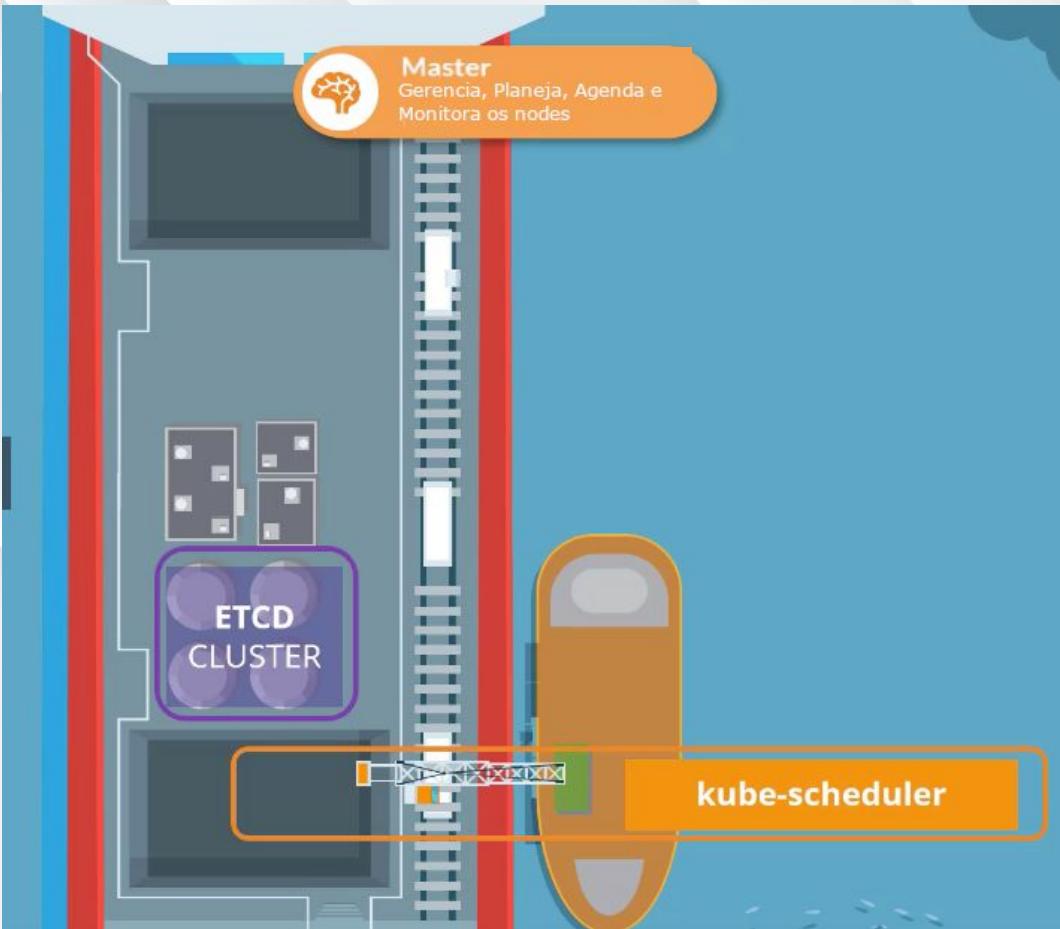


**ETCD CLUSTER**



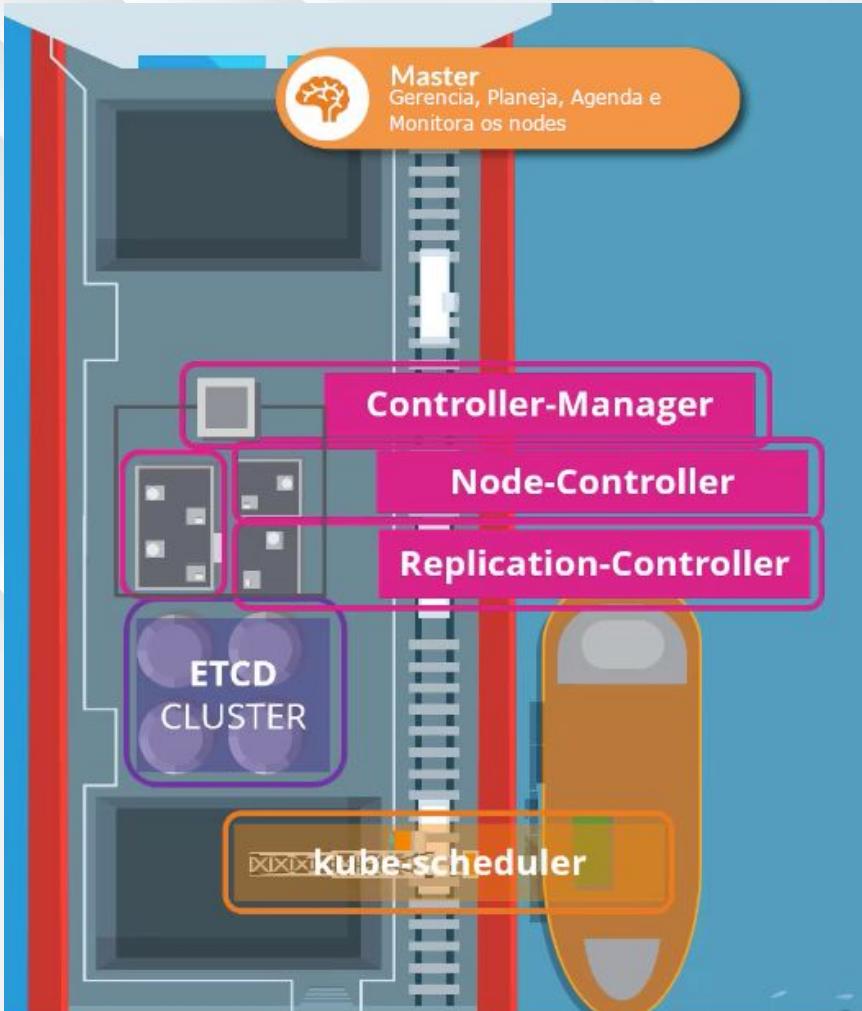
O Kubernetes utiliza-se do **ETCD** para armazenar todas as informações sobre os containers e nodes existentes no cluster.

É uma popular ferramenta de bancos de dados que armazena informações no formato **chave:valor**.



O componente **Scheduler** ou **Kube-Scheduler** é responsável por alocar os containers nos worker nodes apropriados de acordo com uma série de fatores como capacidade do worker node, tamanho do container, destino, tipo de aplicação, exigências e restrições do container ou worker node, etc.

Ele é basicamente o guindaste ou escritório de guindaste, que coloca os containers em cada navio de carga.



**Controller-Managers** são responsáveis por controles específicos no cluster.

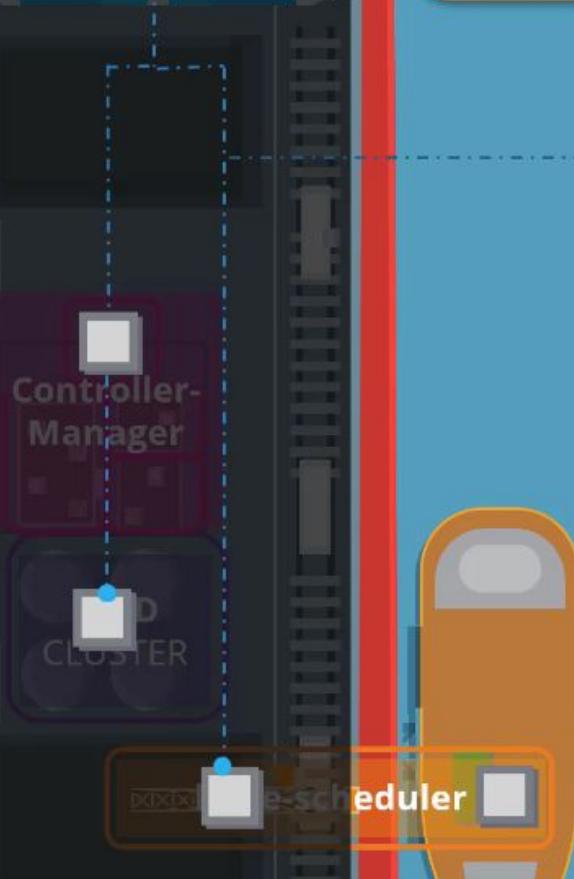
Exemplos de controllers são:

- **Node-Controller**: responsável pelos **nodes** em geral
- **Replication-Controller**: responsável pelas **réplicas**, ou seja o número de containers rodando em um determinado grupo de réplicas

## kube-apiserver



**Master**  
Gerencia, Planeja, Agenda e  
Monitora os nodes



## KUBE-APISERVER

O gerente entre os gerentes.

Responsável por **orquestrar** TODAS as operações no cluster.

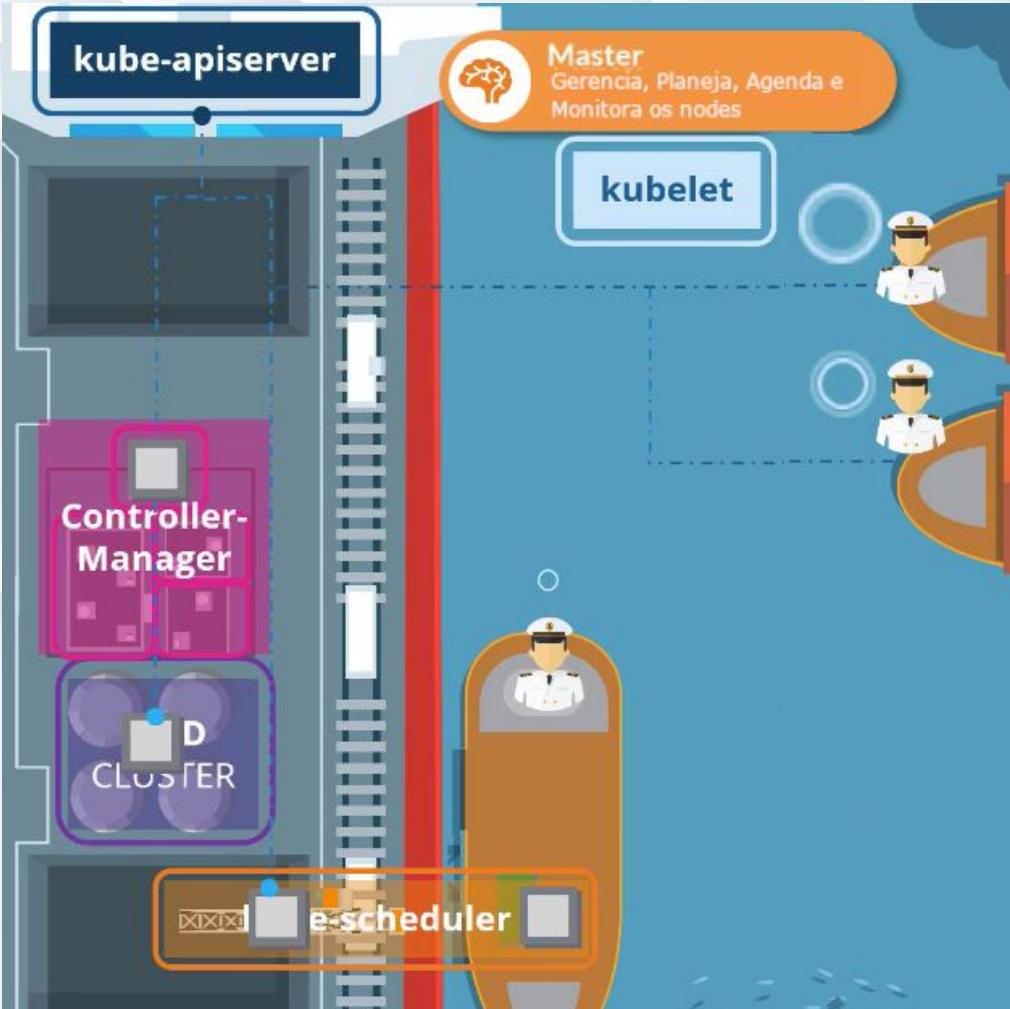
Expõe as **APIs do kubernetes** para que sejam utilizadas por usuários externos durante suas requisições e operações no cluster.

Assim como os demais componentes do Kubernetes, o **kube-apiserver** nada mais é do que um **container** no cluster.

# CONTAINERS, CONTAINERS

# EVERYWHERE

memegenerator.net

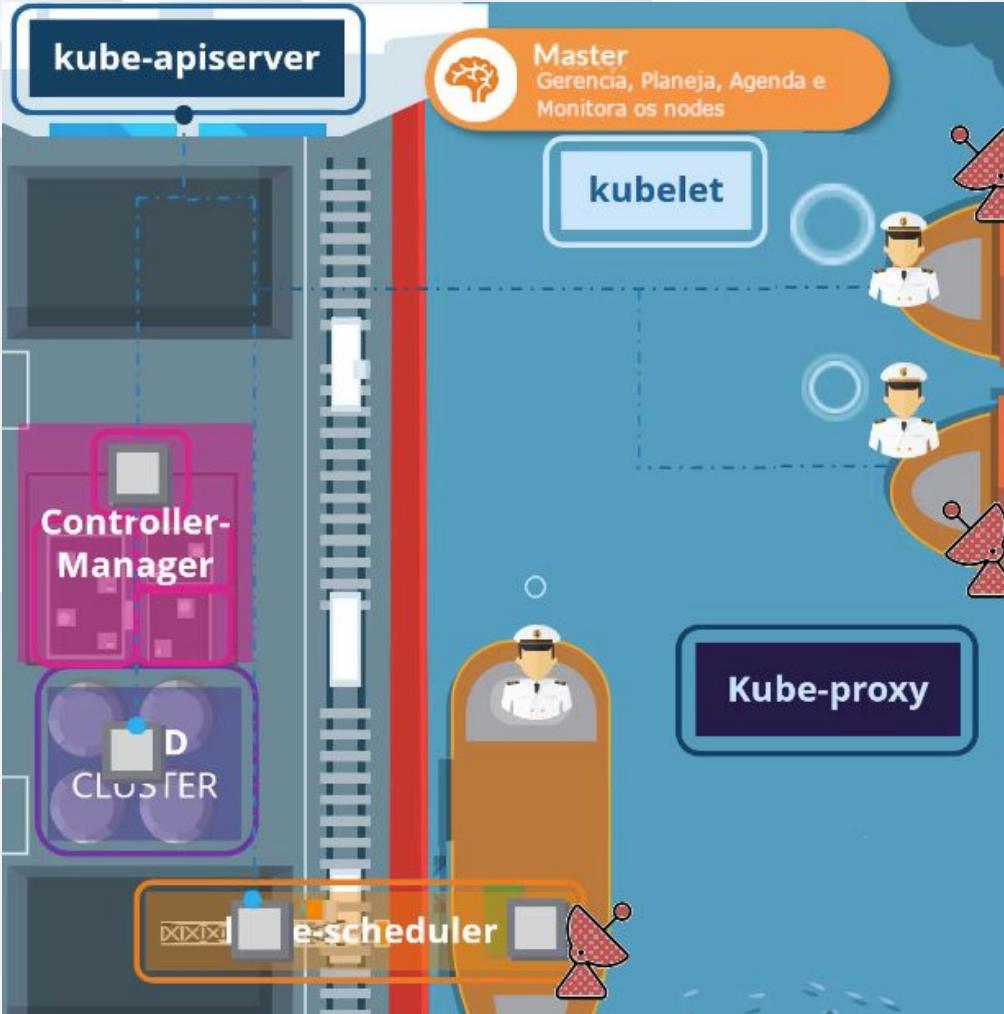


## Kubelet

Todo navio precisa de um capitão e este capitão é o **Kubelet**.

Responsável pela **comunicação** entre o **worker node** e o **master**. Se **reporta** ao kube-apiserver constantemente passando informações sobre o **status** do seu node bem como dos containers que ali estão.

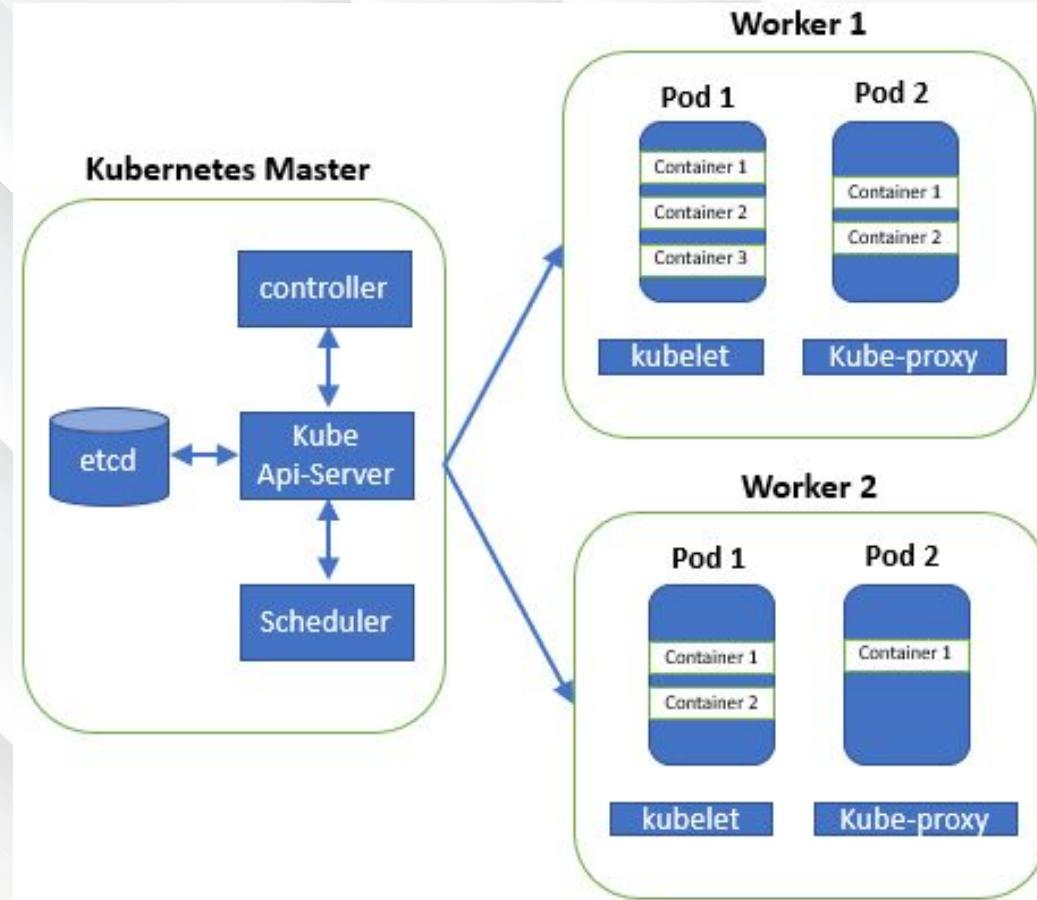
Ele é quem recebe as **instruções** do kube-apiserver sobre o que deve ser feito no seu node, containers que devem ser carregados lá, etc.



## Kube-proxy

O kube-proxy é o serviço responsável por garantir a comunicação entre os containers/aplicações existentes no cluster de nodes seguindo uma série de regras para isso.

## Resumão da arquitetura básica do Kubernetes





- O que é o ETCD?
- O que é um dado de formato Chave-Valor?
- Como começar?
- Como operar o ETCD?
- O que é um sistema distribuído?
- Como o ETCD opera?
- Protocolo RAFT
- Melhores práticas em números de nodes

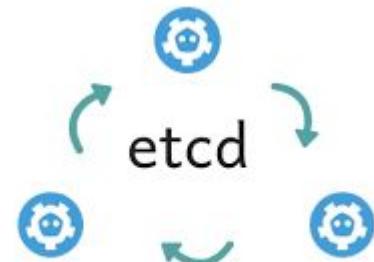
- Armazenamento de dados no formato chave-valor confiável, distribuído, e que é simples, seguro e rápido.

### Key-Value (Chave-Valor)

Chave	Valor
Nome	Clevin
Idade	78
Localização	Paris

### Tabular - Bancos de Dados Relacionais

Nome	Idade	Localização
Berg	18	Sobral
Bruno	45	Cabrobró
Samuel	32	Salvador



- Armazenamento de dados no formato chave-valor confiável, distribuído, e que é simples, seguro e rápido.

### Key-Value (Chave-Valor)

Chave	Valor
Nome	Clevin
Idade	78
Localização	Paris

Put Nome "Clevin"

Get Nome

"Clevin"

### Tabular - Bancos de Dados Relacionais

Nome	Idade	Localização
Berg	18	Sobral
Bruno	45	Cabrobró
Samuel	32	Salvador





Patroni



# Instalação do ETCD

1. Download do binário

```
curl -L https://github.com/etcd-io/etcd/releases/download/v3.3.11/etcd-  
v3.3.11-linux-amd64.tar.gz -o etcd-v3.3.11-linux-amd64.tar.gz
```

2. Extração

```
tar xzvf etcd-v3.3.11-linux-amd64.tar.gz
```

3. Execute o serviço ETCD

```
./etcd
```

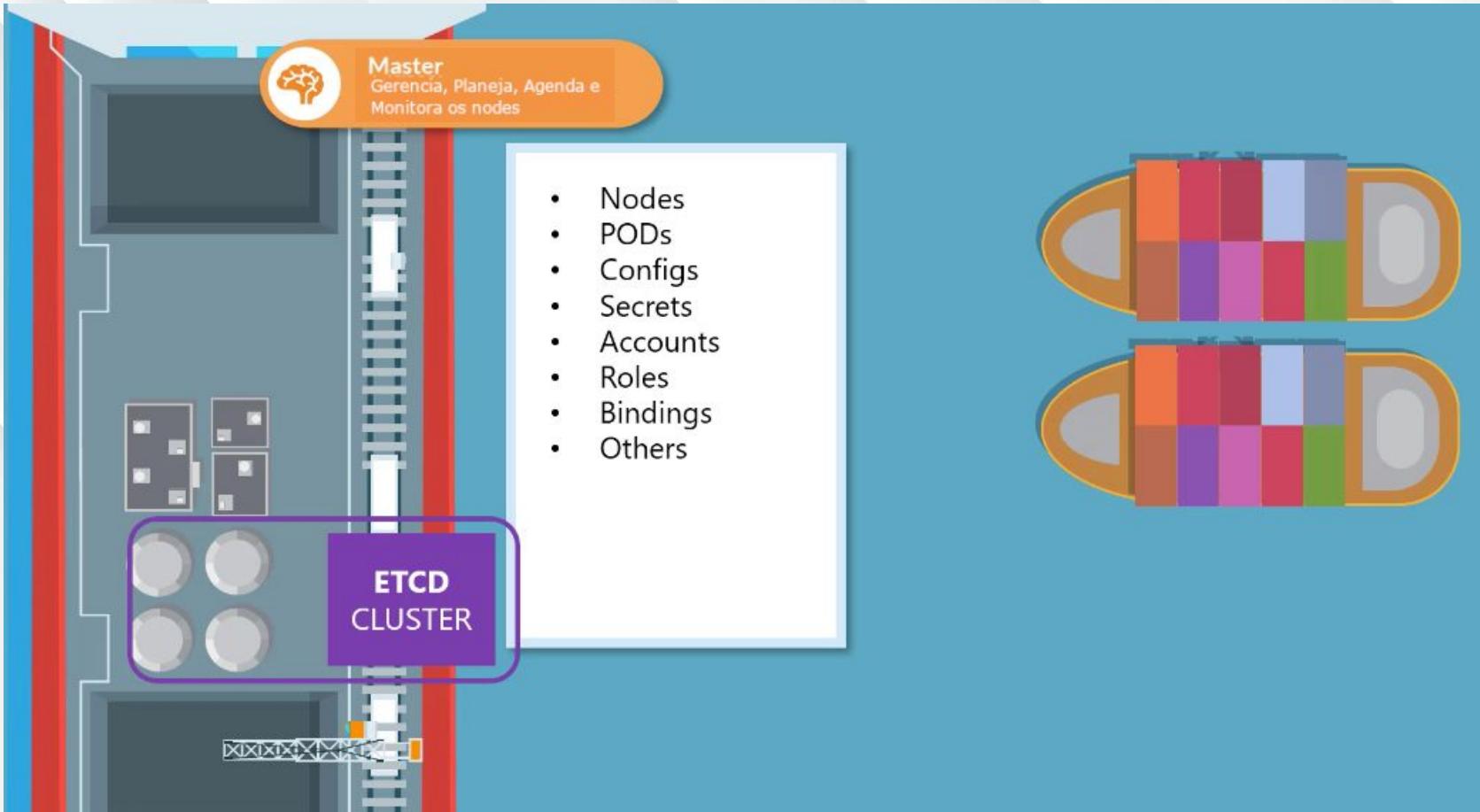
4. Para gerenciar o ETCD, utilizamos o cliente chamado etcdctl:

```
→ etcdctl put chave1 valor1
```

```
→ etcdctl get chave1  
chave1  
valor1
```



# ETCD no Kubernetes



## Kubeadm

```
kubectl get pods -n kube-system
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-78fcdf6894-prwvl	1/1	Running	0	1h
kube-system	coredns-78fcdf6894-vqd9w	1/1	Running	0	1h
kube-system	etcd-master	1/1	Running	0	1h
kube-system	kube-apiserver-master	1/1	Running	0	1h
kube-system	kube-controller-manager-master	1/1	Running	0	1h
kube-system	kube-proxy-f6k26	1/1	Running	0	1h
kube-system	kube-proxy-hnzsw	1/1	Running	0	1h
kube-system	kube-scheduler-master	1/1	Running	0	1h
kube-system	weave-net-924k8	2/2	Running	1	1h
kube-system	weave-net-hzfcz	2/2	Running	1	1h

```
* kubectl get pods -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-5644d7b6d9-2kw9p	1/1	Running	25	209d
coredns-5644d7b6d9-gdfqh	1/1	Running	25	209d
etcd-docker-desktop	1/1	Running	48	209d
kube-apiserver-docker-desktop	1/1	Running	47	209d
kube-controller-manager-docker-desktop	1/1	Running	27	209d
kube-proxy-6b9gz	1/1	Running	25	209d
kube-scheduler-docker-desktop	1/1	Running	27	209d
storage-provisioner	1/1	Running	50	209d
vpnkit-controller	1/1	Running	26	209d

## Docker

# ETCD em alta disponibilidade



```
kubectl exec etcd-master -n kube-system etcdctl get / --prefix -keys-only
/registry/apiregistration.k8s.io/apiservices/v1.
/registry/apiregistration.k8s.io/apiservices/v1.apps
/registry/apiregistration.k8s.io/apiservices/v1.authentication.k8s.io
/registry/apiregistration.k8s.io/apiservices/v1.authorization.k8s.io
/registry/apiregistration.k8s.io/apiservices/v1.autoscaling
/registry/apiregistration.k8s.io/apiservices/v1.batch
/registry/apiregistration.k8s.io/apiservices/v1.networking.k8s.io
/registry/apiregistration.k8s.io/apiservices/v1.rbac.authorization.k8s.io
/registry/apiregistration.k8s.io/apiservices/v1.storage.k8s.io
/registry/apiregistration.k8s.io/apiservices/v1beta1.admissionregistration.k8s.io
```

Registry

minions

pods

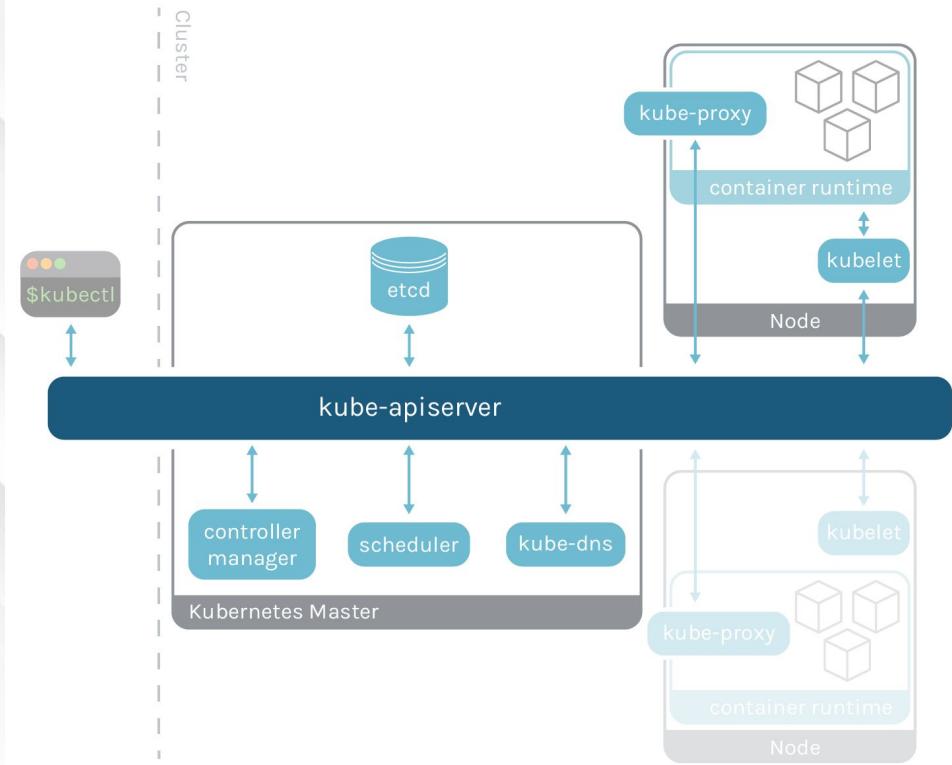
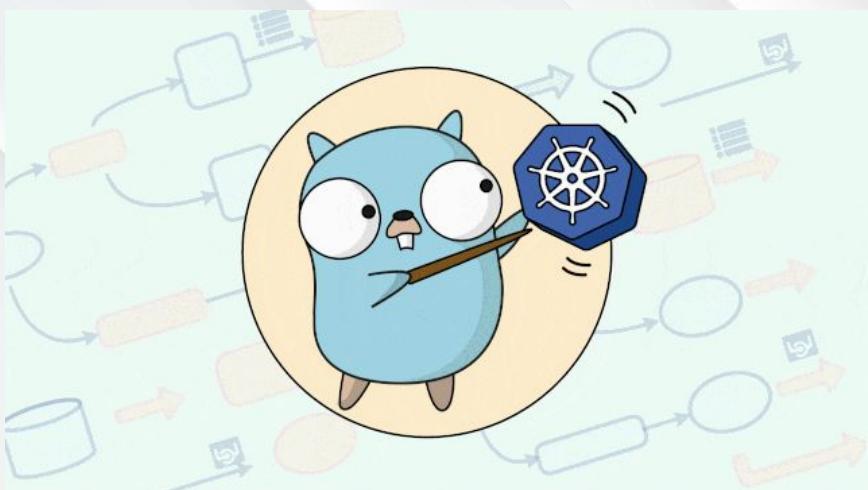
replicasets

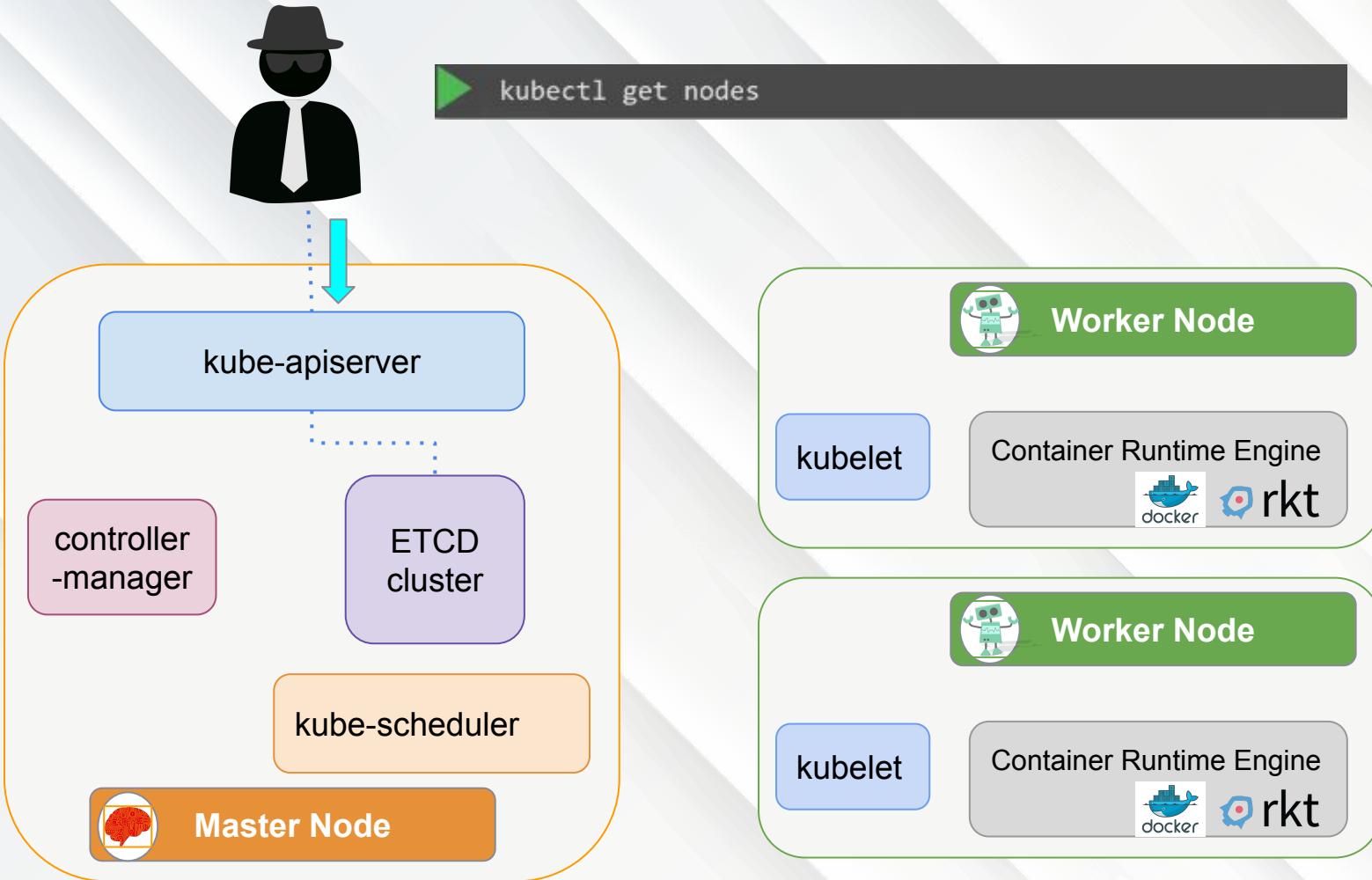
deployments

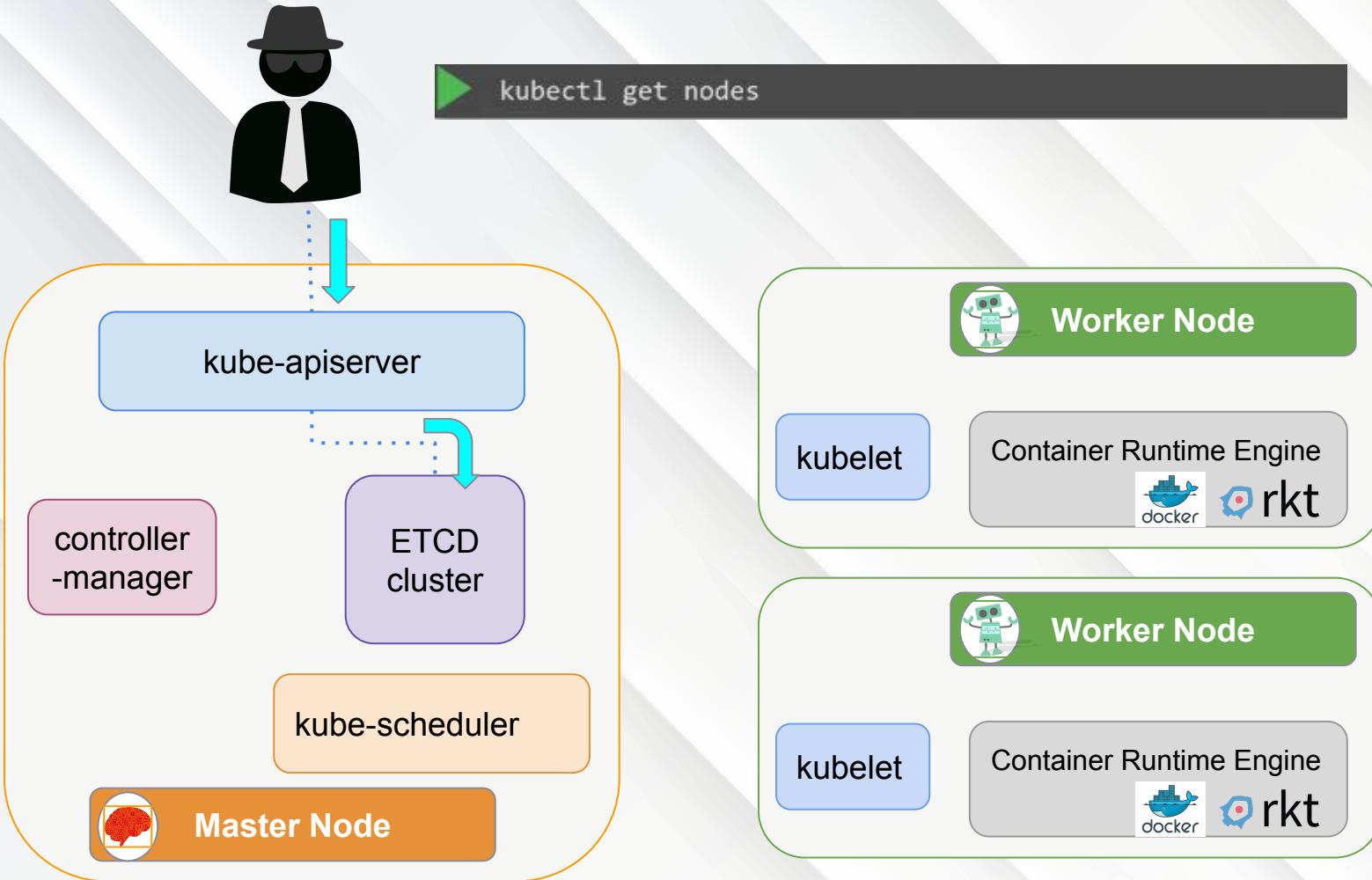
roles

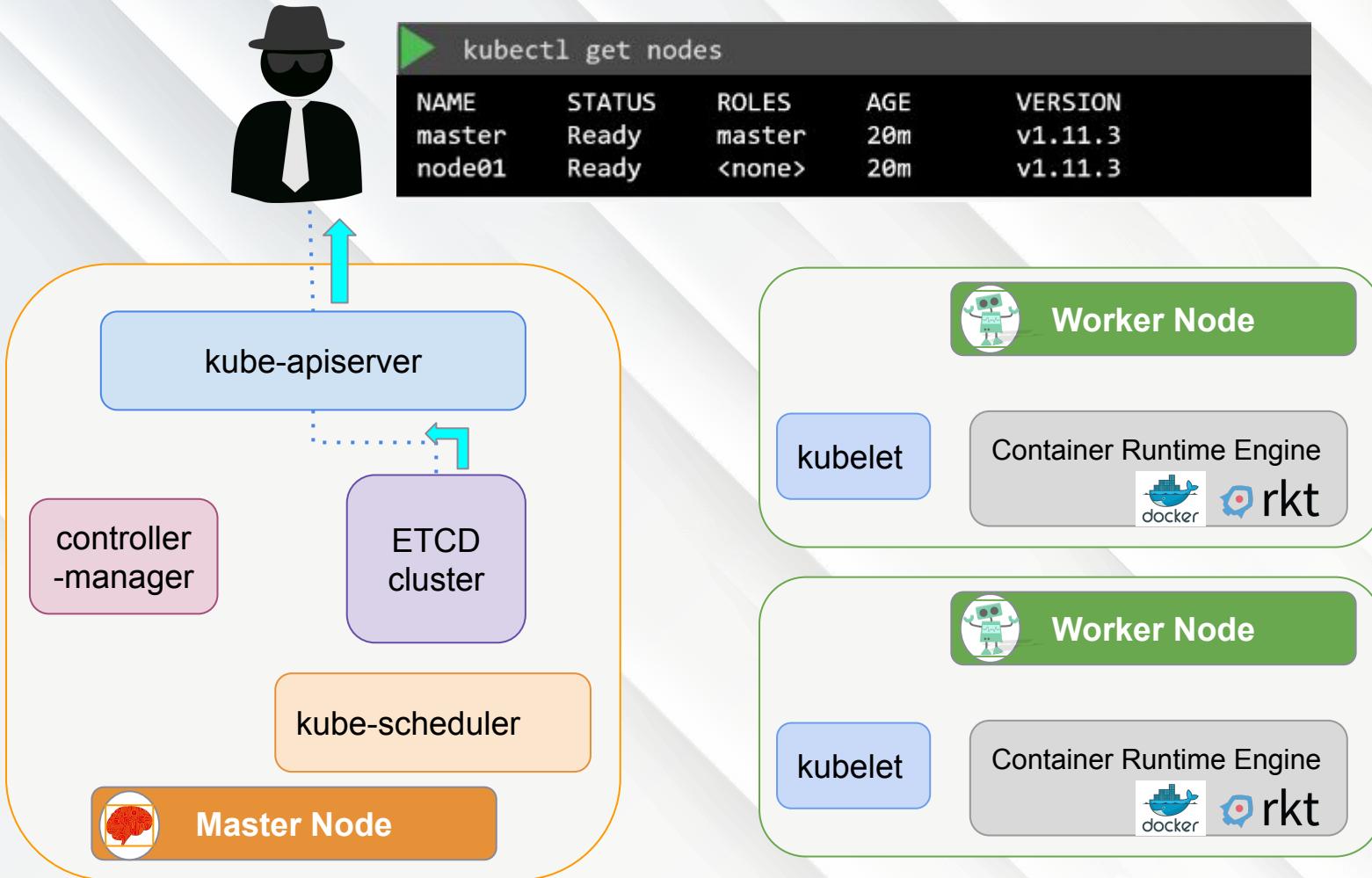
secrets

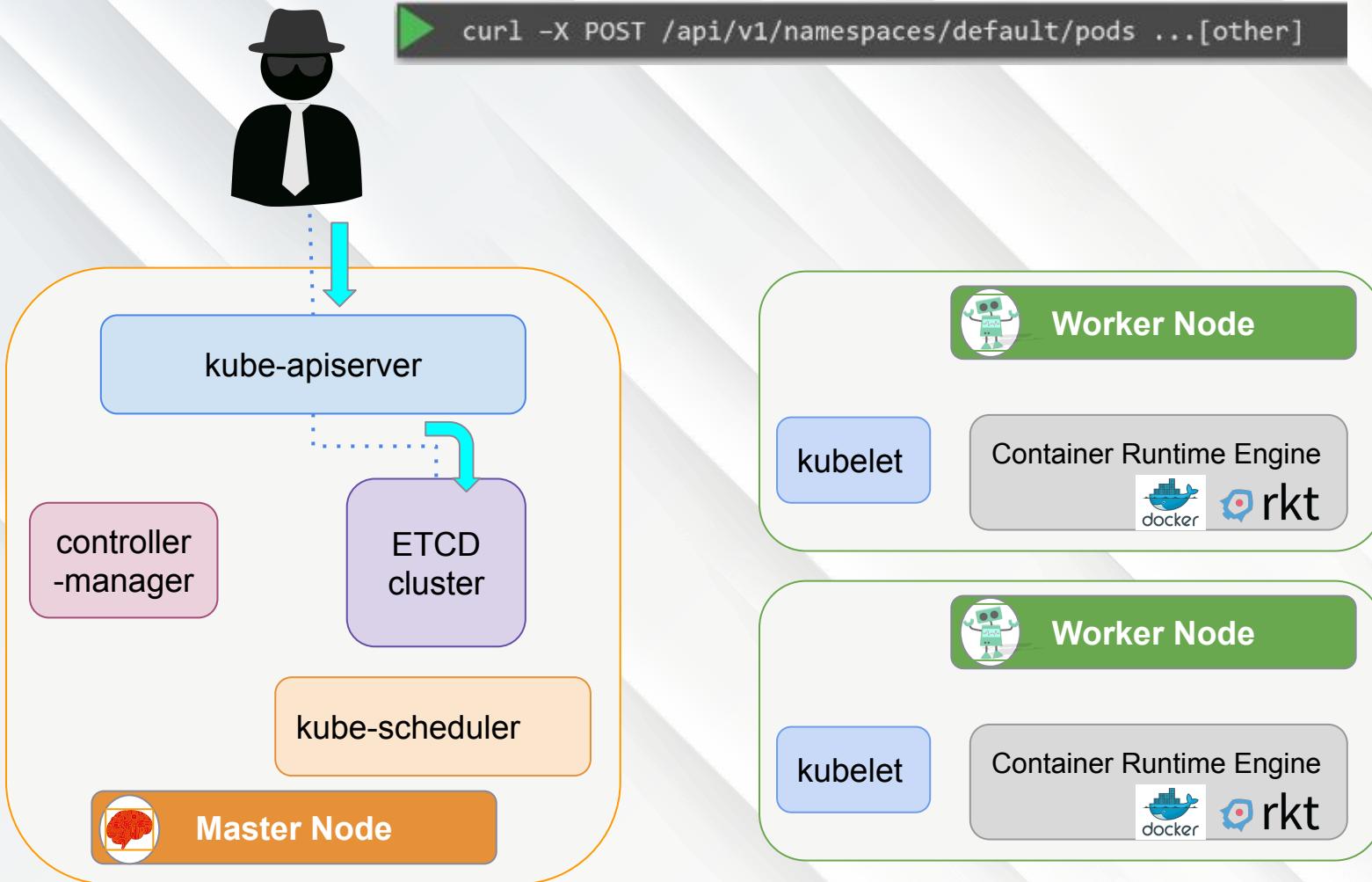
# Kube-apiserver

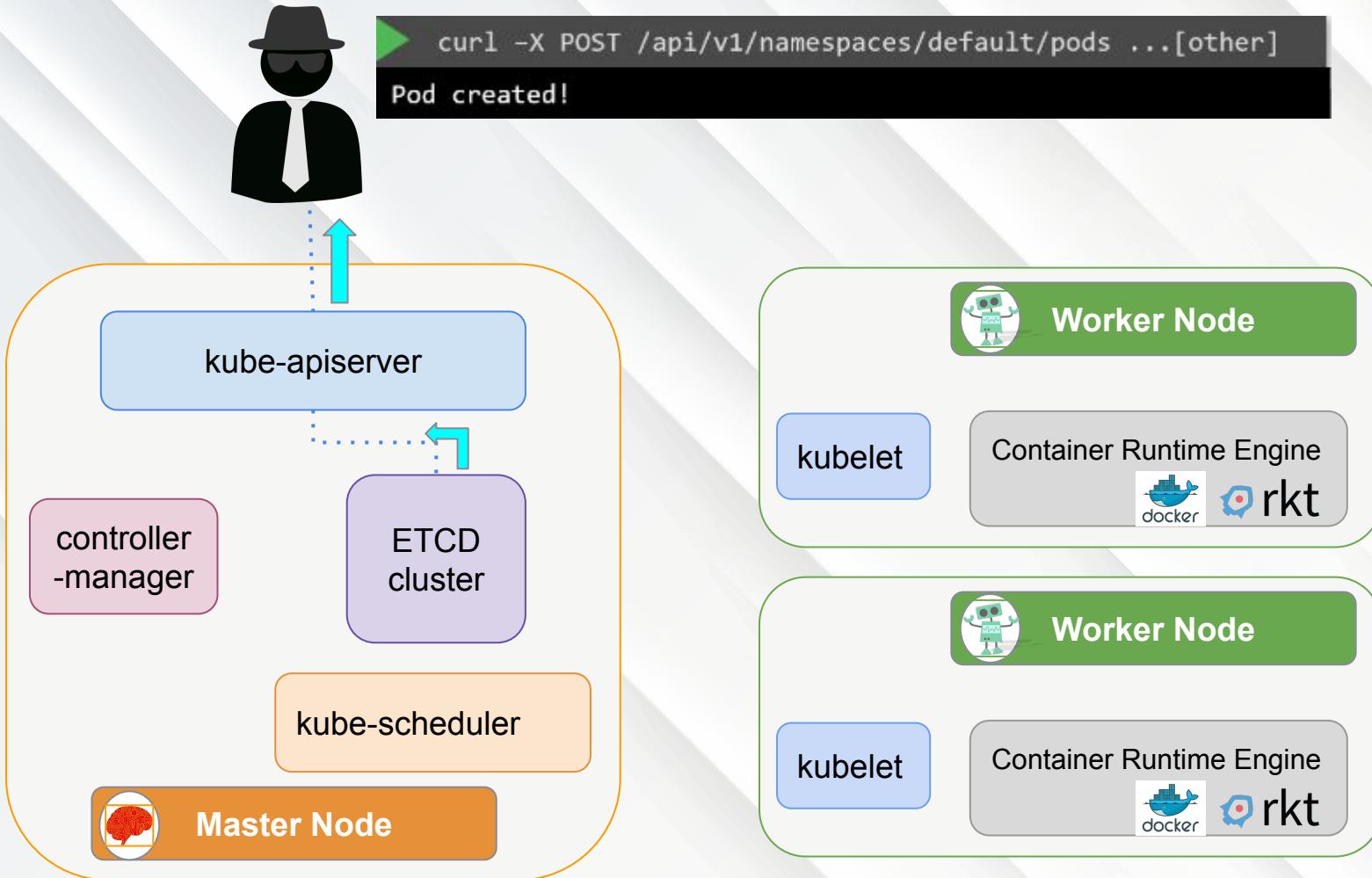








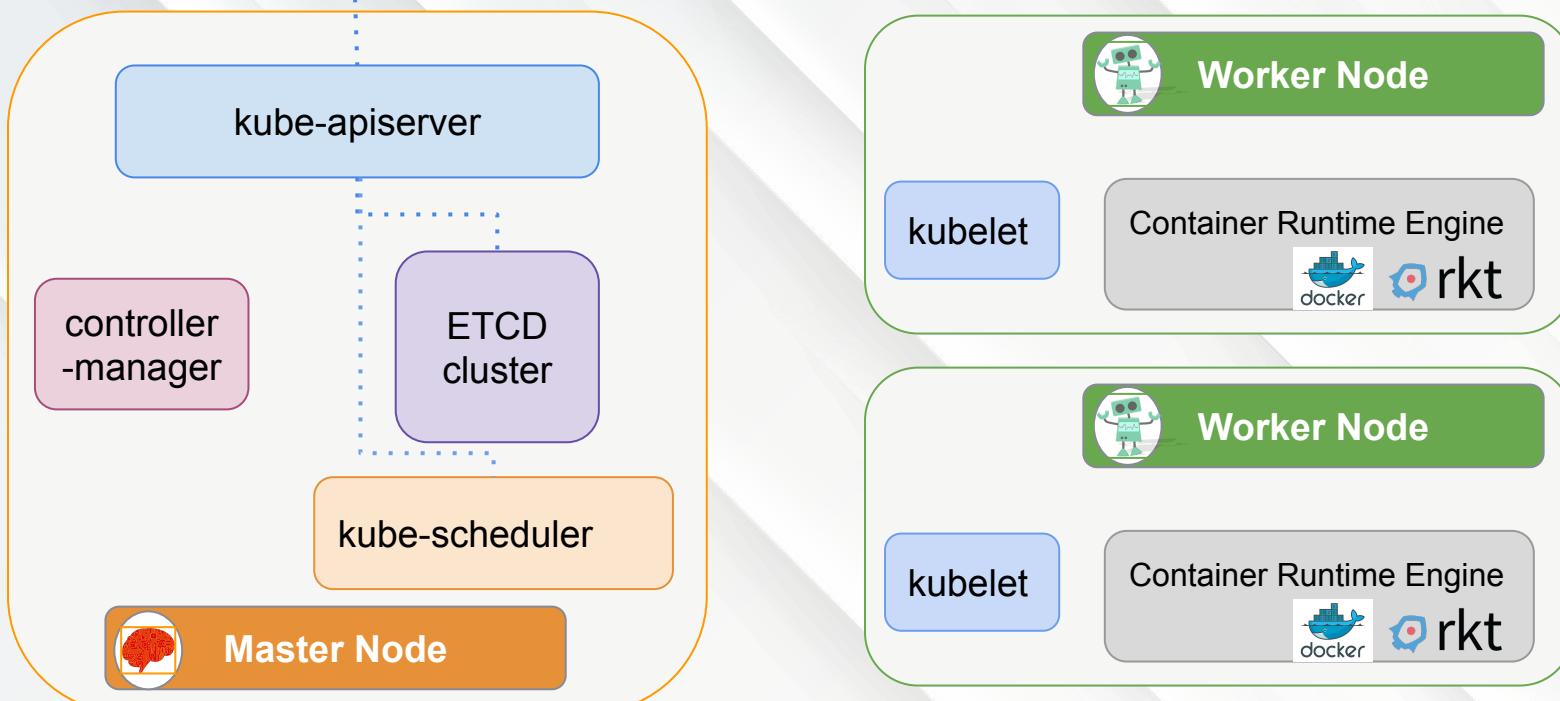






```
curl -X POST /api/v1/namespaces/default/pods ...[other]
```

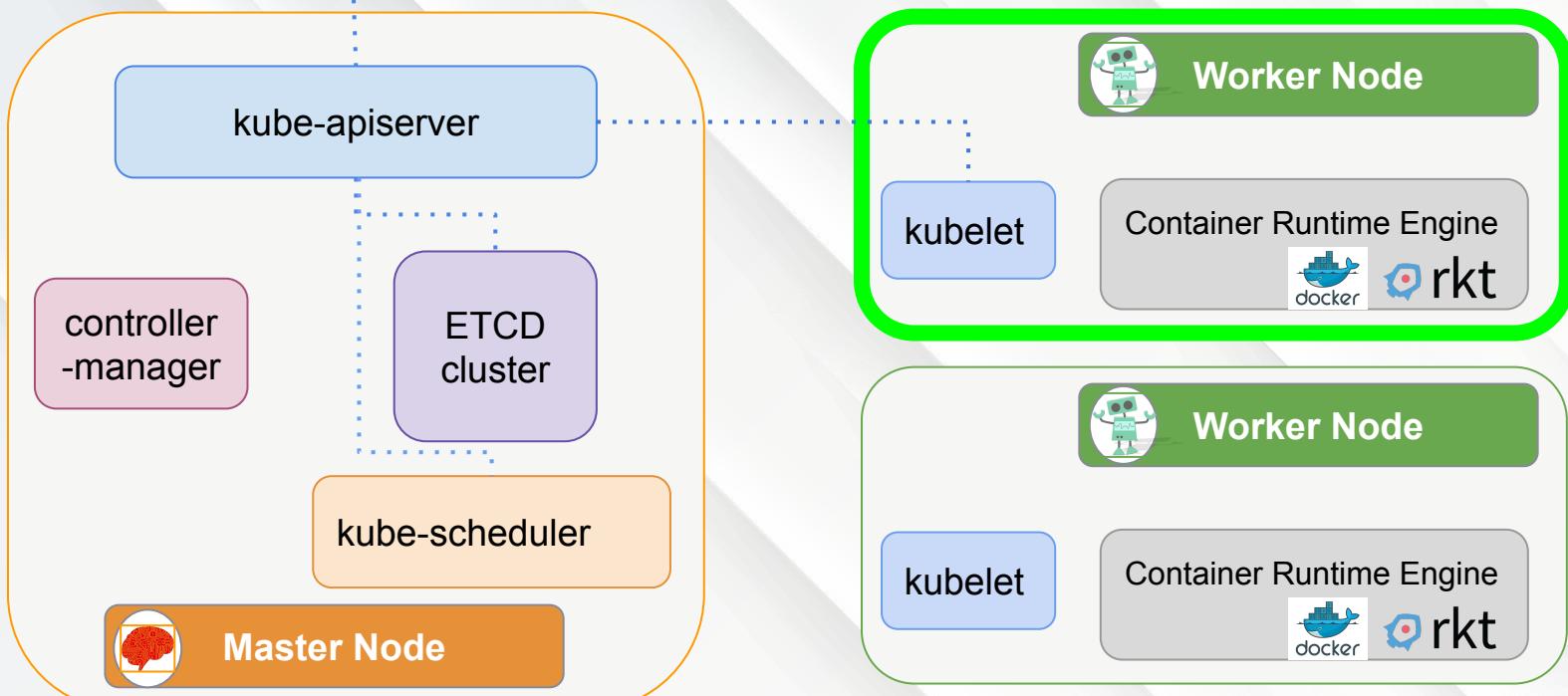
Pod created!





```
curl -X POST /api/v1/namespaces/default/pods ...[other]
```

Pod created!



# Kube-apiserver

Autenticar o usuário

Validar as requisições

Recuperar dados (solicitar)

Atualizar o ETCD

Scheduler

Kubelet

# Kube-apiserver

```
kubectl get pods -n kube-system
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-78fcdf6894-hwrq9	1/1	Running	0	16m
kube-system	coredns-78fcdf6894-rzhjr	1/1	Running	0	16m
kube-system	etcd-master	1/1	Running	0	15m
kube-system	kube-apiserver-master	1/1	Running	0	15m
kube-system	kube-controller-manager-master	1/1	Running	0	15m
kube-system	kube-proxy-lzt6f	1/1	Running	0	16m
kube-system	kube-proxy-zm5qd	1/1	Running	0	16m
kube-system	kube-scheduler-master	1/1	Running	0	15m
kube-system	weave-net-29z42	2/2	Running	1	16m
kube-system	weave-net-snmdl				

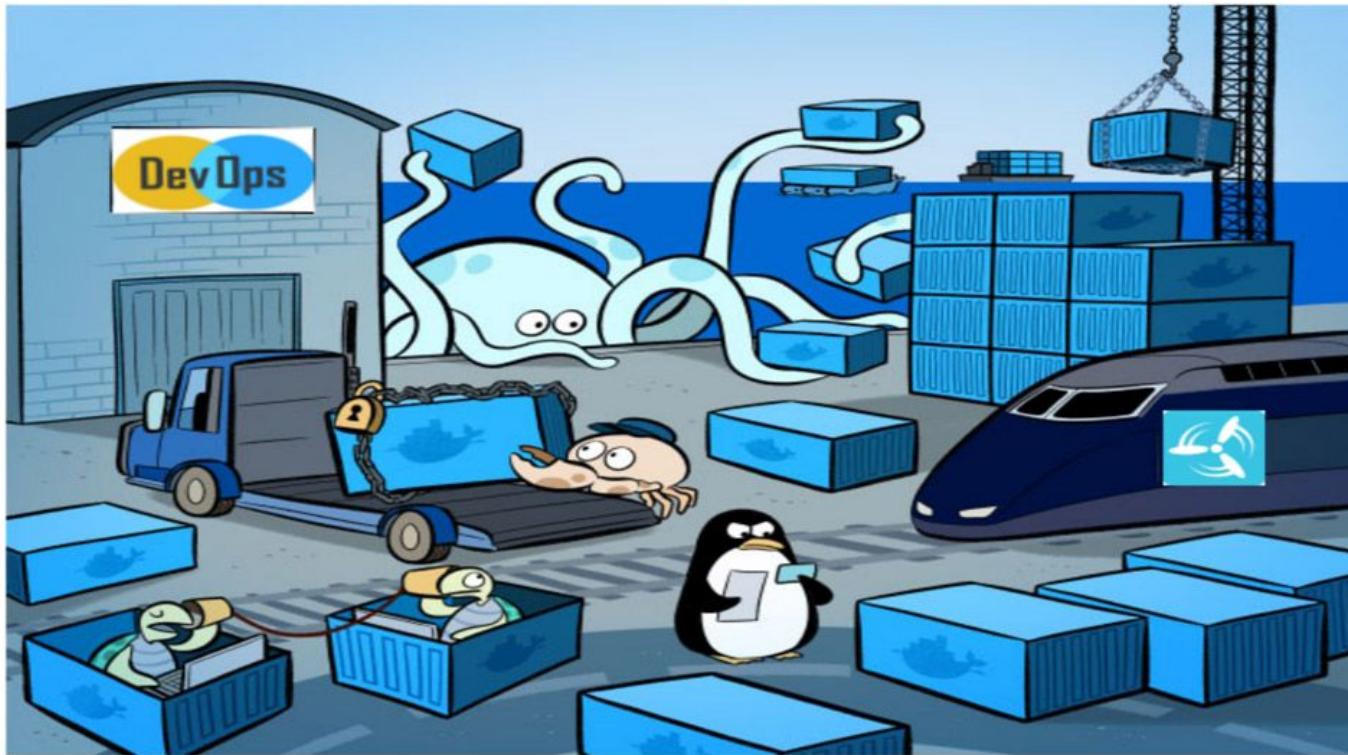
Kubeadm

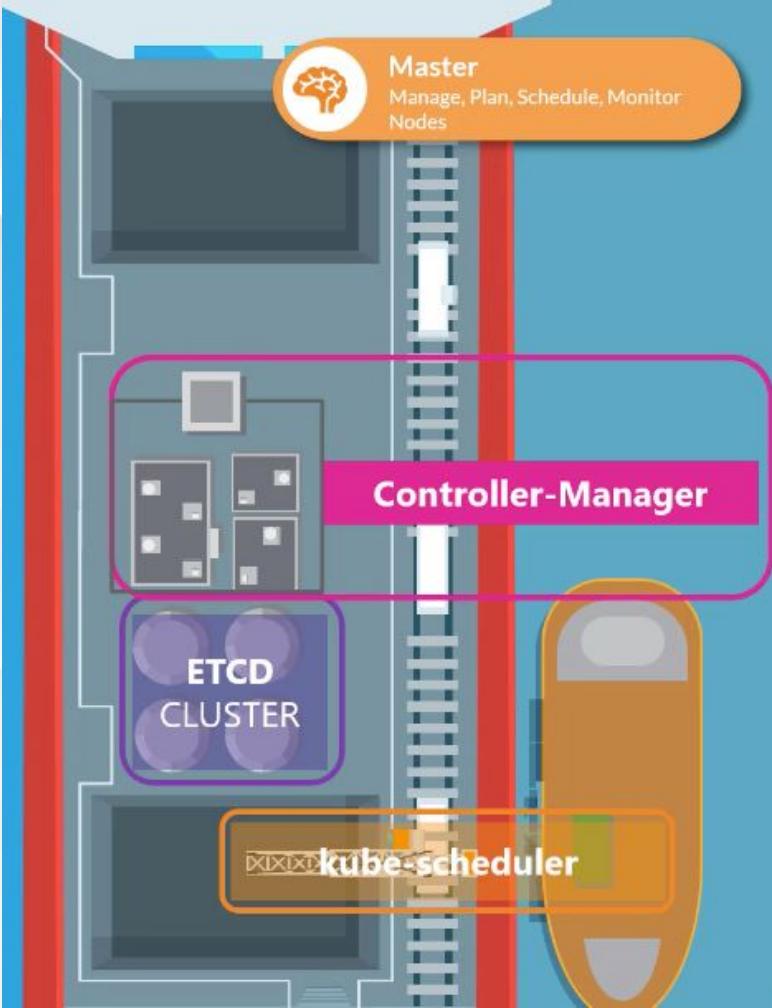
```
+ kubectl get pods -n kube-system
```

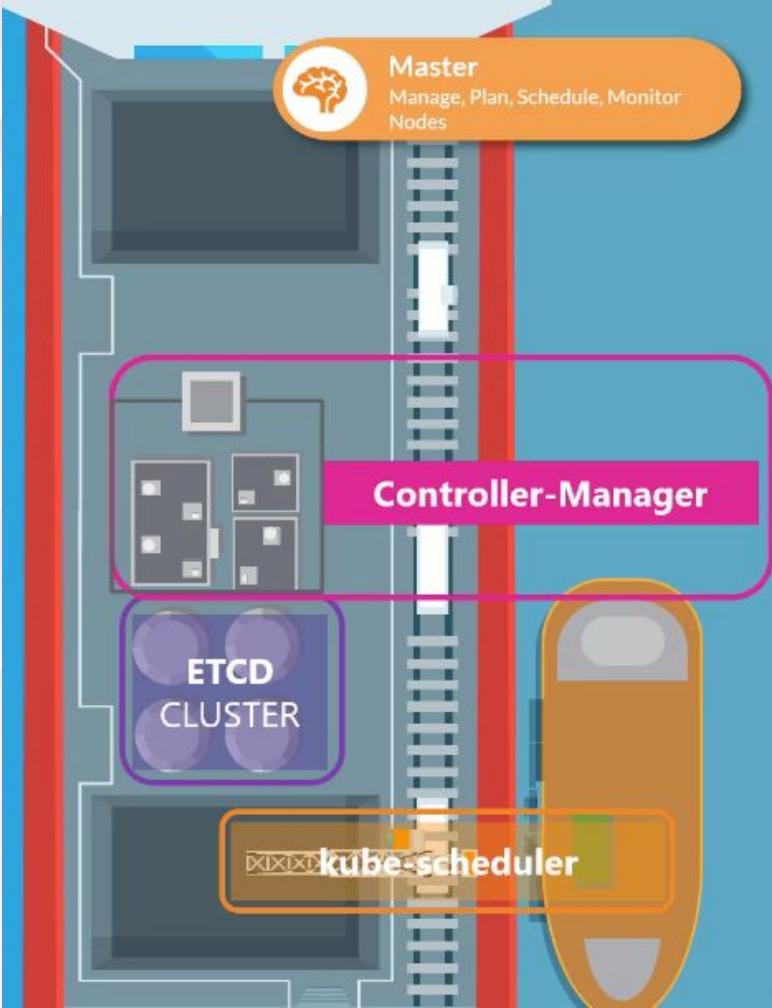
NAME	READY	STATUS	RESTARTS	AGE
coredns-5644d7b6d9-2kw9p	1/1	Running	26	209d
coredns-5644d7b6d9-gdfqh	1/1	Running	26	209d
etcd-docker-desktop	1/1	Running	49	209d
kube-apiserver-docker-desktop	1/1	Running	48	209d
kube-controller-manager-docker-desktop	1/1	Running	28	209d
kube-proxy-6b9gz	1/1	Running	26	209d
kube-scheduler-docker-desktop	1/1	Running	28	209d
storage-provisioner	1/1	Running	52	209d
vpnkit-controller	1/1	Running	27	209d

Docker

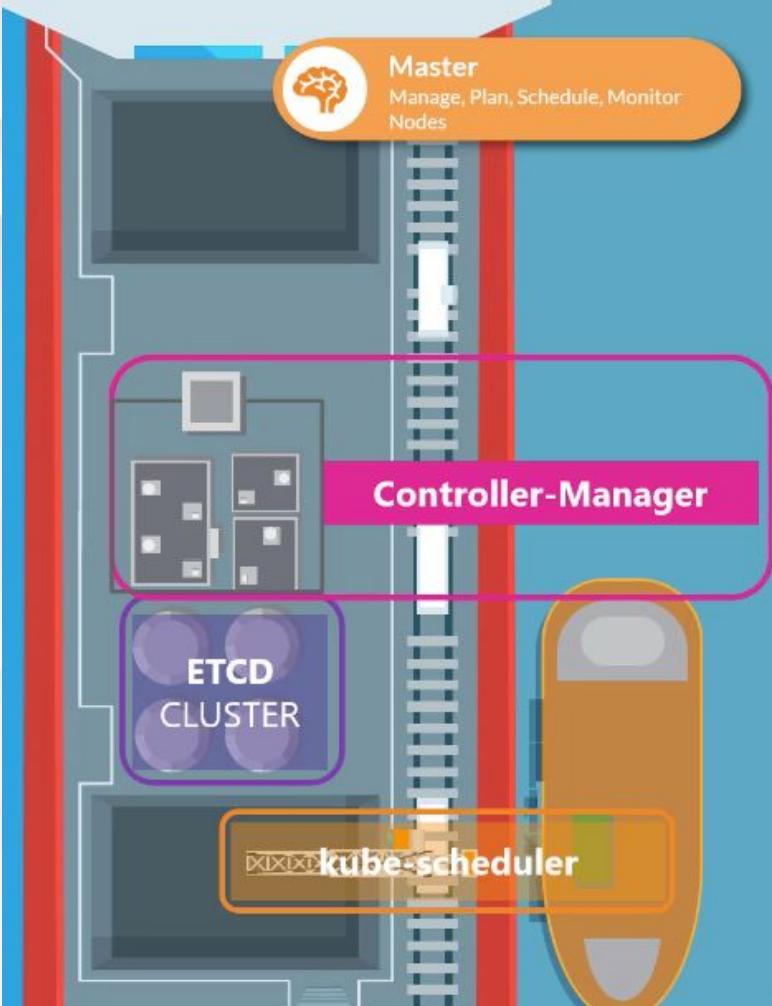
# Kube Controller Manager





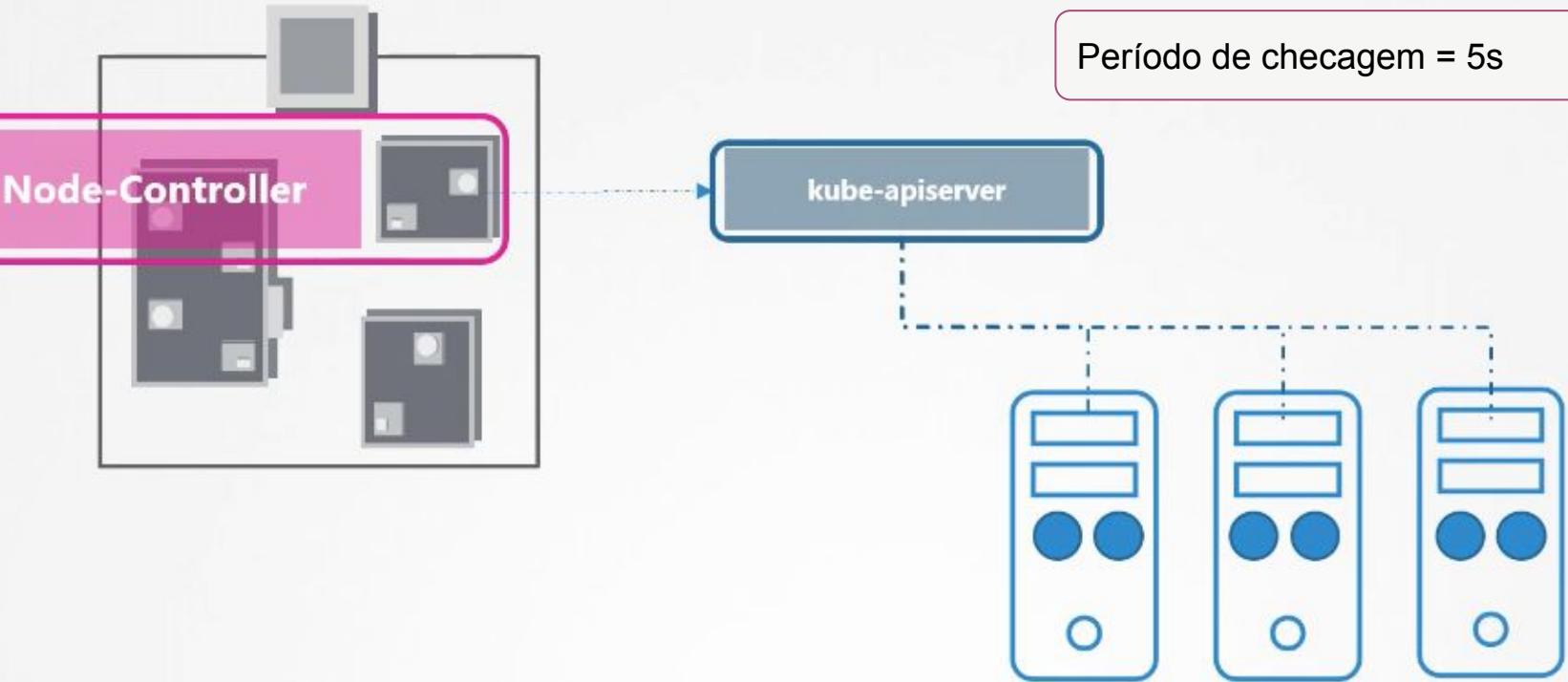


Verificando Status



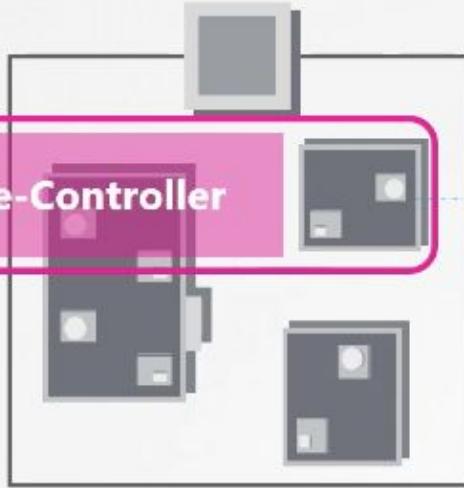
Verificando Status

Remediando situações

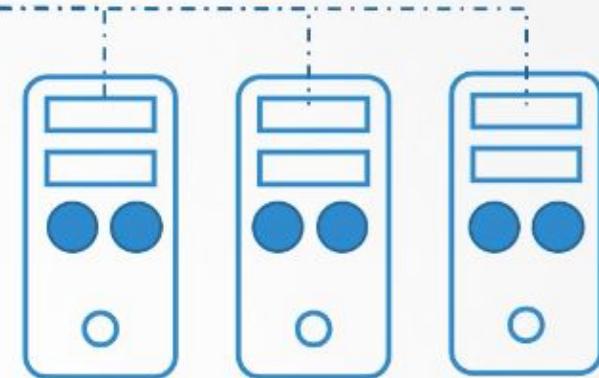


Período de checagem = 5s

Node-Controller

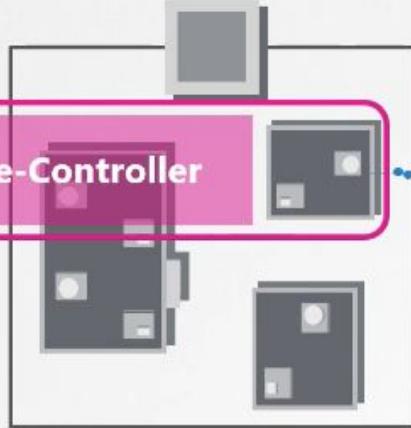


kube-apiserver



▶ kubectl get nodes

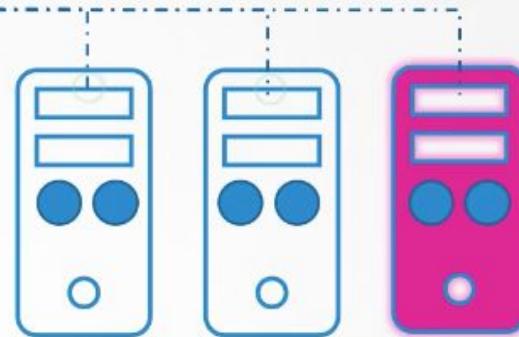
NAME	STATUS	ROLES	AGE	VERSION
worker-1	Ready	<none>	8d	v1.13.0
worker-2	Ready	<none>	8d	v1.13.0



Período de checagem = 5s

Grace Period = 40s

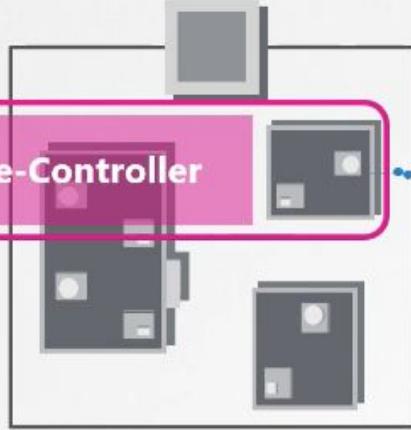
POD Eviction Timeout = 5m



▶ kubectl get nodes

NAME	STATUS	ROLES	AGE	VERSION
worker-1	Ready	<none>	8d	v1.13.0
worker-2	NotReady	<none>	8d	v1.13.0

UNREACHABLE

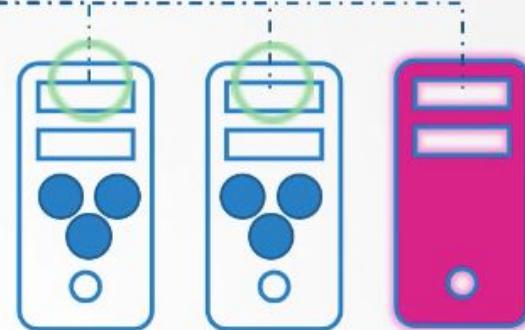


kube-apiserver

Período de checagem = 5s

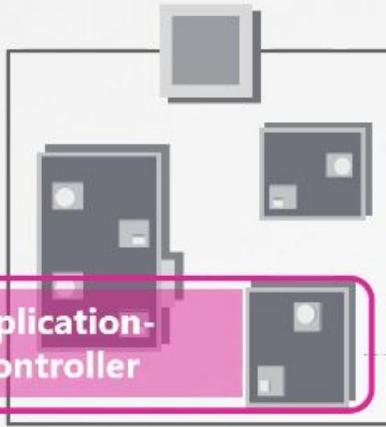
Grace Period = 40s

POD Eviction Timeout = 5m



▶ kubectl get nodes

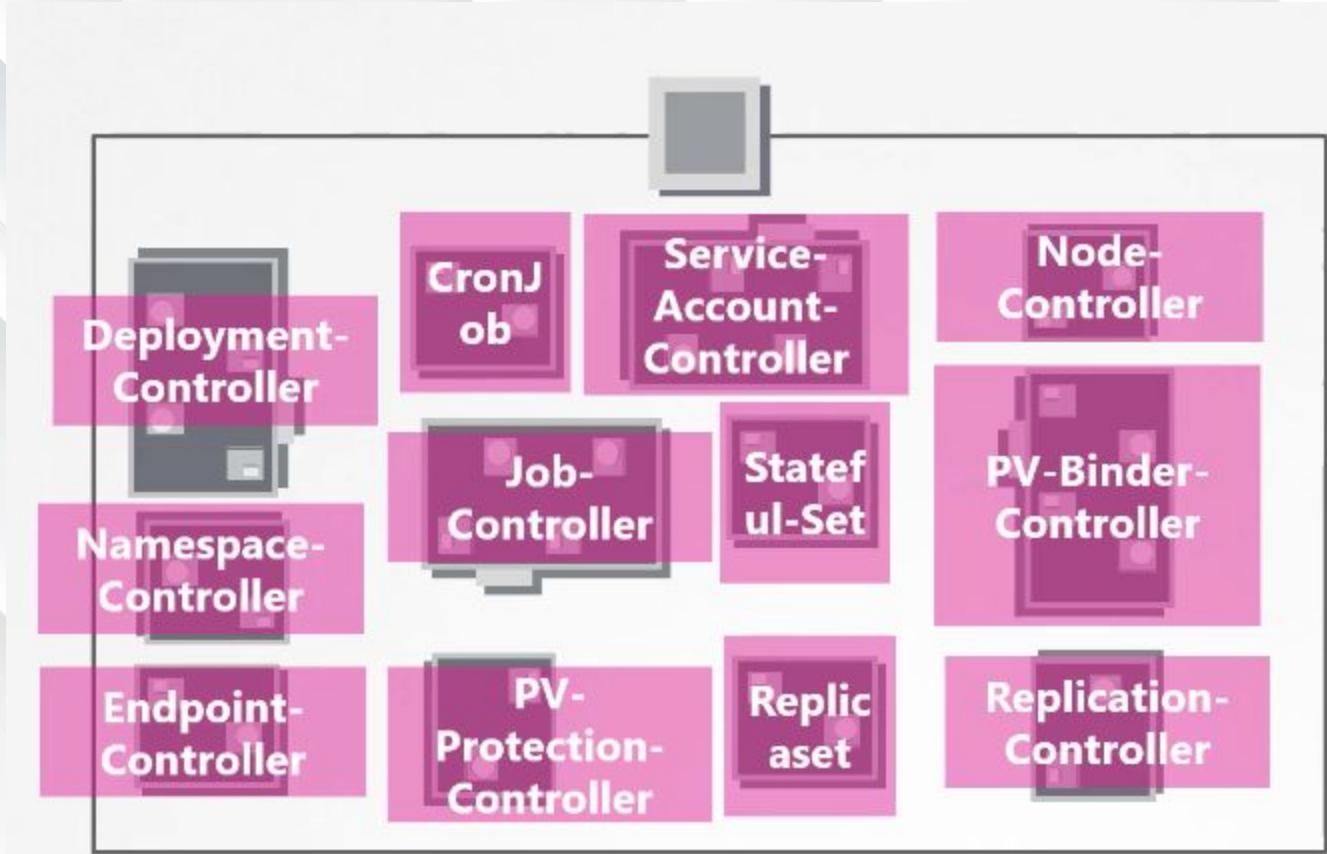
NAME	STATUS	ROLES	AGE	VERSION
worker-1	Ready	<none>	8d	v1.13.0
worker-2	NotReady	<none>	8d	v1.13.0



Período de checagem = 5s

Grace Period = 40s

POD Eviction Timeout = 5m





The diagram illustrates the Kube-Controller-Manager architecture. At the top center is a square node labeled "Kube-Controller-Manager". Below it, several smaller square nodes are arranged in a grid-like pattern, each containing a smaller square icon. A line connects the central manager node to one of the worker nodes.

# Kube-Controller-Manager

```
kubectl get pods -n kube-system
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-78fcdf6894-hwrq9	1/1	Running	0	16m
kube-system	coredns-78fcdf6894-rzhjr	1/1	Running	0	16m
kube-system	etcd-master	1/1	Running	0	15m
kube-system	kube-apiserver-master	1/1	Running	0	15m
kube-system	kube-controller-manager-master	1/1	Running	0	15m
kube-system	kube-proxy-lzt6f	1/1	Running	0	16m
kube-system	kube-proxy-zm5qd	1/1	Running	0	16m
kube-system	kube-scheduler-master	1/1	Running	0	15m
kube-system	weave-net-29z42	2/2	Running	1	16m
kube-system	weave-net-snmdl	2/2	Running	1	16m

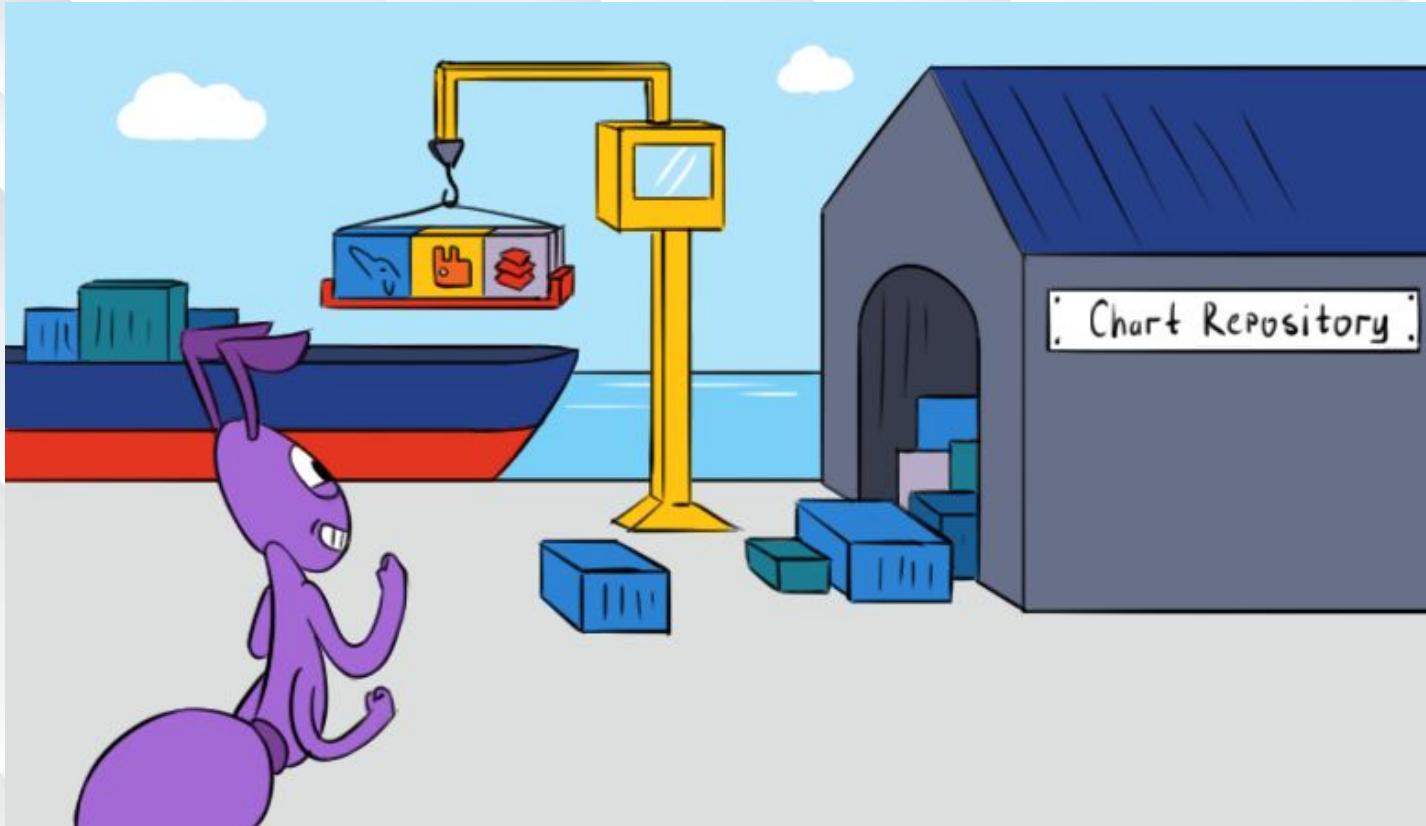
Kubeadm

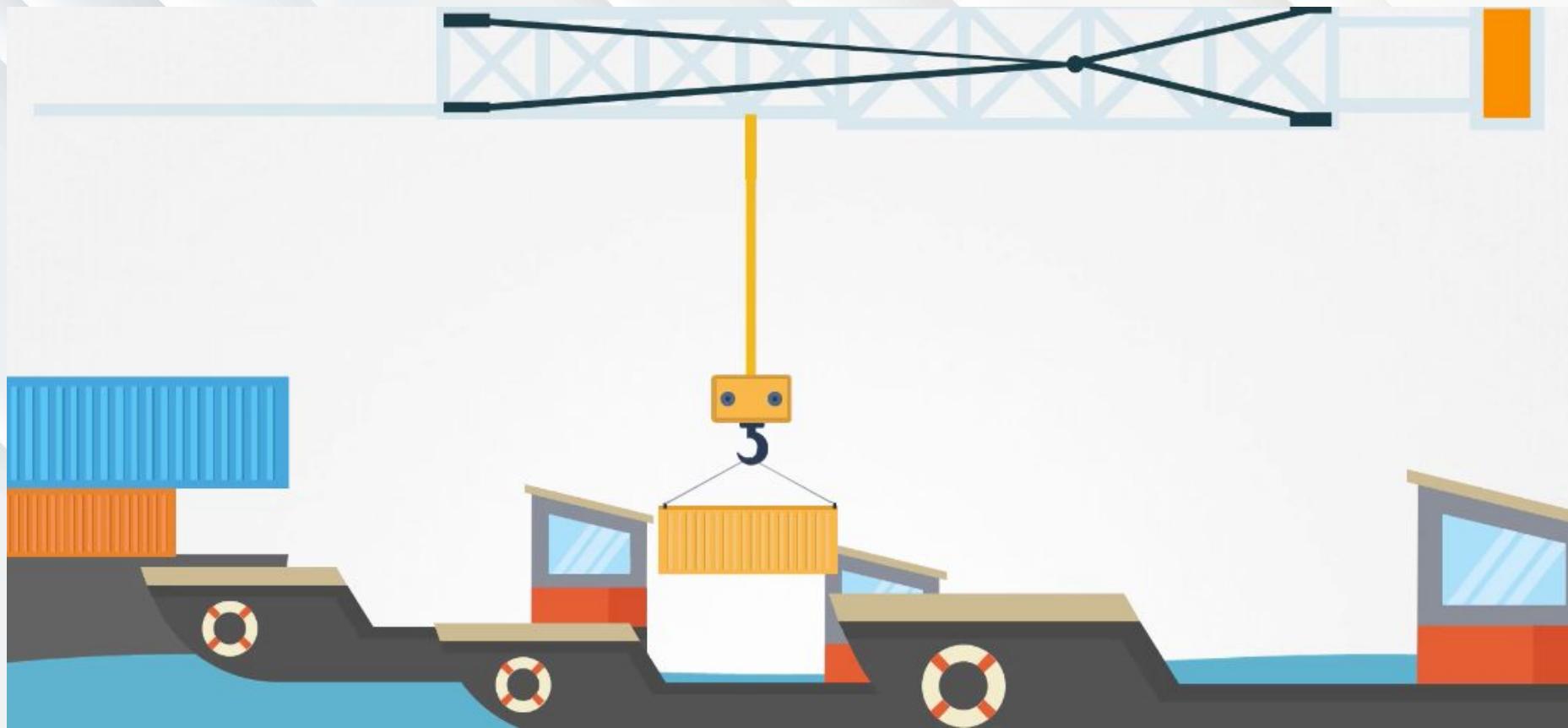
```
+ kubectl get pods -n kube-system
```

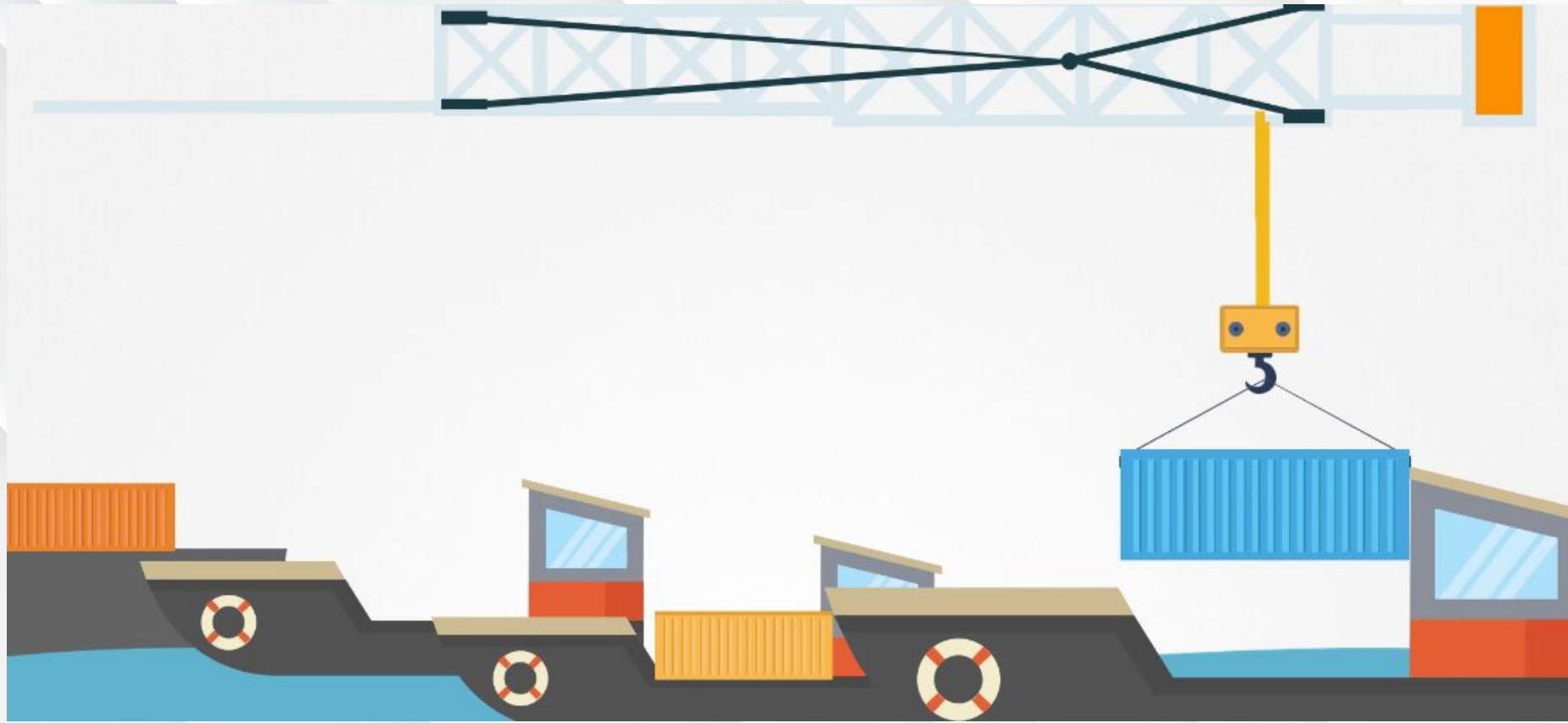
NAME	READY	STATUS	RESTARTS	AGE
coredns-5644d7b6d9-2kw9p	1/1	Running	26	209d
coredns-5644d7b6d9-gdfqh	1/1	Running	26	209d
etcd-docker-desktop	1/1	Running	49	209d
kube-apiserver-docker-desktop	1/1	Running	48	209d
kube-controller-manager-docker-desktop	1/1	Running	28	209d
kube-proxy-6b9gz	1/1	Running	26	209d
kube-scheduler-docker-desktop	1/1	Running	28	209d
storage-provisioner	1/1	Running	52	209d
vpnkit-controller	1/1	Running	27	209d

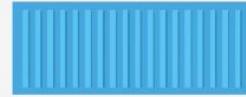
Docker

# Kube Scheduler







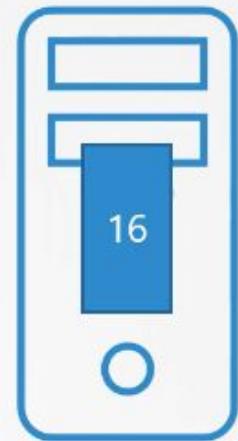
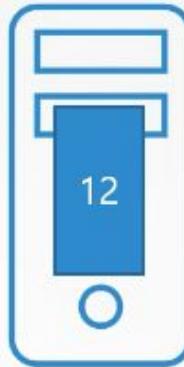
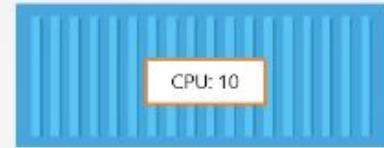


Scheduler verifica o tempo todo os pods que precisam ser alocados e os nodes existentes.

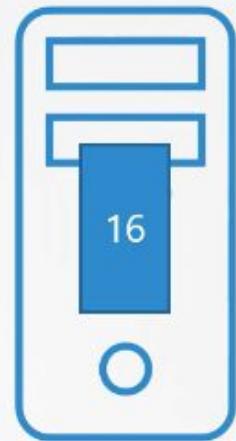
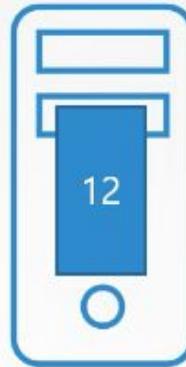
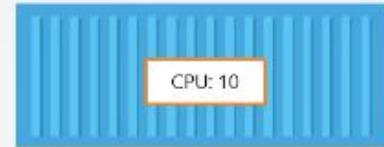
Compara os requisitos e toma as decisões adequadas.



Fase 1: Filtrar os  
nodes

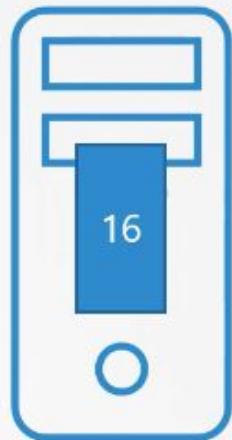
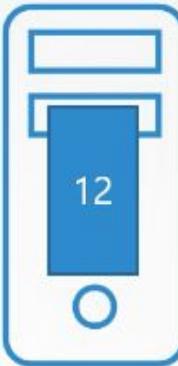
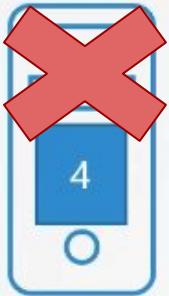
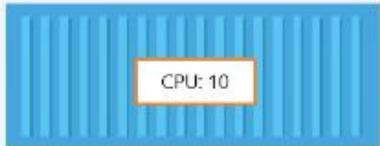


Fase 1: Filtrar os  
nodes



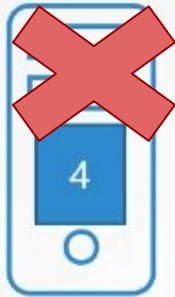
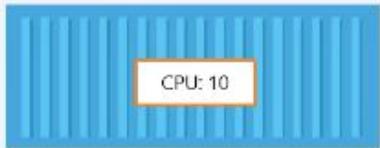
Fase 1: Filtrar os nodes

Fase 2: Ranking dos nodes

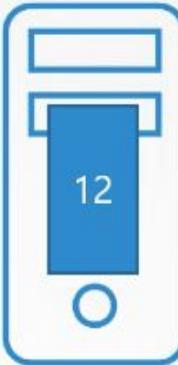


Fase 1: Filtrar os nodes

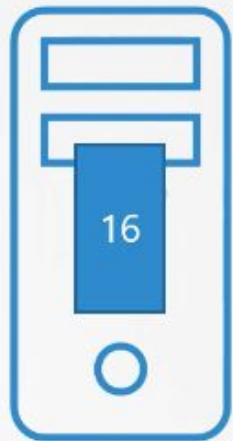
Fase 2: Ranking dos nodes



$$12 - 10 = 2$$

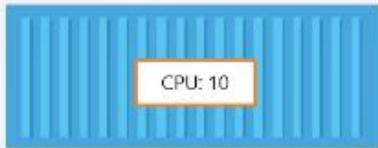


$$16 - 10 = 6$$

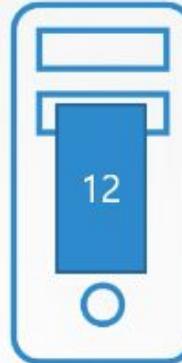


Fase 1: Filtrar os nodes

Fase 2: Ranking dos nodes



$$12 - 10 = 2$$



# Adiante:

- Taints
- Tolerations
- Node Selectors
- Affinity

```
→ kubectl get pods -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-5644d7b6d9-2kw9p	1/1	Running	26	209d
coredns-5644d7b6d9-gdfqh	1/1	Running	26	209d
etcd-docker-desktop	1/1	Running	49	209d
kube-apiserver-docker-desktop	1/1	Running	48	209d
kube-controller-manager-docker-desktop	1/1	Running	28	209d
kube-proxy-6b9gz	1/1	Running	26	209d
kube-scheduler-docker-desktop	1/1	Running	28	209d
storage-provisioner	1/1	Running	52	209d
vpnkit-controller	1/1	Running	27	209d

Docker

# Kubelet





## Master

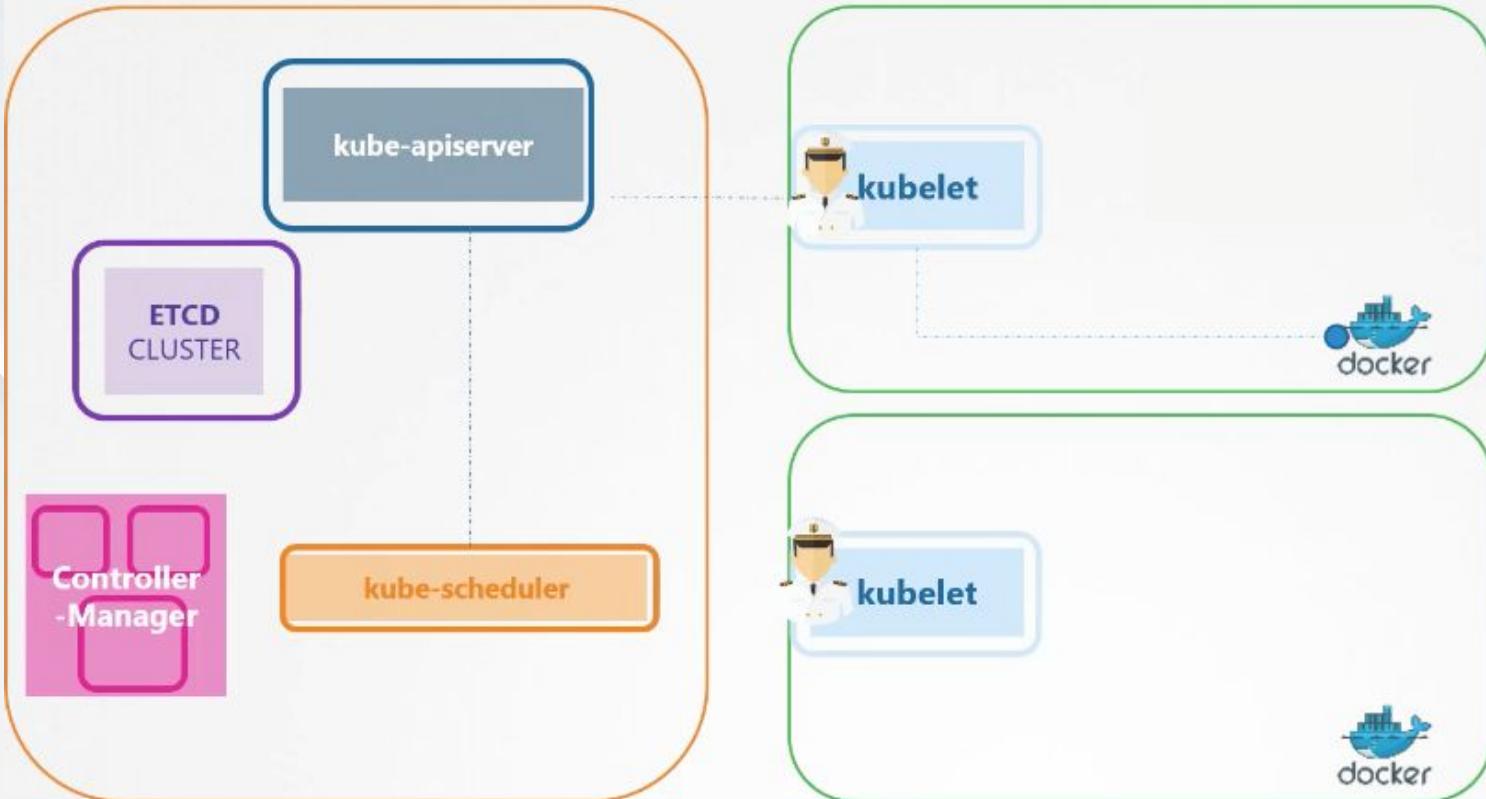
Manage, Plan, Schedule, Monitor  
Nodes



## Worker Nodes

Host Application as Containers

Registrar um  
node





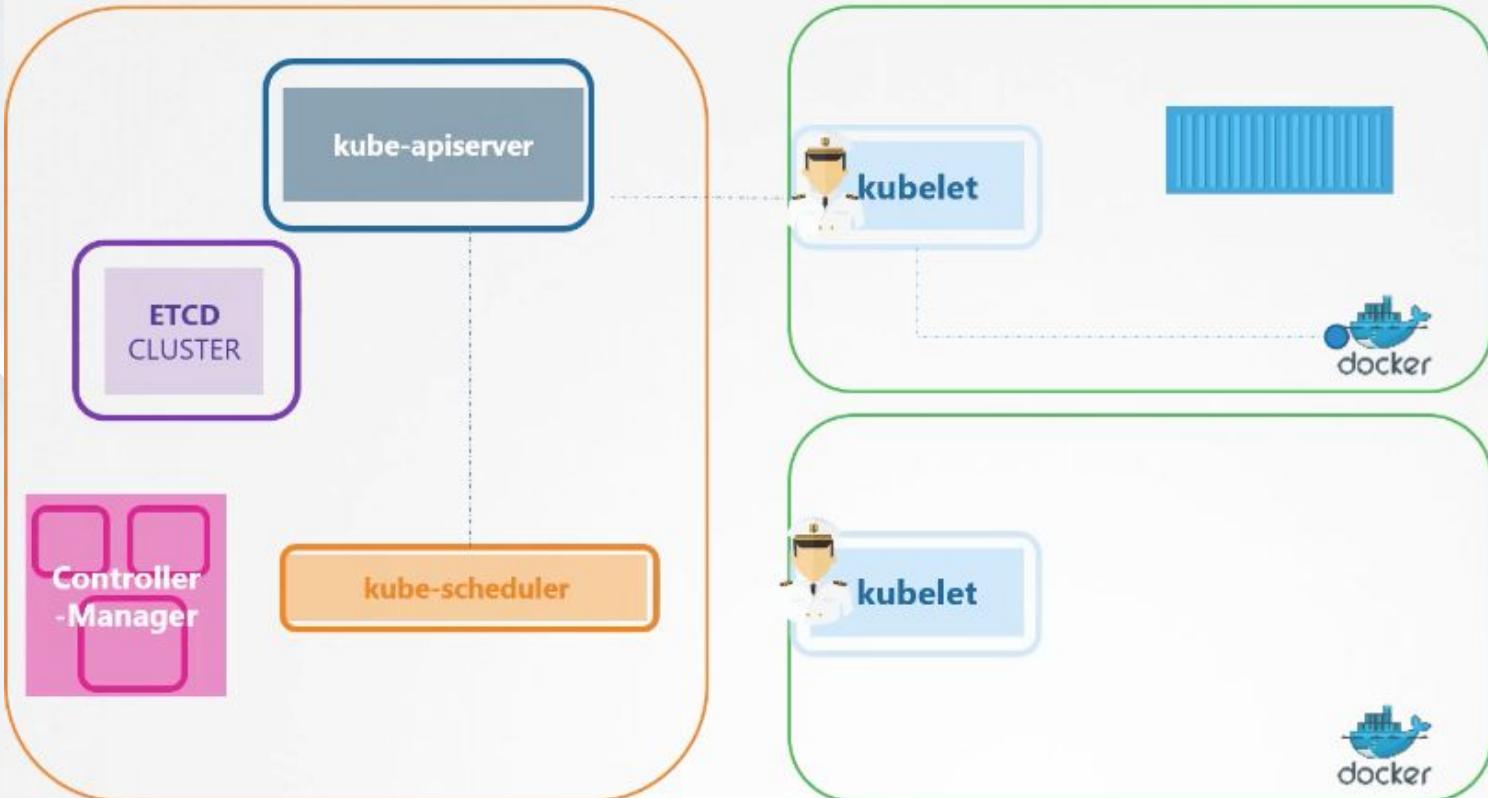
## Master

Manage, Plan, Schedule, Monitor  
Nodes



## Worker Nodes

Host Application as Containers



Registrar um  
node

Criar PODs

# Instalação do Kubelet

```
wget https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kubelet
```

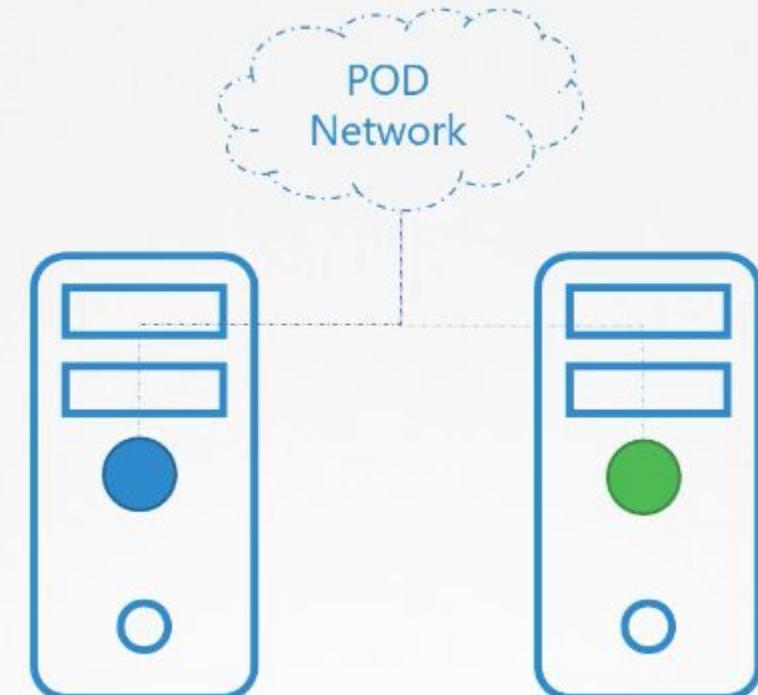
## kubelet.service

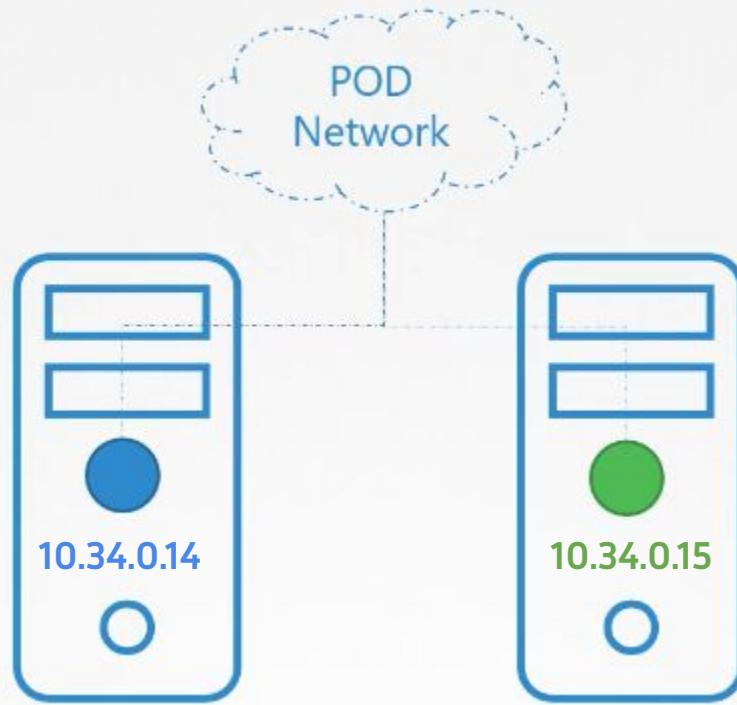
```
ExecStart=/usr/local/bin/kubelet \
--config=/var/lib/kubelet/kubelet-config.yaml \
--container-runtime=remote \
--container-runtime-endpoint=unix:///var/run/containerd/containerd.sock \
--image-pull-progress-deadline=2m \
--kubeconfig=/var/lib/kubelet/kubeconfig \
--network-plugin=cni \
--register-node=true \
--v=2
```

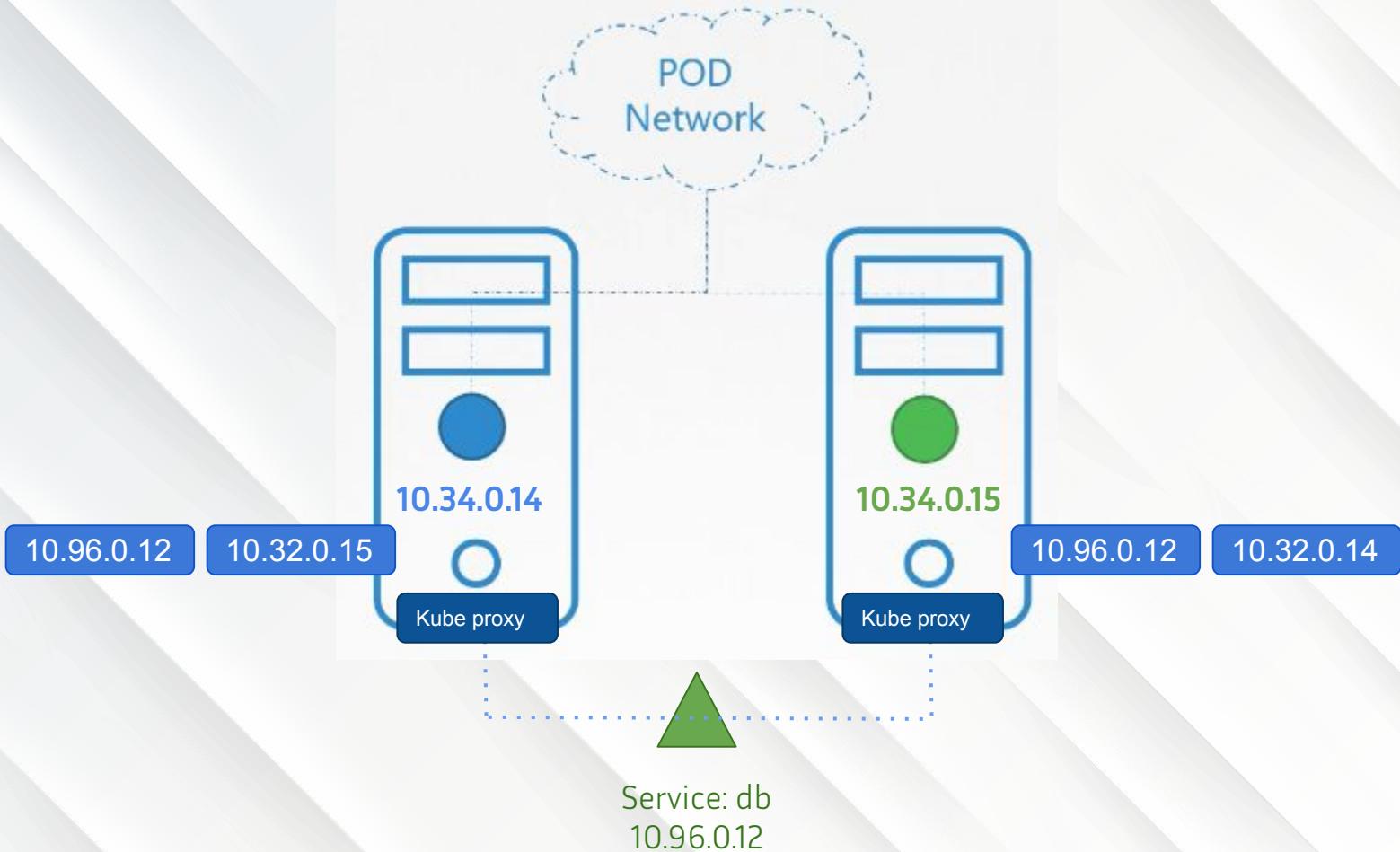
- **Docker** não possui (não precisa) do kubelet pois possui apenas um node que é o master.
- **Kubeadm** requer que você instale manualmente cada kubelet
- **Provedores** cloud automatizam para você

# Kube Proxy









# Instalação

```
→ kubectl get pods -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-5644d7b6d9-2kw9p	1/1	Running	27	211d
coredns-5644d7b6d9-gdfqh	1/1	Running	27	211d
etcd-docker-desktop	1/1	Running	50	211d
kube-apiserver-docker-desktop	1/1	Running	49	211d
kube-controller-manager-docker-desktop	1/1	Running	29	211d
kube-proxy-6b9gz	1/1	Running	27	211d
kube-scheduler-docker-desktop	1/1	Running	29	211d
storage-provisioner	1/1	Running	53	211d
vpnkit-controller	1/1	Running	28	211d

```
~ took 2s
```

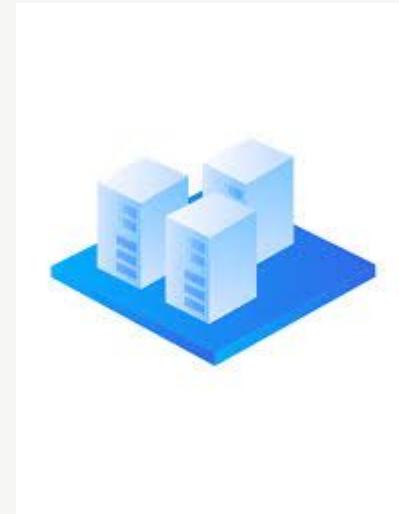
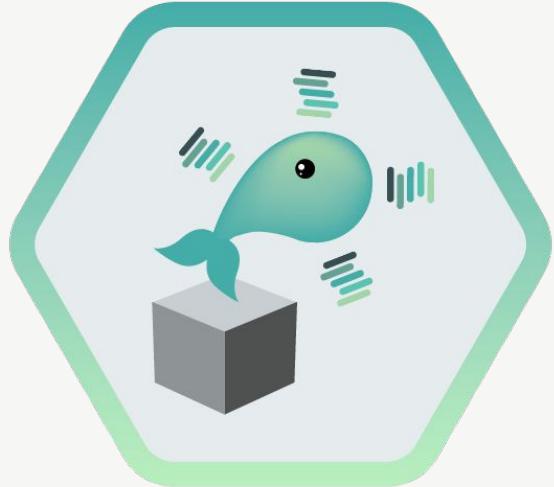
```
→ kubectl get daemonset -n kube-system
```

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
kube-proxy	1	1	1	1	1	beta.kubernetes.io/os=	54 1h
linux	211d						

Docker

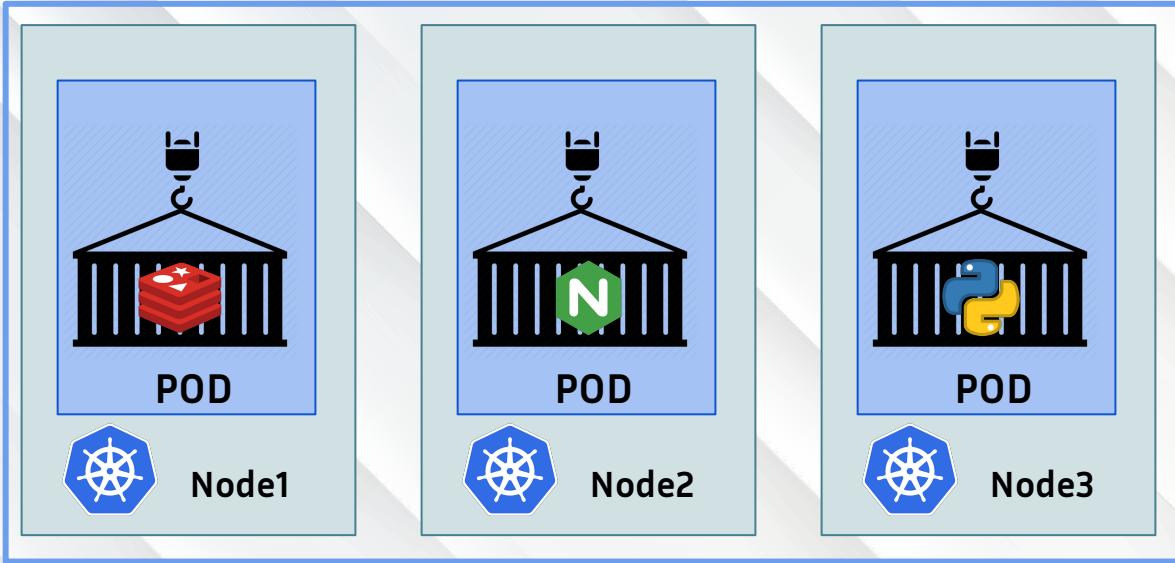


# Pods

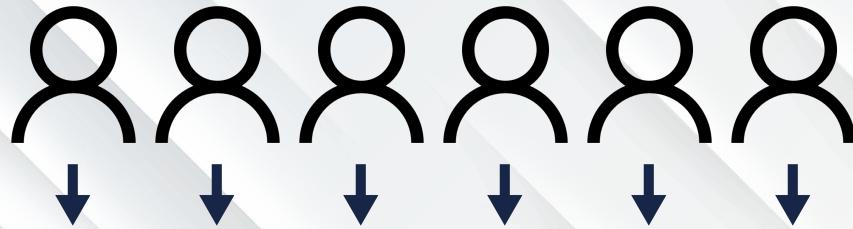


Node

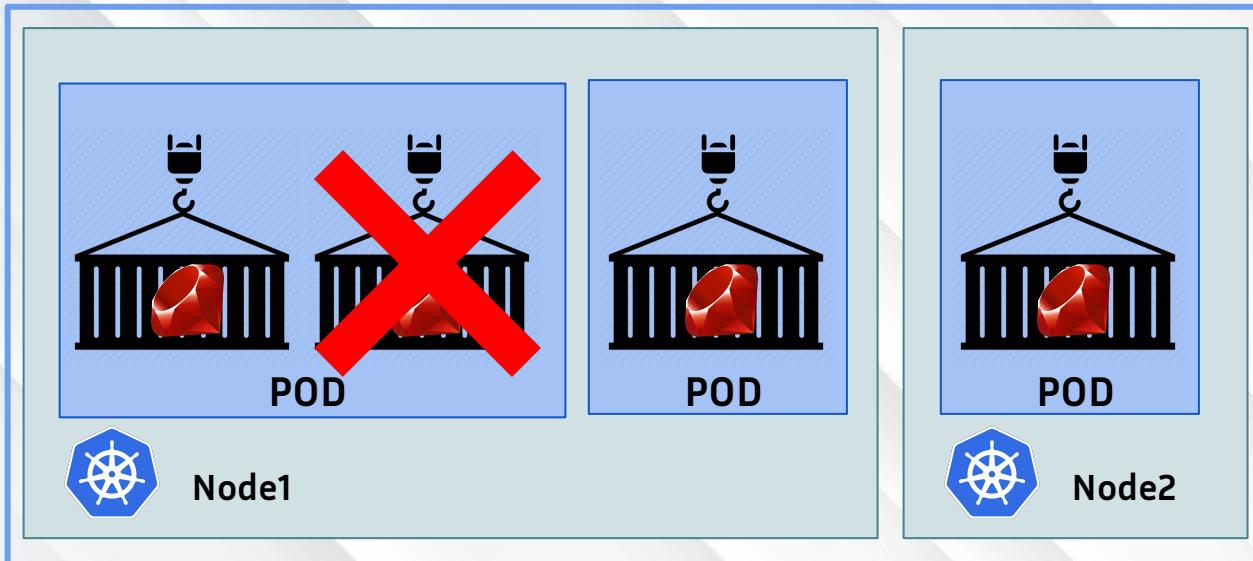
d



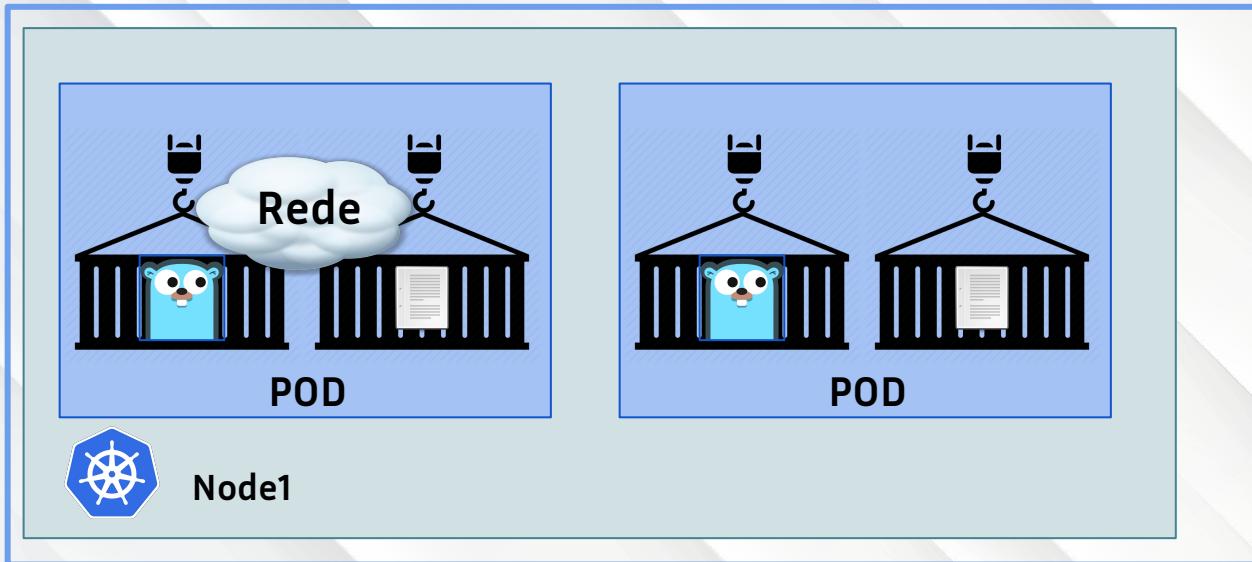
## Pods



Pods



# Pods - Multi-Container



## Pods - kubectl

```
kubectl run nginx --image nginx
```

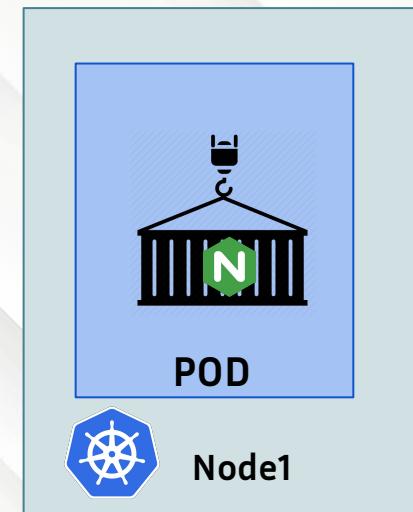
OU --image=nginx

```
kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx	0/1	ContainerCreating	0	1s

NAME	READY	STATUS	RESTARTS	AGE
nginx	1/1	Running	0	20s

```
kubectl delete pod nginx
```



# Pods - kubectl - Demo & Prática

<https://www.katacoda.com/courses/kubernetes/playground>



## Pods - Com YAML

```
- name: create users
  hosts: all
  tasks:
    - user:
        name: "{{ item }}"
        state: present
        groups: "{{ item }}"
      with_items:
        - { name: 'lin'
        - { name: 'lis
```



# Pods - Com YAML

pod-nginx.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: meuapp-pod
  labels:
    app: meuapp
    tipo: front-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```

The diagram illustrates the structure of the pod-nginx.yml file. It shows the following components:

- apiVersion:** v1 (String)
- kind:** Pod (String)
- metadata:** (Dicionário)
- name:** meuapp-pod
- labels:** (Dicionário)
  - app: meuapp
  - tipo: front-end
- spec:** (Lista/Array)
  - containers:** (Dicionário)
    - name: nginx-container
    - image: nginx

Tipo de Objeto	Versão de API
POD	v1
Service	v1
ReplicaSet	apps/v1
Deployment	apps/v1

kubectl create -f pod-nginx.yml

kubectl get pods

kubectl describe pod meuapp-pod

# Pods - Com YAML

```
Mounts:
  /var/run/secrets/kubernetes.io/serviceaccount from default-token-v5v8t (ro)
Conditions:
  Type        Status
  Initialized  True
  Ready       True
  ContainersReady  True
  PodScheduled  True
Volumes:
  default-token-v5v8t:
    Type:      Secret (a volume populated by a Secret)
    SecretName: default-token-v5v8t
    Optional:   false
  QoS Class:  BestEffort
  Node-Selectors: <none>
  Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s
                node.kubernetes.io/unreachable:NoExecute for 300s
                4006fb9
Events:
  Type  Reason  Age   From          Message
  ----  -----  --  --  -----
  Normal Scheduled  54s  default-scheduler  Successfully assigned default/nginx to node01
  Normal Pulling   54s  kubelet, node01  Pulling image "nginx"
  Normal Pulled    48s  kubelet, node01  Successfully pulled image "nginx"
  Normal Created   47s  kubelet, node01  Created container nginx
  Normal Started   46s  kubelet, node01  Started container nginx
```

# Pods ( com yml )- kubectl - Demo & Prática

<https://www.katacoda.com/courses/kubernetes/playground>



```
kubectl describe pod <nome_do_pod>
```

```
kubectl describe pod <nome_do_pod> -o wide
```

```
kubectl delete pod <nome_do_pod>
```

```
→ kubectl run nginxsemyml --image=nginx
```

```
→ kubectl get pods
```

```
apiVersion: v1
kind: Pod
metadata:
  name: nginxcomyml
  labels:
    app: meunginx
    tipo: servidorweb
    autor: <seu nome>
spec:
  containers:
    - name: nginxcomyml
      image: nginx
```

# Pods ( com yml )- kubectl - Demo & Prática

<https://www.katacoda.com/courses/kubernetes/playground>

## redispod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: rediscomyml
  labels:
    app: meuredis
    tipo: banco
    autor: <seu nome>
spec:
  containers:
    - name: rediscomyml
      image: redis
```

kubectl create -f <nome\_do\_arquivo>

kubectl delete pods -l tipo=servidorweb

## apachepod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: apachecomyml
  labels:
    app: meuapache
    tipo: servidorweb
    autor: <seu nome>
spec:
  containers:
    - name: apachecomyml
      image: httpd
```

ou      kubectl create -f nginxpod.yml -f apachepod.yml  
         -f redispod.yml

# Pods ( com yml )- kubectl - Demo & Prática

<https://www.katacoda.com/courses/kubernetes/playground>

Arquivo: **pod2containers.yml**

**Contaiier 1:**

- **Image:** nginx

**Container 2:**

- **Image:** fluentd

```
apiVersion: v1
kind: Pod
metadata:
  name: rediscomyml
  labels:
    app: meureredis
    tipo: banco
    autor: <seu nome>
spec:
  containers:
    - name: rediscomyml
      image: redis
    - name: algum-nome
      image: alguma-imagem
```



# Pods - Quizzzzzz



```
brew install --cask google-cloud-sdk
```



```
curl -O  
https://dl.google.com/dl/cloudsdk/channels/rapid/downlo  
ads/google-cloud-sdk-323.0.0-linux-x86_64.tar.gz
```

```
tar -xf google-cloud-sdk-323.0.0-linux-x86_64.tar.gz
```

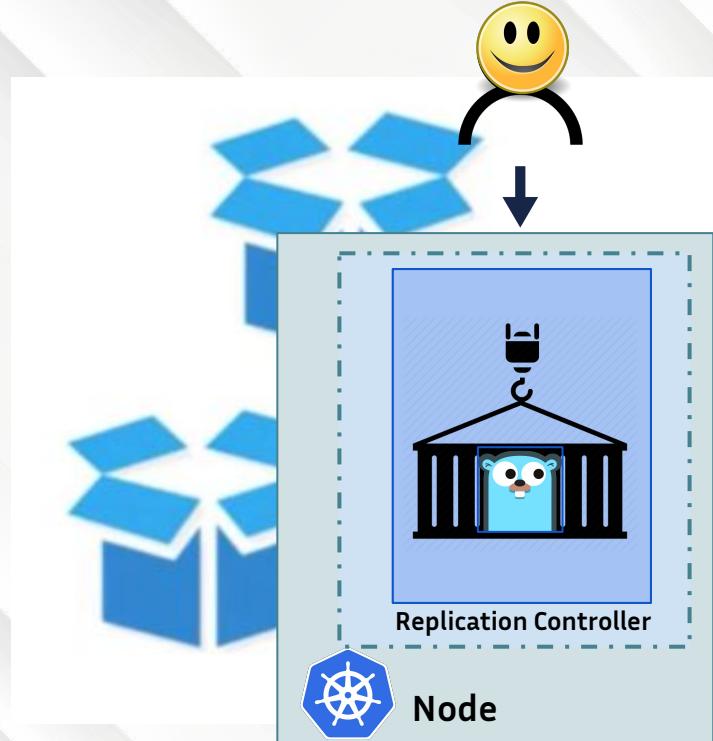
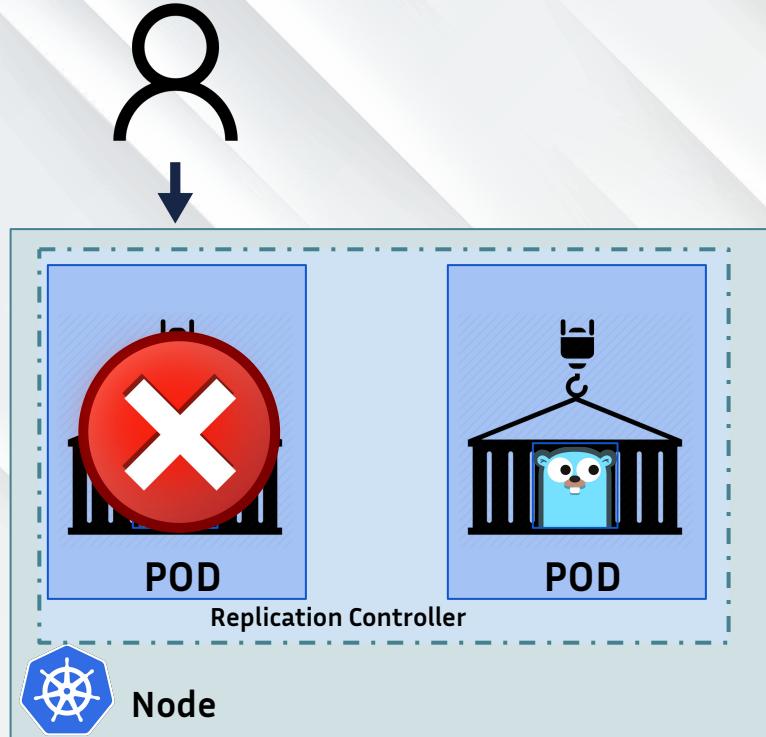
```
./google-cloud-sdk/install.sh
```



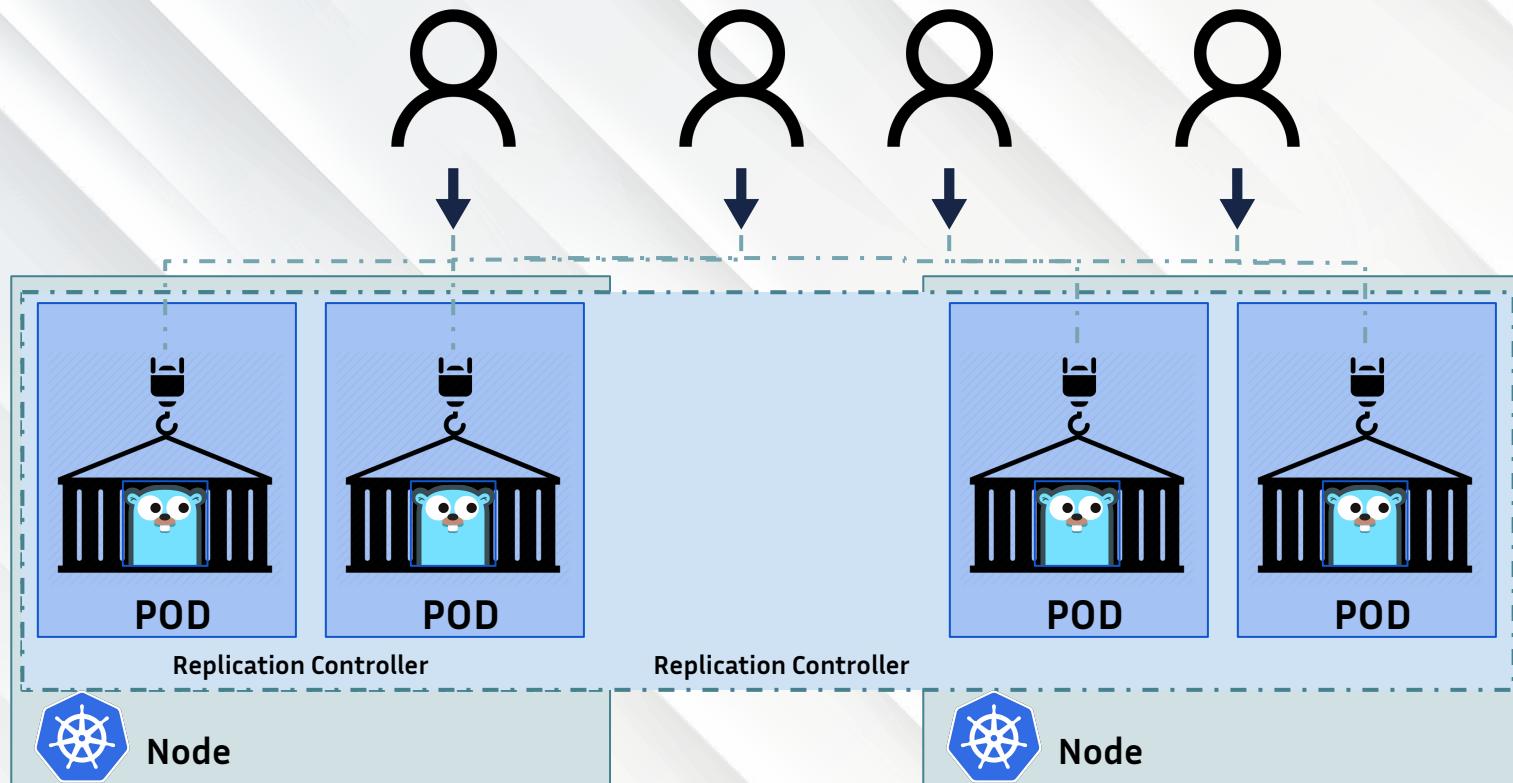
```
(New-Object  
Net.WebClient).DownloadFile("https://dl.google.com/  
dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.  
exe", "$env:Temp\GoogleCloudSDKInstaller.exe")  
  
& $env:Temp\GoogleCloudSDKInstaller.exe
```

dsdk/channels/rapid/Go

# Kubernetes Controllers - Replication Controller



# Load Balancing & Scaling



# Replication Controller & Replica Set

**Replication Controller**

**Replica Set**

# Replication Controller

rc-teste.yml

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: meuapp-rc
  labels:
    app: meuapp
    tipo: front-end
spec:
  template:
    metadata:
      name: meuapp-pod
      labels:
        app: meuapp
        tipo: front-end
    spec:
      containers:
        - name: nginx-container
          image: nginx
    replicas: 3
```

pod-teste.yml

```
apiVersion: v1
kind: Pod
```

```
> kubectl create -f rc-teste.yml
```

```
Replicationcontroller "meuapp-rc" created
```

# Replication Controller

```
> kubectl get replicationcontroller
```

NAME	DESIRED	CURRENT	READY	AGE
Meuapp-rc	3	3	3	19s

```
> kubectl get pods
```

NAME	DESIRED	CURRENT	READY	AGE
Meuapp-rc-41vk9	3	3	3	19s
Meuapp-rc-23y4h	3	3	3	19s
Meuapp-rc-4hgv7	3	3	3	19s

# Replication Controller - Prática

```
> gcloud container clusters get-credentials possible-sun-222819-gke --region us-central1
```

<seunome>-app-rc.yml

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: <nome>-app-rc
  labels:
    app: <nome>-app
    tipo: web
spec:
  template:
    metadata:
      name: <nome>-app-pod
      labels:
        app: <nome>-app
        tipo: web
    spec:
      containers:
        - name: <nome>-nginx-container
          image: nginx
replicas: 3
```

```
> kubectl create -f <seunome>-app-rc.yml
```

```
> kubectl get replicationcontroller
```



# ReplicaSet

meuapp-replicaset.yml

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: meuapp-replicaset
  labels:
    app: meuapp
    tipo: front-end
spec:
  template:
    metadata:
      name: meuapp-pod
      labels:
        app: meuapp
        tipo: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
replicas: 4
```

meupod.yml

```
apiVersion: v1
kind: Pod
```

# ReplicaSet

```
> kubectl create -f  
meuapp-replicaset.yml
```

```
Replicaset "meuapp-replicaset" created
```

```
> kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
replicaset	3	3	3	19s

```
> kubectl get pods
```

NAME	DESIRED	CURRENT	READY	AGE
repset-41v9	3	3	3	19s
repaset-234h	3	3	3	19s
recaset-4hv7	3	3	3	19s

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: meuapp-replicaset
  labels:
    app: meuapp
    tipo: front-end
spec:
  template:
    metadata:
      name: meuapp-pod
      labels:
        app: meuapp
        tipo: front-end
    spec:
      containers:
        - name: nginx-container
          image: nginx
      replicas: 3
      selector:
        matchLabels:
          type: front-end
```



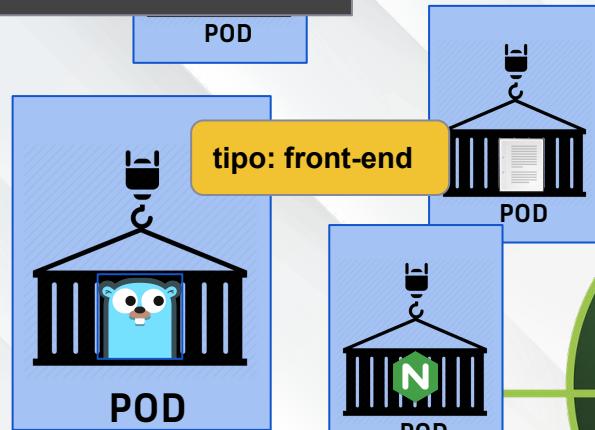
# Labels and Selectors

replicaset.yml

```
selector:  
  matchLabels:  
    tipo: front-end
```



POD



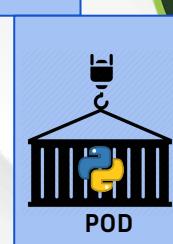
POD



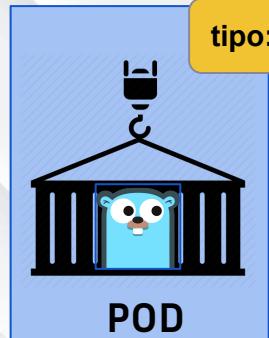
POD



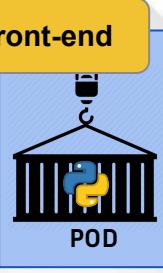
POD



POD



tipo: front-end



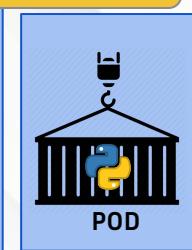
POD



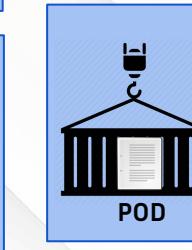
POD



POD



POD



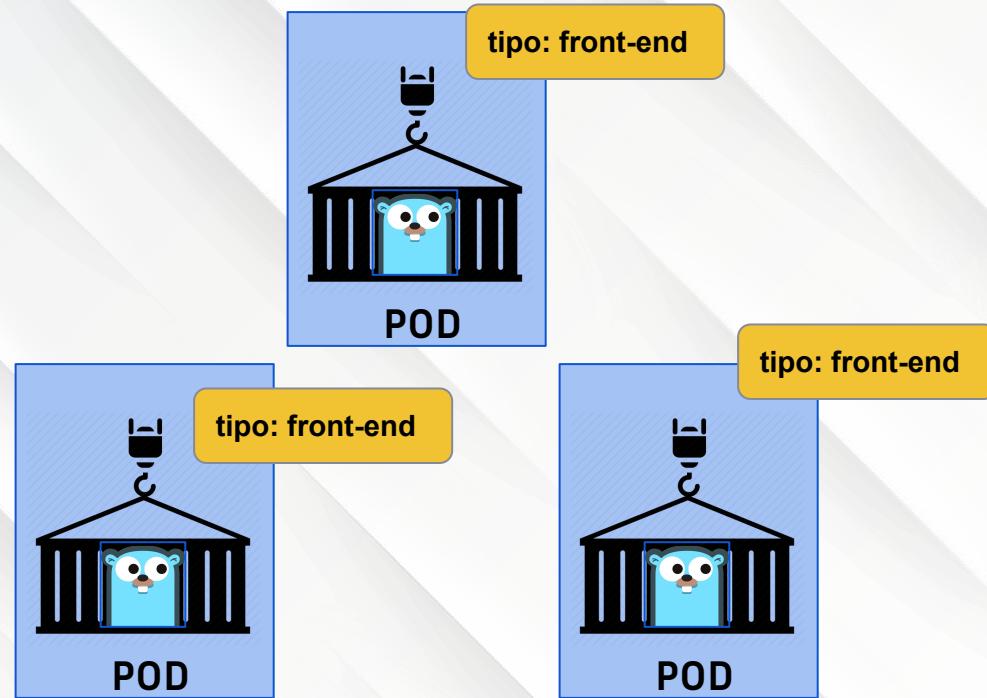
POD

metadata:  
 name: meuapp-pod  
 labels:  
 tipo: front-end



## replicaset.yml

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: meuapp-replicaset
  labels:
    app: meuapp
    type: front-end
spec:
  template:
    metadata:
      name: meu-app-pod
      labels:
        app: meu-app
        tipo: front-end
    spec:
      containers:
        - name: nginx-container
          image: nginx
  replicas: 3
  selector:
    matchLabels:
      tipo: front-end
```



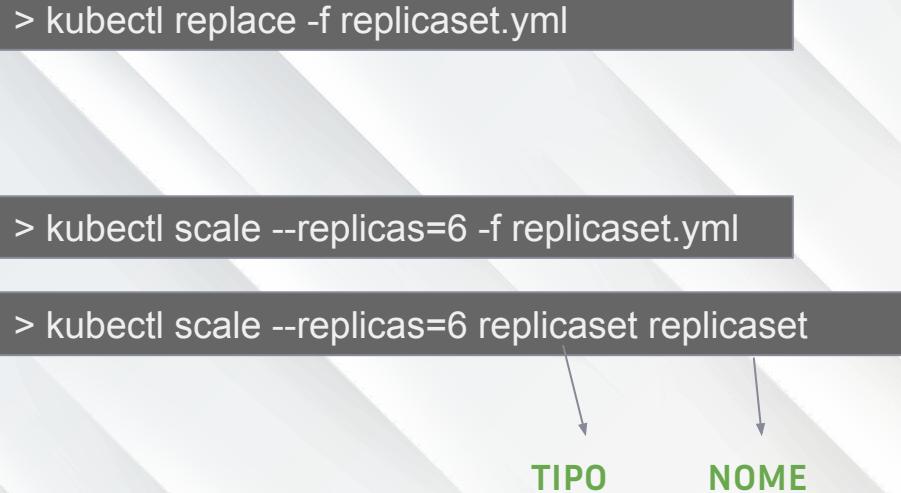
# Scale

```
> kubectl replace -f replicaset.yml
```

```
> kubectl scale --replicas=6 -f replicaset.yml
```

```
> kubectl scale --replicas=6 replicaset replicaset
```

TIPO      NOME



The diagram consists of three separate command examples from the left side, each enclosed in a dark grey box. Arrows from each example point downwards towards the 'TIPO' and 'NOME' labels located in the center of the slide.

replicaset.yml

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: meuapp-replicaset
  labels:
    app: meuapp
    type: front-end
spec:
  template:
    metadata:
      name: meu-app-pod
      labels:
        app: meu-app
        tipo: front-end
    spec:
      containers:
        - name: nginx-container
          image: nginx
  replicas: 6
  selector:
    matchLabels:
      tipo: front-end
```

# ReplicaSet - Prática

**Conectando-se ao cluster:**

```
> gcloud container clusters get-credentials possible-sun-222819-gke --region us-central1
```

**Testando o cluster:**

```
> kubectl get nodes
```



```
<nome>-replicaset.yml
```

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: <nome>-replicaset
  labels:
    app: <nome>-meuapp
    type: <nome>-web
spec:
  template:
    metadata:
      name: <nome>-app-pod
      labels:
        app: <nome>-app
        tipo: <nome>-web
    spec:
      containers:
        - name: nginx-container
          image: nginx
  replicas: 3
  selector:
    matchLabels:
      tipo: <nome>-web
```

## Criando o replica set:      **ReplicaSet - Prática**

```
> kubectl create -f <nome>-replicaset.yml
```

### Listando replicaset:

```
> kubectl get replicaset
```

### Listando pods:

```
> kubectl get pods
```

### Listando tudo:

```
> kubectl get all
```

### Outra opção:

```
> kubectl get rs
```

```
> kubectl replace -f <nome>-replicaset.yml
```

```
> kubectl scale --replicas=6 -f <nome>-replicaset.yml
```

```
> kubectl scale --replicas=6 replicaset <noem>-replicaset.yml
```

## <nome>-pod.yml

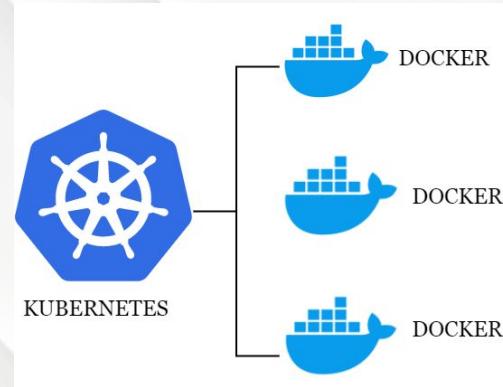
```
apiVersion: v1
kind: Pod
metadata:
  name: <nome>-app-pod
  labels:
    app: <nome>-app
    tipo: <nome>-web
    autor: <seu nome>
spec:
  containers:
    - name: nginx-container
      image: nginx
```



# Kind - Instalação

[shorturl.at/avMZ3](http://shorturl.at/avMZ3)

Kubernetes in Docker



# Kind - Setup

Listar containers rodando no sistema:

```
> docker ps
```

Criando o cluster:

```
> kind create cluster --name teste
```

Verifiquem informações do cluster:

```
> kubectl cluster-info --context kind-teste
```

Verifiquem se estão no contexto correto:

```
> kubectl config get-contexts
```

Listar clusters:

```
> kind get clusters
```

Deletar clusters:

```
> kind delete cluster --name teste
```

# Kind - Setup

Definições do cluster:

```
> cluster-teste.yml
```

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
- role: worker
- role: worker
```

Criando o cluster:

```
> kind create cluster --name teste --config cluster-teste.yml
```

Listar containers rodando no sistema:

```
> docker ps
```

Verifiquem informações do cluster:

```
> kubectl cluster-info --context kind-teste
```

```
$ kind create cluster --name teste --config cluster-teste.yml
Creating cluster "teste" ...
✓ Ensuring node image (kindest/node:v1.19.1) 
✓ Preparing nodes 
✓ Writing configuration 
✓ Starting control-plane 
✓ Installing CNI 
✓ Installing StorageClass 
✓ Joining worker nodes 
Set kubectl context to "kind-teste"
You can now use your cluster with:

kubectl cluster-info --context kind-teste

Have a nice day! 
```

Listar clusters:

```
> kind get clusters
```

# Kind - Prática

nginx-deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.16.1
          ports:
            - containerPort: 80
```

nginx-service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  labels:
    app: nginx
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30225
  selector:
    app: nginx
```

# Kind - Prática

Criando o deployment:

```
> kubectl create -f nginx-deployment.yml
```

Criando o service:

```
> kubectl create -f nginx-service.yml
```

Vendo tudo:

```
> kubectl get all
```

Mapeando porta:

```
> kubectl port-forward service/nginx-service 8000:80
```

Forwarding from 127.0.0.1:8000 -> 80

Forwarding from [::1]:8000 -> 80

Handling connection for 8000

Handling connection for 8000



```
 127.0.0.1:8000
```

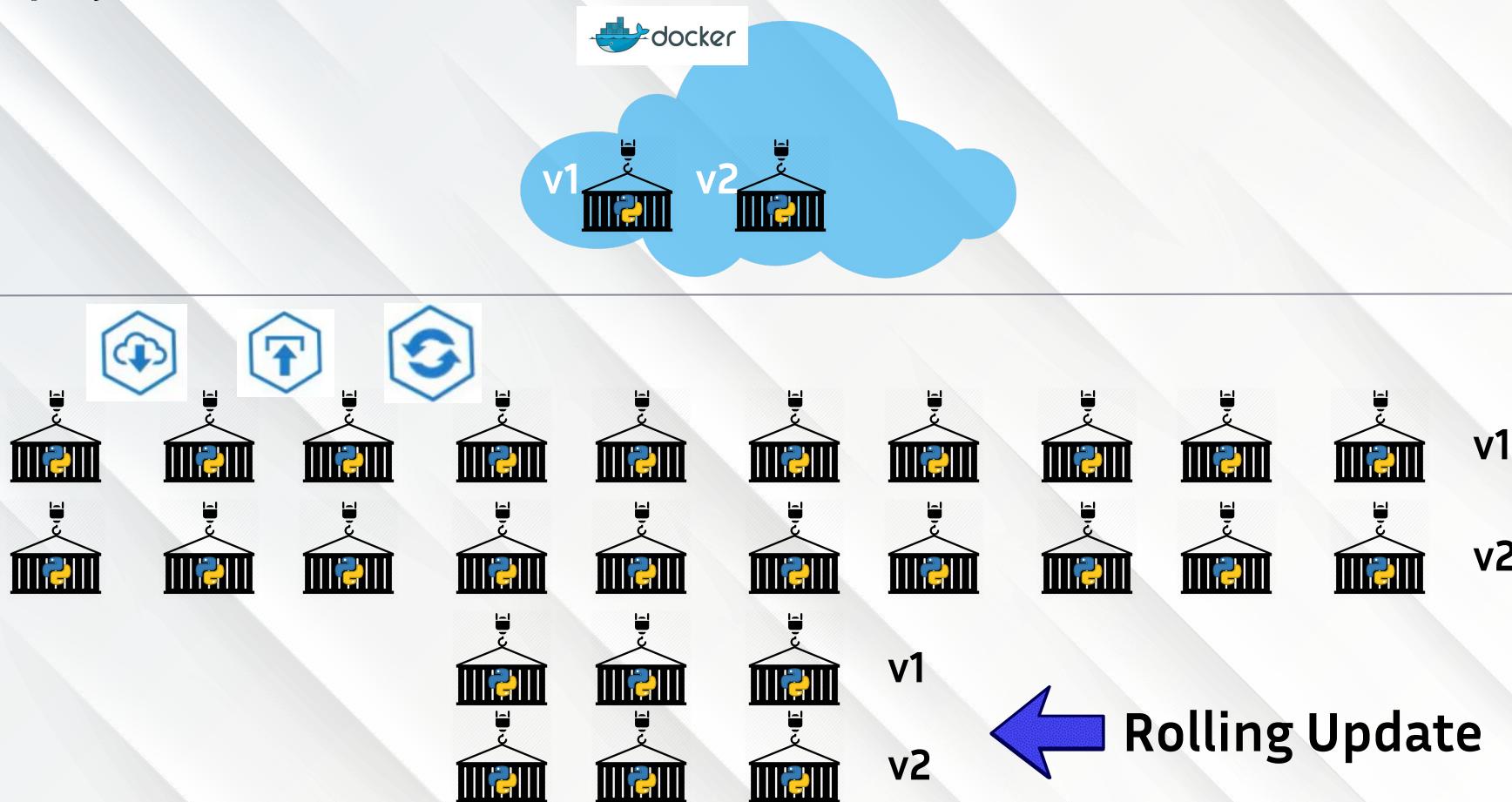
## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

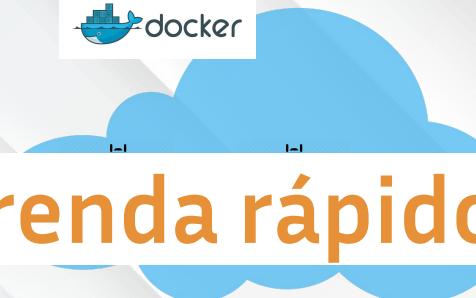
For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

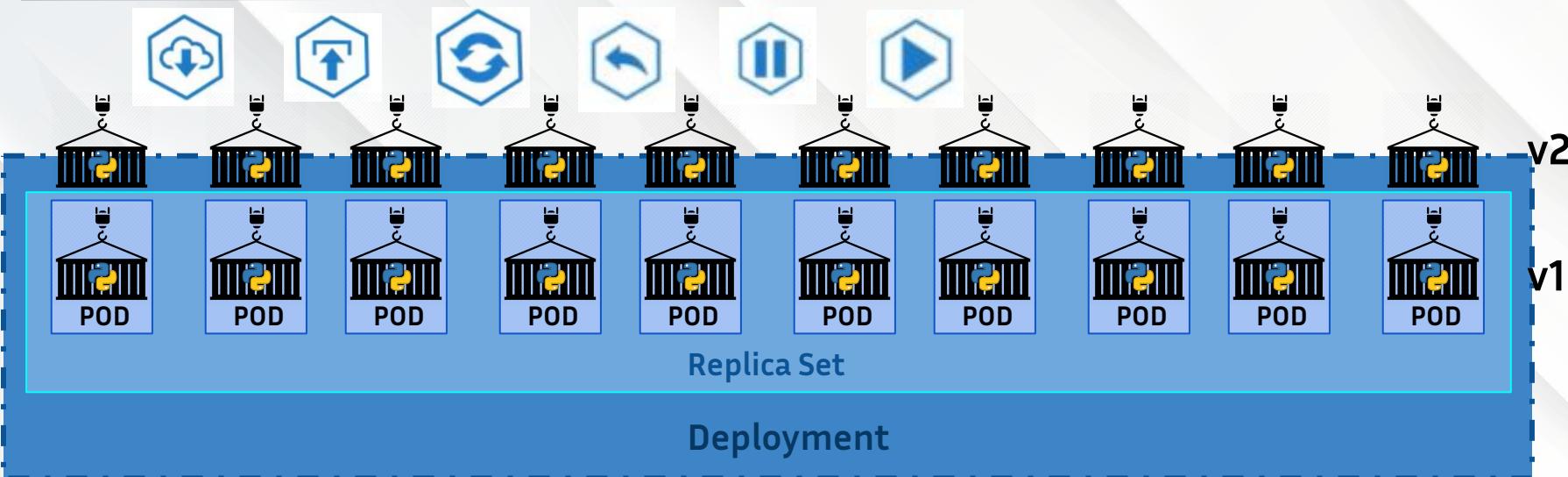
# Deployments



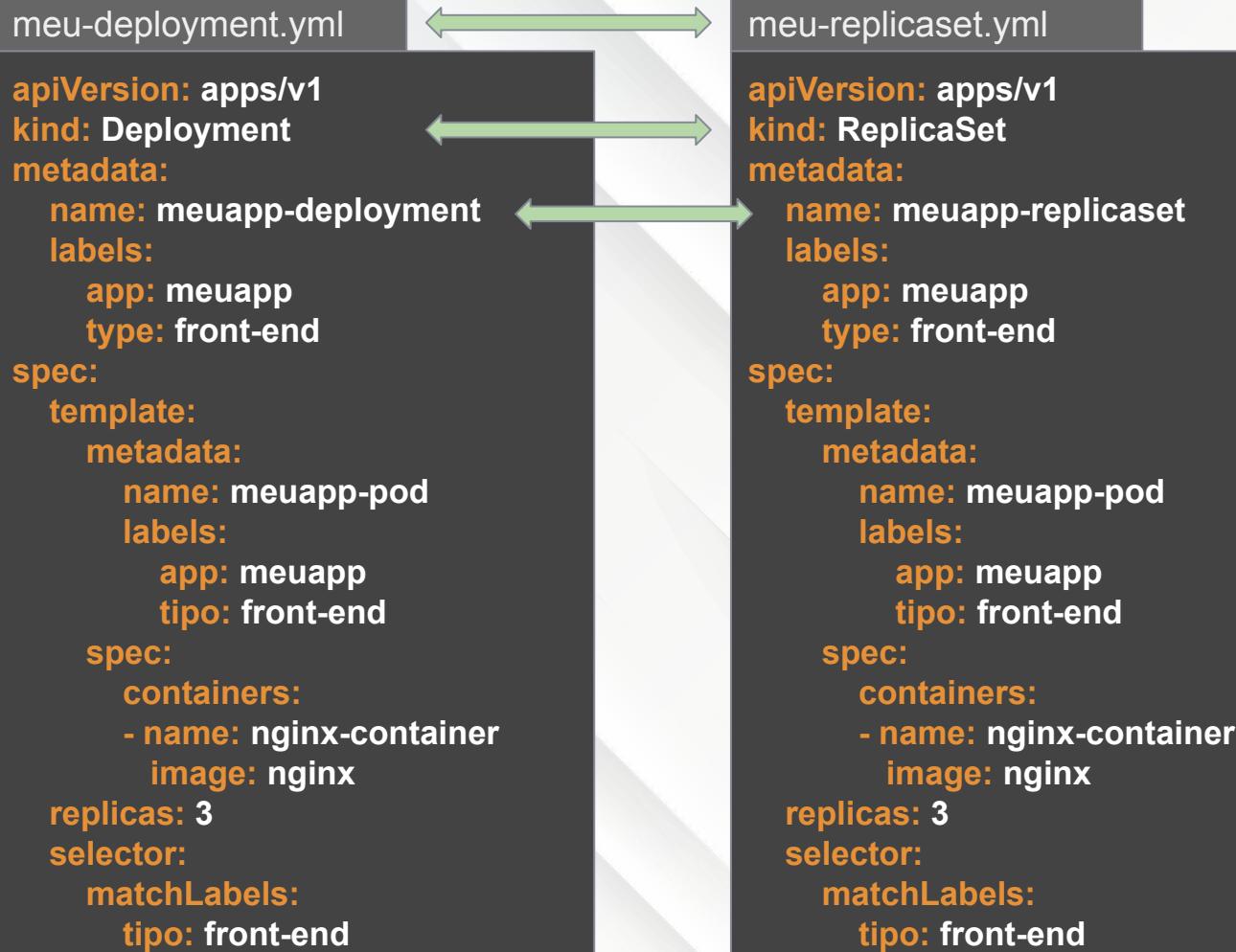
# Deployments



Erre rápido, aprenda rápido e corrija rápido.



# Deployments



# Deployments

Criando nosso deployment:

```
> kubectl create -f meu-deployment.yml
```

```
Deployment "meu-deployment" created
```

```
> kubectl get deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
meuapp-deployment	3/3	3	3	26s

```
> kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
meuapp-deployment-74bdc9f869	3	3	3	17s

```
> kubectl get all
```

```
meu-deployment.yml
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: meuapp-deployment
```

```
  labels:
```

```
    app: meuapp
```

```
    type: front-end
```

```
spec:
```

```
  template:
```

```
    metadata:
```

```
      name: meuapp-pod
```

```
    labels:
```

```
      app: meuapp
```

```
      tipo: front-end
```

```
spec:
```

```
  containers:
```

```
  - name: nginx-container  
    image: nginx
```

```
replicas: 3
```

```
selector:
```

```
  matchLabels:
```

```
    tipo: front-end
```

# Imperativo x Declarativo



<code>



```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      ports:
        - containerPort: 80
          name: http
```



# Kubernetes de forma Imperativa

```
> kubectl run --image=nginx nginx
```

```
pod/nginx created
```

```
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx	1/1	Running	0	45s

- Com a forma imperativa podemos criar arquivos yaml declarativos

IMPERATIVA



DECLARATIVA

# Kubernetes de forma Imperativa

## dry-run

```
> kubectl run --image=nginx nginx --dry-run=client
```

```
pod/nginx created
```

```
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx	1/1	Running	0	45s

```
> kubectl run --imagee=nginx nginx --dry-run=client
```

```
Error: unknown flag: --imagee
```

```
See 'kubectl run --help' for usage.
```

# Kubernetes de forma Imperativa

```
> kubectl run --image=nginx nginx --dry-run=client
```

```
> kubectl run --image=nginx nginx --dry-run=client -o json
```

```
> kubectl run --image=nginx nginx --dry-run=client -o yaml
```



```
> kubectl run --image=nginx nginx --dry-run=client -o yaml > nginx-pod-novo.yml
```

```
> kubectl create -f nginx-pod-novo.yml
```

```
> mkdir novos-arquivos
```

```
> cd novos-arquivos
```

# Kubernetes de forma Imperativa

Criando um arquivo para um pod fluentd:

```
> kubectl run --image=fluentd fluentd --dry-run=client -o yaml > pod-fluentd.yml
```

Criando um deployment nginx com 3 réplicas:

```
> kubectl create deployment --image=nginx nginx --replicas=3 --dry-run=client -o yaml > deploy-nginx.yml
```

```
> kubectl create -f .
```

```
deployment.apps/nginx created  
pod/fluentd created
```

# Deployments

> kubectl create -f meu-deployment.yml



meu-deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: MeuApp
  labels:
    app: meuapp
    tipo: front-end
spec:
  template:
    metadata:
      name: meuapp-pod
      labels:
        app: meuapp
        tipo: front-end
    spec:
      containers:
        - name: nginx-container
          image: nginx
  replicas: 2
  selector:
    matchLabels:
      tipo: front-end
```

# Deployments

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx
    name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
      resources: {}
status: {}
```