ESCOLA SUPERIOR DE REDES

# Orquestração de Containers

## Arquitetura e conceitos do Kubernetes

# Tópicos abordados

- Vantagens da orquestração de containers
- Arquitetura do Kubernetes
- Componentes do *control plane*
- Componentes nos *nodes*
- *Pods*
- *ReplicaSets*

# Tópicos abordados

- Deployments
- Services
- Namespaces
- Comandos imperativos e declarativos
- Interações com resolução DNS

# Como vimos anteriormente...

Containers são artefatos em nível de aplicação

Viabilizam processos de CI/CD

Rápido deployment de serviços

Portabilidade entre ambientes *dev* x *prod*

# Por outro lado, temos alguns desafios

- Gerência de microserviços
- Escalonamento de recursos
- Gerência de rede
- Segurança

# Vantagens da orquestração de containers


Ambiente de desenvolvimento

Ambiente de produção

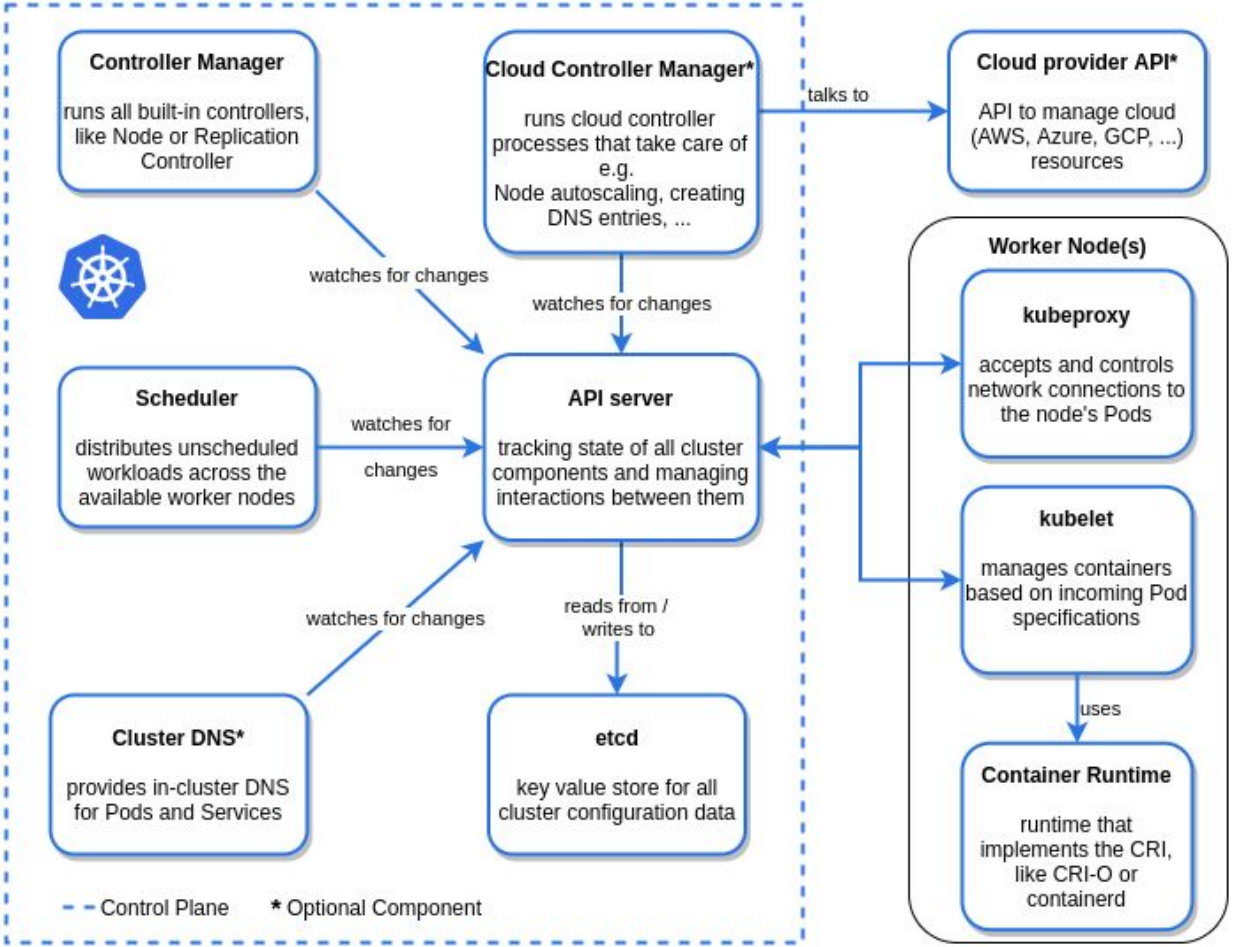
# Histórico do Kubernetes

Originalmente desenvolvido por engenheiros do Google e lançado em 2014, o Kubernetes foi tornado *open source* e doado para a recém-criada CNCF em 2015

O Kubernetes, grego para "*capitão*" ou "*piloto*", é implementado na linguagem Go

https://blog.risingstack.com/the-history-of-kubernetes

# Kubernetes Architecture




ESCOLA SUPERIOR DE REDES

# Master Node: kube-system

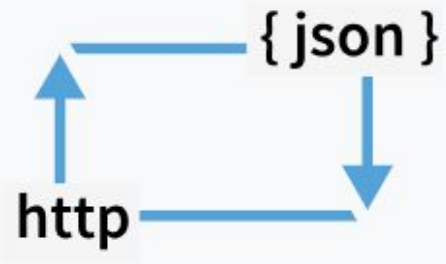
vagrant@s2-master-1: ~

```
root@s2-master-1:~# kubectl get pod -n kube-system
NAME                                     READY   STATUS    RESTARTS   AGE
coredns-558bd4d5db-l6nsj                 1/1     Running   0          16m
coredns-558bd4d5db-q6n8c                 1/1     Running   0          16m
etcd-s2-master-1                         1/1     Running   0          17m
kube-apiserver-s2-master-1               1/1     Running   0          17m
kube-controller-manager-s2-master-1      1/1     Running   0          17m
kube-proxy-lv49b                         1/1     Running   0          13m
kube-proxy-qqgnj                         1/1     Running   0          16m
kube-scheduler-s2-master-1               1/1     Running   0          17m
weave-net-jr8qb                          2/2     Running   1          16m
weave-net-w28rh                          2/2     Running   1          13m
root@s2-master-1:~#
```

# *etcd*

## Simple interface

Read and write values using standard HTTP tools, such as curl




## Key-value storage

Store data in hierarchically organized directories, as in a standard filesystem

```
/config
└ /database
/feature-flags
├ /verbose-logging
└ /redesign
```

## Watch for changes

Watch specific keys or directories for changes and react to changes in values


https://etcd.io

# *etcd* no Kubernetes

Usado para armazenar configurações do *cluster*, como:

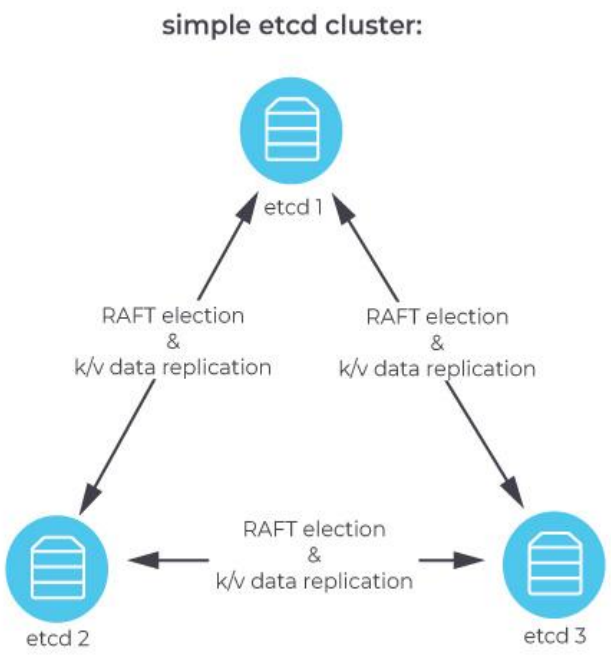
- CONFIGURAÇÃO
- METADADOS
- ESTADO

- *Nodes*
- *Accounts*
- *Pods*
- *Roles*
- *Configs*
- *RoleBindings*
- *Secrets*
- *Etc.*

# *etcd* em produção

O etcd utiliza o protocolo RAFT para construir consenso e garantir alta disponibilidade

https://raft.github.io/



# *kube-apiserver*

Faz a gestão e coordenação dos demais componentes do *cluster* Kubernetes

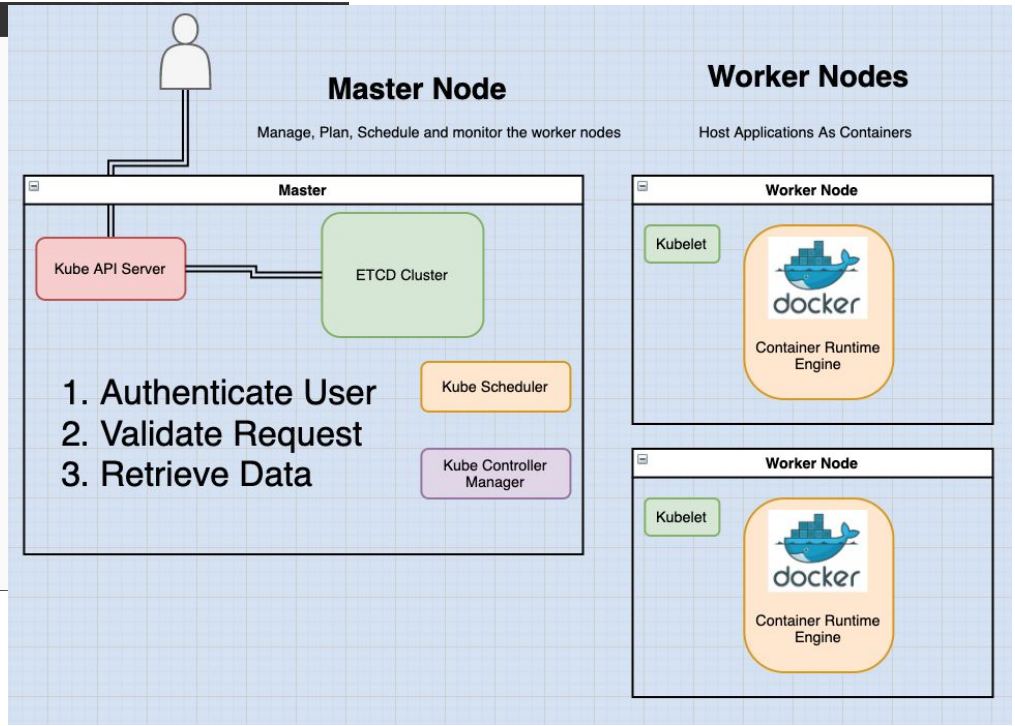
Componente responsável pela **exposição** da API de controle do Kubernetes

Ponto de **interação** entre o usuário e o *cluster* via comandos *kubectl* ou requisições POST

Comandos/requisições devem ser autenticados e validados antes de seu processamento

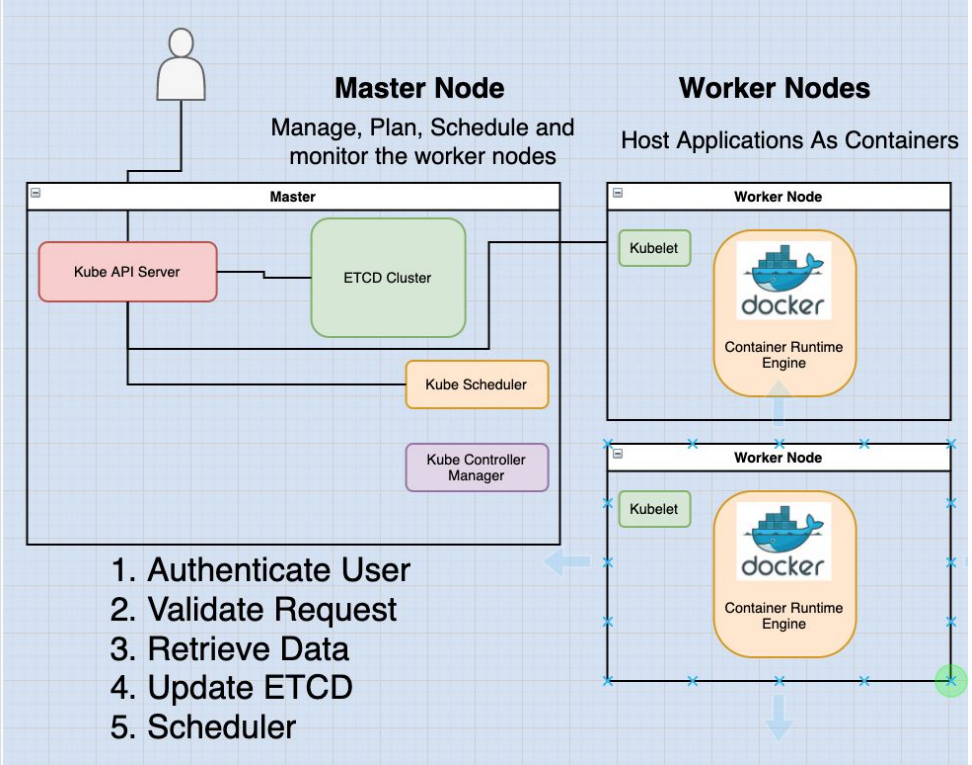
# *kube-apiserver*: consulta

```
vagrant@s2-master-1: ~
root@s2-master-1:~# kubectl get node
NAME          STATUS   ROLES                  AGE   VERSION
s2-master-1   Ready    control-plane,master   21m   v1.21.1
s2-node-1     Ready    <none>                 17m   v1.21.1
root@s2-master-1:~#
```



# *kube-apiserver*: modificação

```
vagrant@s2-master-1: ~
root@s2-master-1:~# kubectl run nginx --image=nginx
pod/nginx created
root@s2-master-1:~#
root@s2-master-1:~#
root@s2-master-1:~# kubectl get pod
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          10s
root@s2-master-1:~#
```



# *kube-apiserver*: interação

Note que o *kube-apiserver* é o único componente que se comunica com o usuário e interage com o *etcd*

Os demais componentes comunicam-se com o *kube-apiserver* para efetivar mudanças no *cluster*

# *kube-controller-manager*

Meta-componente que executa outros processos **controladores**

Embora logicamente distintos, esses controladores são **agrupados**

**Observa** e **remedia** o estado de componentes do *cluster*

Visa trazer o *cluster* para o **estado** desejado

# *kube-controller-manager*: alguns tipos de controladores

*Node controller*: Monitora e responde quando nós tornam-se indisponíveis

*Job controller:* Monitora objetos do tipo *job* para eventos agendados, criando *pods* para sua execução

*Endpoint controller:* Popula objetos do tipo *endpoint*, como serviços e *pods*

*Service account & token controller*: Cria contas padrão e acesso via API para novos *namespaces*

# *kube-scheduler*

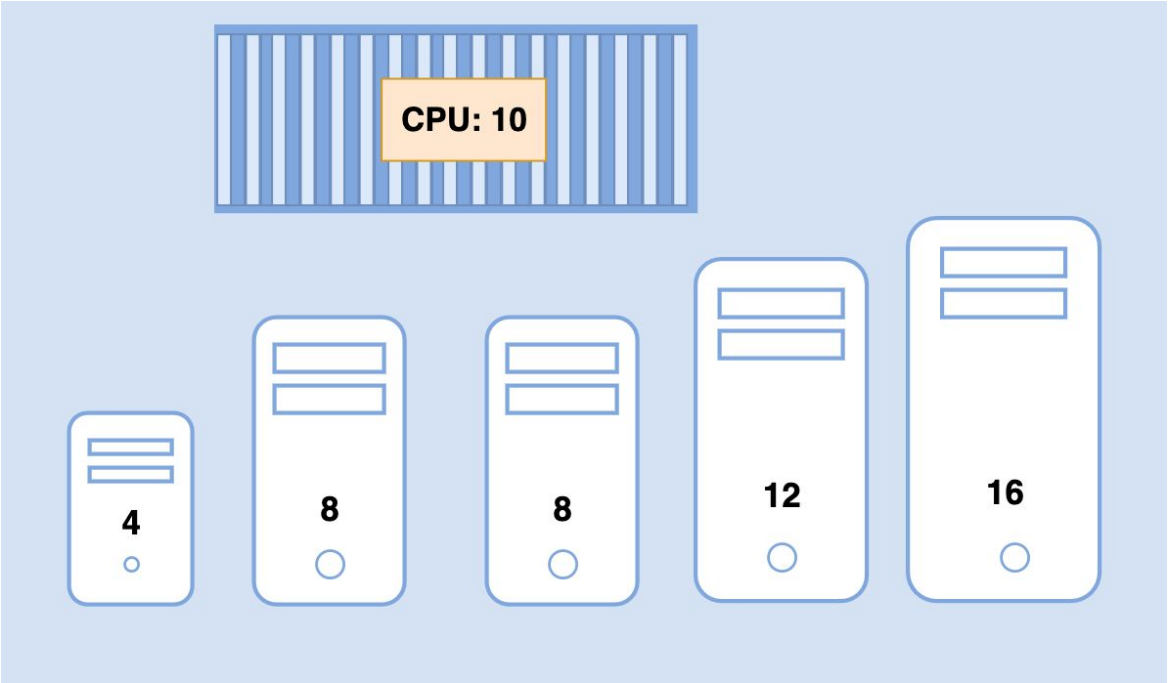
Monitora os recursos do cluster escalonando um pod em um **node**

**Fatores de decisão** são levados em consideração no agendamento

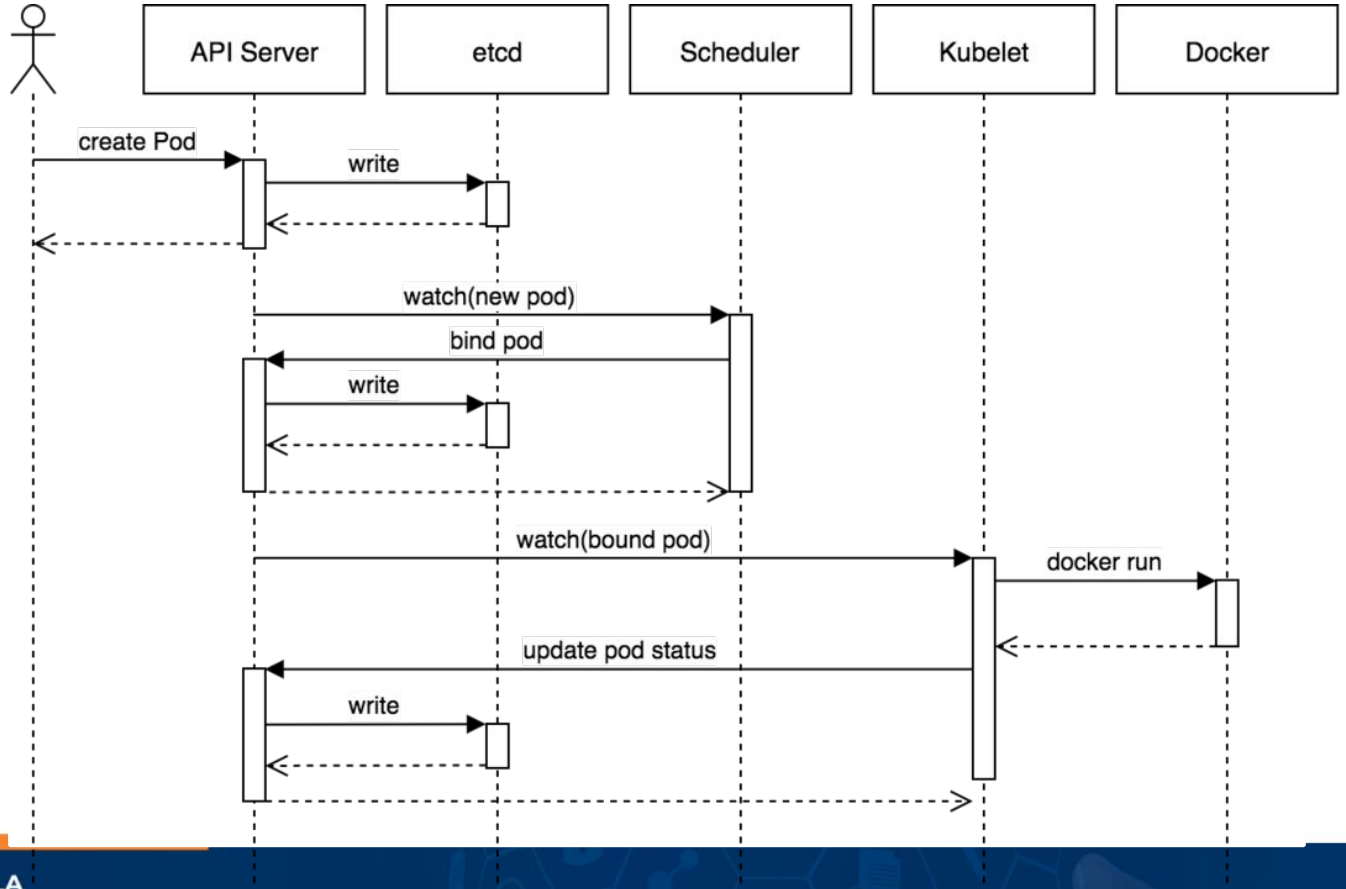
Recursos como **CPU** e **memória** são relevantes

Considera aspectos de **afinidade** e **anti-afinidade**

# *kube-scheduler*: considerando recursos mínimos



# Interação entre componentes do control-plane



# Componentes nos *nodes*

*kubelet*

*kube-proxy*

*Container runtime*

# *kubelet*

Agente que opera em **cada um** dos *nodes* do *cluster*

**Garante** que *containers* estão executando em um *pod*

Recebe um conjunto de **PodSpecs** e visa atendê-los

**Não gerencia** containers que não foram criados pelo Kubernetes

# *kubelet*: implementado como serviço ou como container

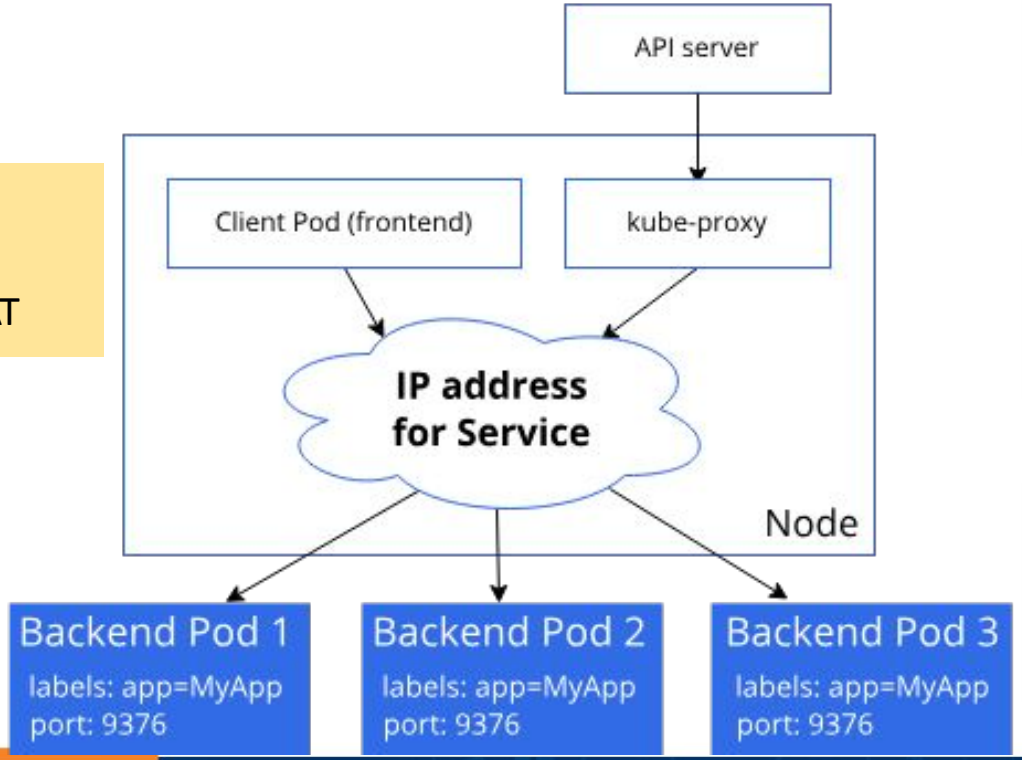
vagrant@s2-master-1: ~

```
root@s2-master-1:~# systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
   Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/kubelet.service.d
           └─10-kubeadm.conf
   Active: active (running) since Mon 2021-06-21 19:11:58 UTC; 58min ago
     Docs: https://kubernetes.io/docs/home/
 Main PID: 18151 (kubelet)
    Tasks: 19 (limit: 2356)
   Memory: 73.7M
   CGroup: /system.slice/kubelet.service
           └─18151 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf

Jun 21 19:15:56 s2-master-1 kubelet[18151]: I0621 19:15:56.424366   18151 docker_sandbox.go:401] "
Jun 21 19:15:56 s2-master-1 kubelet[18151]: I0621 19:15:56.426461   18151 pod_container_deletor.go
Jun 21 19:15:56 s2-master-1 kubelet[18151]: I0621 19:15:56.429220   18151 cni.go:333] "CNI failed
Jun 21 19:15:56 s2-master-1 kubelet[18151]: I0621 19:15:56.429987   18151 docker_sandbox.go:401] "
Jun 21 19:15:56 s2-master-1 kubelet[18151]: I0621 19:15:56.431370   18151 pod_container_deletor.go
Jun 21 19:15:56 s2-master-1 kubelet[18151]: I0621 19:15:56.432247   18151 cni.go:333] "CNI failed
Jun 21 19:15:56 s2-master-1 kubelet[18151]: weave-cni: Delete: no addresses for 4b03b8550b7f31beb2
Jun 21 19:15:56 s2-master-1 kubelet[18151]: weave-cni: Delete: no addresses for 16ed912b3259023bd2
Jun 21 19:15:58 s2-master-1 kubelet[18151]: I0621 19:15:58.484571   18151 pod_container_deletor.go
Jun 21 19:15:58 s2-master-1 kubelet[18151]: I0621 19:15:58.496850   18151 pod_container_deletor.go
lines 1-22/22 (END)
```

# *kube-proxy*

**IPTABLES**
-> TABELA NAT
-> TARGET DNAT




**Proxy** de rede que opera em cada um dos *nodes* do *cluster*

Implementa parte do conceito de **serviços**

Trabalha junto com as network policies, configurando regras de firewall

# *Container runtimes*

Software responsável por efetivamente **executar containers**

Compatível com implementações da CRI (*Container Runtime Interface*)

docker

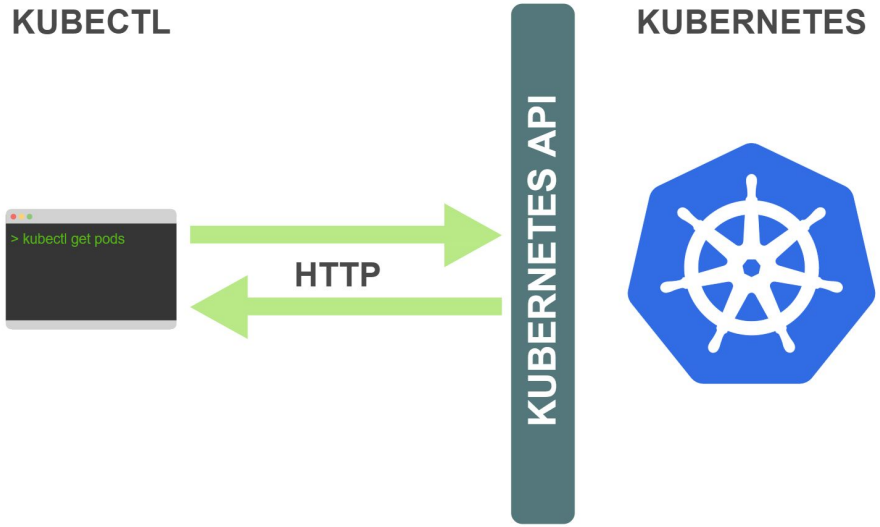
containerd

cri-o

https://kubernetes.io/blog/2020/12/02/dont-panic-kubernetes-and-docker/
https://kubernetes.io/docs/tasks/administer-cluster/migrating-from-dockershim/migrate-dockershim-dockerd/

# *kubectl*

Do lado do usuário, a interação com o *cluster* pode ser feita através do comando *kubectl*:



# E agora?

O que podemos fazer com isso tudo?

# *Pods*

**Menor** unidade computacional que pode ser criada e gerenciada no Kubernetes

Grupo de **um ou mais** containers com armazenamento e rede compartilhados

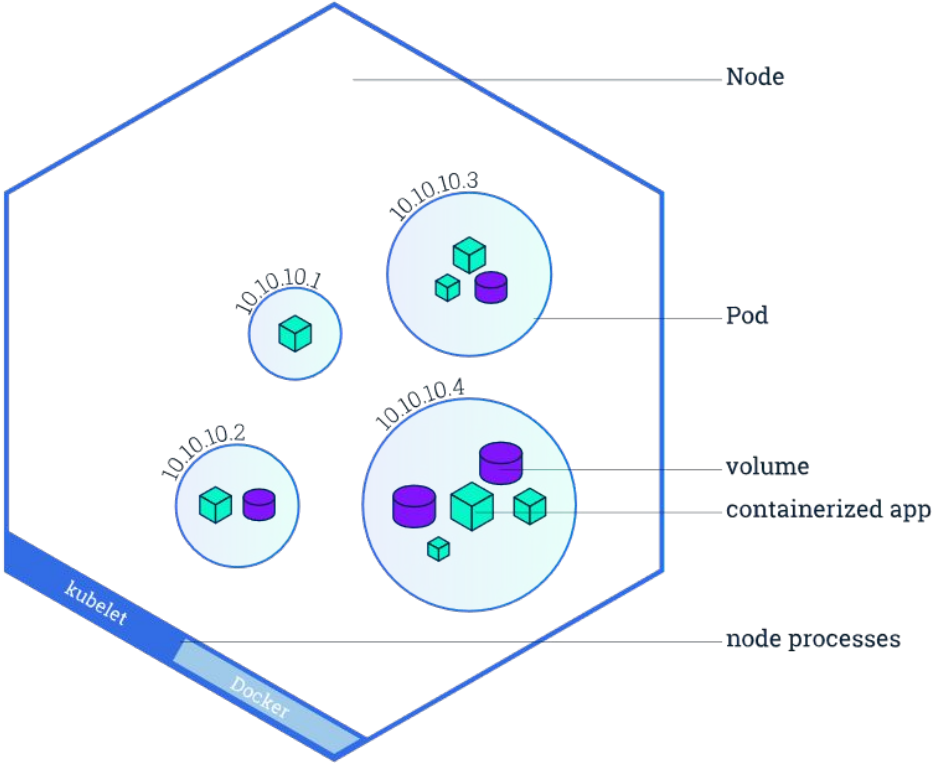
Em baixo nível, conjunto de Linux **namespaces** e **cgroups**

**Semelhante** conceitualmente a um grupo de containers Docker

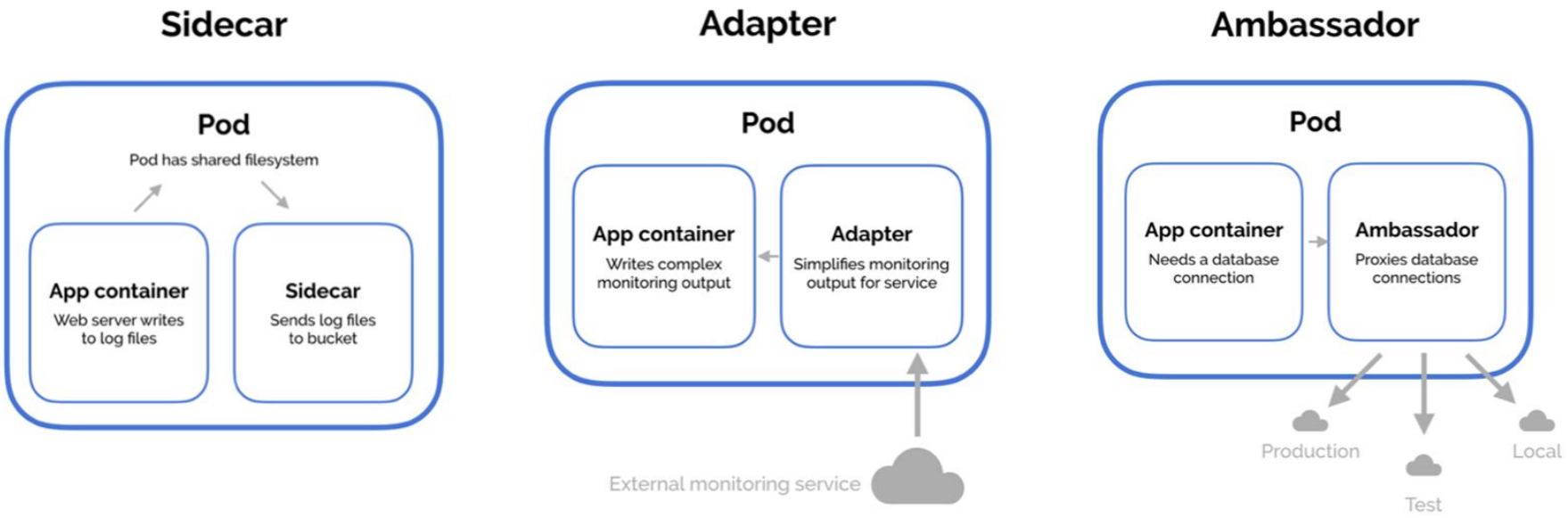
# PODs

Há uma relação **um-para-muitos** entre *pods* e *containers*

A **escalabilidade** de *pods* pode ser feita criando-se mais instâncias



# Multi-container Pods



# Criando *Pods* interativamente

vagrant@s2-master-1: ~

```
root@s2-master-1:~# kubectl run myapp --image=nginx
pod/myapp created
root@s2-master-1:~#
root@s2-master-1:~#
root@s2-master-1:~# kubectl get pod
NAME    READY   STATUS              RESTARTS   AGE
myapp   0/1     ContainerCreating   0          2s
root@s2-master-1:~#
root@s2-master-1:~#
root@s2-master-1:~# kubectl get pod
NAME    READY   STATUS    RESTARTS   AGE
myapp   1/1     Running   0          13s
root@s2-master-1:~#
```

# Removendo *Pods*

vagrant@s2-master-1: ~

```
root@s2-master-1:~# kubectl get pod
NAME    READY   STATUS    RESTARTS   AGE
myapp   1/1     Running   0          99s
root@s2-master-1:~#
root@s2-master-1:~#
root@s2-master-1:~# kubectl delete pod myapp
pod "myapp" deleted
root@s2-master-1:~#
root@s2-master-1:~#
root@s2-master-1:~# kubectl get pod
No resources found in default namespace.
root@s2-master-1:~#
```

# Criando *Pods* e deployments via arquivos YAML

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: app
spec:
  containers:
  - image: alpine
    name: app
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: deploy-app
  name: deploy-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-app
  template:
    metadata:
      labels:
        app: deploy-app
    spec:
      containers:
        - name: nginx
          image: nginx:alpine
```

# E se eu não souber a sintaxe?

Consulte a documentação!

https://kubernetes.io/docs/concepts/workloads/pods/

https://faun.pub/understanding-the-kubernetes-manifest-e96d680f2a11

# PODs

**kubectl api-version |less**

```
admissionregistration.k8s.io/v1
apiextensions.k8s.io/v1
apiregistration.k8s.io/v1
apps/v1
…
```

**kubectl api-resources |less**

| NAME | SHORTNAMES | APIVERSION | NAMESPACED | KIND |
|---|---|---|---|---|
| bindings | | v1 | true | Binding |
| componentstatuses | cs | v1 | false | ComponentStatus |
| configmaps | cm | v1 | true | ConfigMap |
| pods | po | v1 | true | Pod |

…

**kubectl explain Pod**

# Outra opção: utilizando *dry-runs*

```
> k run app --image=alpine --dry-run=client -o yaml
```

```
> k create deployment deploy-app --image=nginx:alpine --dry-run=client -o yaml
```

# Visualizando informações sobre *Pods*

vagrant@s2-master-1: ~

```
root@s2-master-1:~# kubectl describe pod myapp
Name:         myapp
Namespace:    default
Priority:     0
Node:         s2-node-1/192.168.68.25
Start Time:   Mon, 21 Jun 2021 20:42:05 +0000
Labels:       class=test
Annotations:  <none>
Status:       Running
IP:           10.44.0.1
IPs:
  IP:  10.44.0.1
Containers:
  myapp:
    Container ID:   docker://155b38a42bbfc07a99e781c0a9b2395505ca8f3c264c55ba994a936e5bea7551
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:6d75c99af15565a301e48297fa2d121e15d80ad526f8369
c526324f0f7ccb750
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 21 Jun 2021 20:42:08 +0000
    Ready:          True
    Restart Count:  0
```

# Monitorando eventos de *Pods*

vagrant@s2-master-1: ~

```
root@s2-master-1:~# kubectl describe pod myapp | awk '/Events:/,0'
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  50s   default-scheduler  Successfully assigned default/myapp to s2-node-1
  Normal  Pulling    47s   kubelet            Pulling image "nginx"
  Normal  Pulled     44s   kubelet            Successfully pulled image "nginx" in 2.357215882s
  Normal  Created    44s   kubelet            Created container myapp
  Normal  Started    44s   kubelet            Started container myapp
root@s2-master-1:~#
```

# Alterando a configuração de *Pods*

```
> k edit deploy deploy-app
```

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"labels":{"a
loy-app","namespace":"dev"},"spec":{"replicas":1,"selector":{"matchLabels":{"app":"deploy-a
{"labels":{"app":"deploy-app"}},"spec":{"containers":[{"image":"nginx:alpine","name":"nginx
  creationTimestamp: "2023-10-23T00:34:33Z"
  generation: 1
  labels:
    app: deploy-app
  name: deploy-app
  namespace: dev
  resourceVersion: "33918"
  uid: 3b060d21-9285-4530-bc3b-480e89083539
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: deploy-app
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: deploy-app
```

# *ReplicaSet*

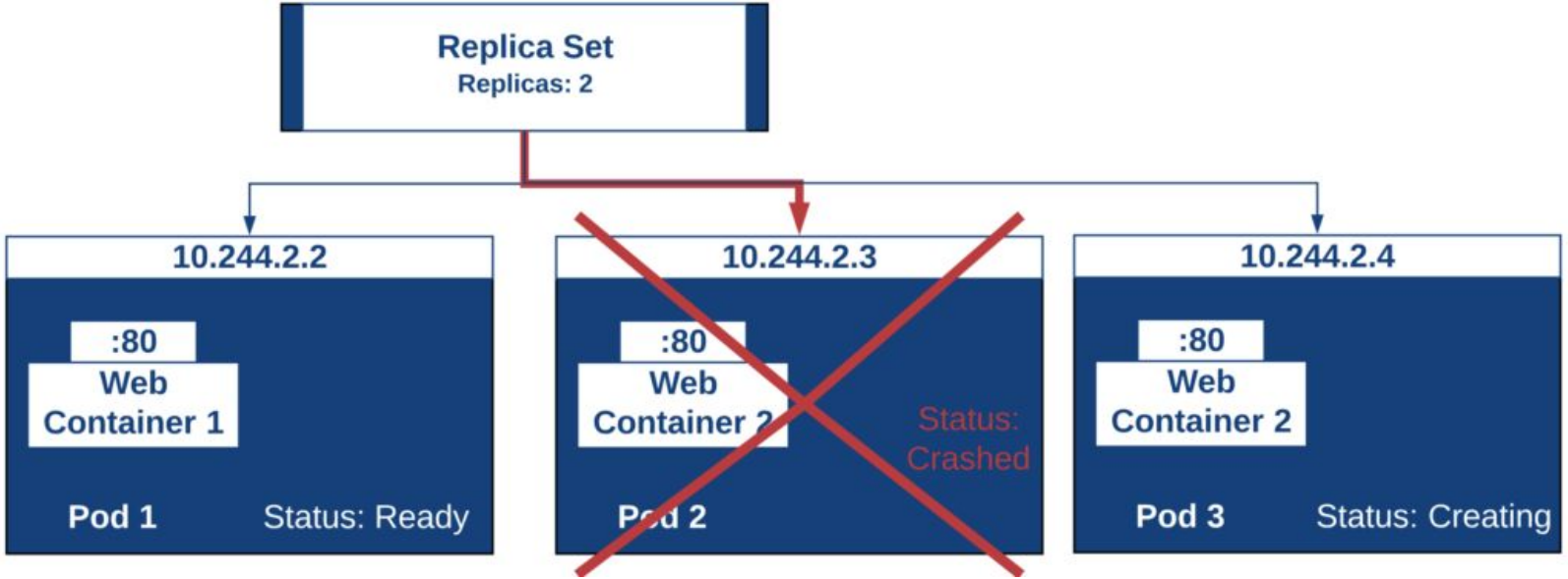
**Construto** cujo objetivo é manter um número estável de cópias de um determinado *pod*

Define *pods-alvo* através de **seletores**

**Adiciona** ou **remove** *pods* para atingir critérios especificados

Visa **garantir** alta disponibilidade, balanceamento de carga e escalabilidade

# ReplicaSet


Replica Set
Replicas: 2
10.244.2.2
:80
Web
Container 1
Pod 1
Status: Ready
10.244.2.3
:80
Web
Container 2
Status:
Crashed
Pod 2
10.244.2.4
:80
Web
Container 2
Pod 3
Status: Creating

# *ReplicaSet*: como saber quais *pods* monitorar?

vagrant@s2-master-1: ~

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      class: test
  template:
    metadata:
      labels:
        class: test
    spec:
      containers:
      - name: myapp
        image: nginx
~
~
~
~
~
~
                                                    17,1          All
```

*Labels* e seletores

# ReplicaSet

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: rs-nginx
  template:
    metadata:
      labels:
        app: rs-nginx
    spec:
      containers:
      - name: rs-nginx
        image: nginx:alpine
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-app
  template:
    metadata:
      labels:
        app: deploy-app
    spec:
      containers:
      - name: nginx
        image: nginx:alpine
```

# O que acontece ao remover *Pods* de um *ReplicaSet*?

```
> k get rs
NAME        DESIRED   CURRENT   READY   AGE
rs-nginx    3         3         3       9s
 ~/ads19
> k get pod
NAME              READY   STATUS    RESTARTS   AGE
rs-nginx-4ht64    1/1     Running   0          19s
rs-nginx-kqqrl    1/1     Running   0          19s
rs-nginx-w48c7    1/1     Running   0          19s
 ~/ads19
> k delete pod rs-nginx-4ht64
pod "rs-nginx-4ht64" deleted
 ~/ads19
> k get pod
NAME              READY   STATUS    RESTARTS   AGE
rs-nginx-9vc26    1/1     Running   0          4s
rs-nginx-kqqrl    1/1     Running   0          44s
rs-nginx-w48c7    1/1     Running   0          44s
 ~/ads19
```

# Escalando *ReplicaSets*

```
> k scale rs rs-nginx --replicas 5
replicaset.apps/rs-nginx scaled
 ~/ads19
> k get pod
NAME              READY   STATUS    RESTARTS   AGE
rs-nginx-9vc26    1/1     Running   0          89s
rs-nginx-crdk7    1/1     Running   0          3s
rs-nginx-kqqrl    1/1     Running   0          2m9s
rs-nginx-szrms    1/1     Running   0          3s
rs-nginx-w48c7    1/1     Running   0          2m9s
```

# Atualizando *ReplicaSets*

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: rs-nginx
  template:
    metadata:
      labels:
        app: rs-nginx
    spec:
      containers:
      - name: rs-nginx
        image: nginx:alpine
```

```
> k apply -f manifest/rs-app.yaml
replicaset.apps/rs-nginx created
> k get pods
NAME              READY   STATUS    RESTARTS   AGE
rs-nginx-9vjx7    1/1     Running   0          10s
rs-nginx-gmdb8    1/1     Running   0          10s
rs-nginx-mkcsr    1/1     Running   0          10s
> k describe rs rs-nginx
Name:         rs-nginx
Namespace:    dev
Pod Template:
  Labels:  app=rs-nginx
  Containers:
   rs-nginx:
    Image:        nginx:alpine
    Port:         <none>
```

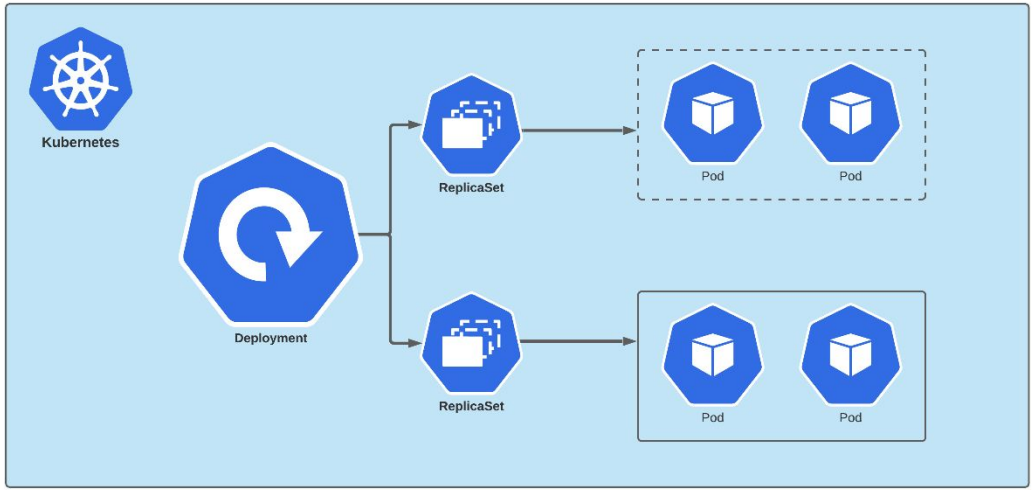
# Atualizando *ReplicaSets*

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: rs-nginx
  template:
    metadata:
      labels:
        app: rs-nginx
    spec:
      containers:
      - name: rs-nginx
        image: nginx
```

```
> k apply -f manifest/rs-app.yaml
replicaset.apps/rs-nginx configured
> k describe rs rs-nginx
Name:         rs-nginx
Namespace:    dev
Pod Template:
  Labels:  app=rs-nginx
  Containers:
   rs-nginx:
    Image:        nginx
    Port:         <none>
> k describe pod rs-nginx-9vjx7
Name:             rs-nginx-9vjx7
Namespace:        dev
Containers:
  rs-nginx:
    Container ID:   docker://a8c462081eb5cd0a3dece83
    Image:          nginx:alpine
```

# *Deployments*

Método **declarativo** para descrição de estados de *Pods* e *ReplicaSets*



# Casos de uso para *Deployments*

Provê um nível de abstração mais elevado que ReplicaSets, com mais funcionalidades

Permite gerenciar recursos de forma declarativa, com mais parâmetros editáveis

Permite a realização de *rollbacks*

Suporta diferentes estratégias de *rollout* de novas versões

# Casos de uso para *Deployments*

Permite a escalabilidade para aumentar capacidade de atendimento

Possibilita a pausa do *deployment* enquanto múltiplas modificações são feitas a *PodTemplateSpecs*

Possui indicadores de estado que facilitam decisões sobre *rollouts*

Remove *ReplicaSets* mais antigos automaticamente

# Criando *Deployments* via imperativa

```
> k create deployment deploy-nginx --image=nginx:alpine
deployment.apps/deploy-nginx created
 ~/ads19
> k get pod
NAME                          READY   STATUS    RESTARTS   AGE
deploy-nginx-86cdd9dc84-hf9vm   1/1     Running   0          10s
```

# Criando *Deployments* via declarativa (arquivos YAML)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: deploy-app
  template:
    metadata:
      labels:
        app: deploy-app
    spec:
      containers:
        - name: nginx
          image: nginx:alpine
```

# Criando *Deployments* via declarativa (arquivos YAML)

```
> k apply -f manifest/deploy-app.yaml
deployment.apps/deploy-app created
 ~/ads19                                                    kuberr
> k get pod
NAME                          READY   STATUS    RESTARTS   AGE
deploy-app-6485d658d7-h92lx   1/1     Running   0          10s
deploy-app-6485d658d7-j4lzt   1/1     Running   0          10s
 ~/ads19                                                    kuberr
> k get all
NAME                                  READY   STATUS    RESTARTS   AGE
pod/deploy-app-6485d658d7-h92lx       1/1     Running   0          24s
pod/deploy-app-6485d658d7-j4lzt       1/1     Running   0          24s

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/deploy-app    2/2     2            2           25s

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/deploy-app-6485d658d7    2         2         2       24s
```

# Atualizando *Deployments*

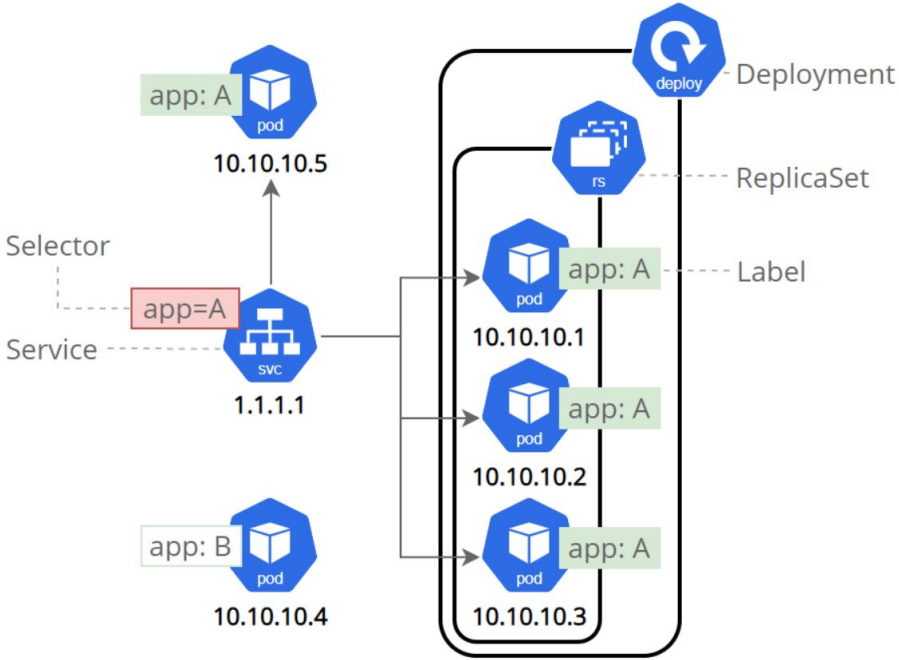
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: deploy-app
  template:
    metadata:
      labels:
        app: deploy-app
    spec:
      containers:
        - name: nginx
          image: nginx
```

```
> k apply -f manifest/deploy-app.yaml
deployment.apps/deploy-app configured
 ~/ads19
>
 ~/ads19
> k get pod
NAME                          READY   STATUS        RESTARTS   AGE
deploy-app-6485d658d7-h92lx   1/1     Terminating   0          4m56s
deploy-app-755f77d794-cvmdc   1/1     Running       0          12s
deploy-app-755f77d794-rftnq   1/1     Running       0          7s
```

```
> k describe pod deploy-app-755f77d794-cvmdc
Name:             deploy-app-755f77d794-cvmdc
Namespace:        dev
Containers:
  nginx:
    Container ID:   docker://97906a097344
    Image:          nginx
    Image ID:       docker-pullable://ngi
```

# *Services*

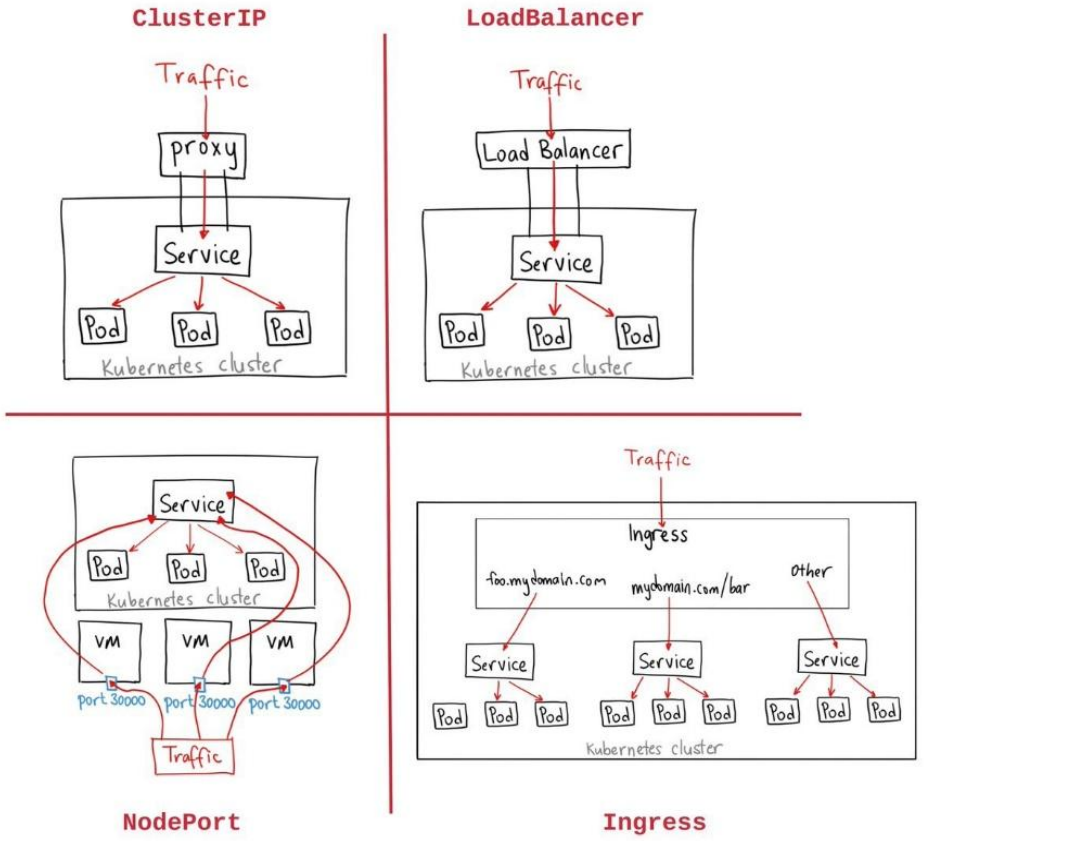
Maneira abstrata para expor uma aplicação executando em um conjunto de *Pods*



# Tipos de *services*

PADRÃO

- *ClusterIP*
- *NodePort*
- *LoadBalancer*
- *ExternalName*



# Criando *Services*

```
> k create service clusterip svc-app --tcp=8080:80 --dry-run=client -o yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: svc-app
  name: svc-app
spec:
  ports:
  - name: 8080-80
    port: 8080
    protocol: TCP
    targetPort: 80
  selector:
    app: svc-app
  type: ClusterIP
```

# Service

```
> k get pod
NAME                          READY   STATUS    RESTARTS   AGE
deploy-app-6485d658d7-f59ph   1/1     Running   0          2m51s
 ~/ads19                                                                  kuber
> k get svc
NAME       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
svc-app    ClusterIP   10.108.17.48   <none>        8080/TCP   2m16s
 ~/ads19                                                                  kuber
> k run curl --image=curlimages/curl -- sleep 360
pod/curl created
 ~/ads19                                                                  kuber
> k exec curl -- curl svc-app:8080
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
curl: (7) Failed to connect to svc-app port 8080 after 6 ms: Couldn't connect to server
command terminated with exit code 7
```

# Service

```
> k describe svc svc-app
Name:                     svc-app
Namespace:                dev
Labels:                   app=svc-app
Annotations:              <none>
Selector:                 app=svc-app
Type:                     ClusterIP
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.108.17.48
IPs:                      10.108.17.48
Port:                     8080-80  8080/TCP
TargetPort:               80/TCP
Endpoints:                <none>
Session Affinity:         None
Events:                   <none>
```

# Service

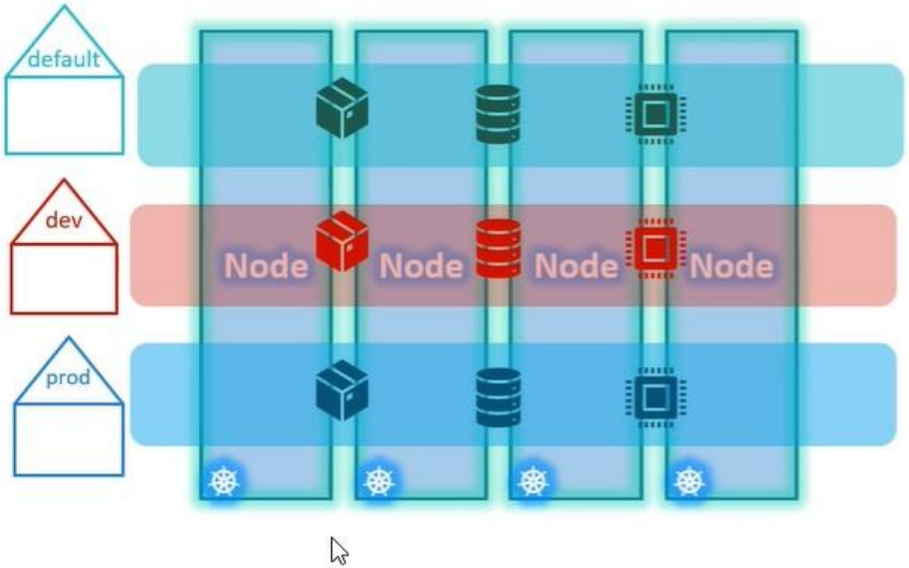
```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: svc-app
  name: svc-app
spec:
  ports:
  - name: 8080-80
    port: 8080
    protocol: TCP
    targetPort: 80
  selector:
    app: svc-app
  type: ClusterIP
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: deploy-app
  template:
    metadata:
      labels:
        app: deploy-app
    spec:
      containers:
        - name: nginx
          image: nginx:alpine
```

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: svc-app
  name: svc-app
spec:
  ports:
  - name: 8080-80
    port: 8080
    protocol: TCP
    targetPort: 80
  selector:
    app: deploy-app
  type: ClusterIP
```

# Namespace

Construto que permite a criação de *clusters* virtuais no mesmo *cluster* físico



# Quando usar *namespaces*?

*Namespaces* são ideais para ambientes complexos, com múltiplos usuários/times/projetos

Deve-se ter cuidado para não usar *namespaces* de forma a dificultar a gestão do ambiente

É possível segmentar recursos do *cluster* entre *namespaces*

Para diferenciar recursos levemente diferentes, utilize ***labels***

# Operando com *namespaces*

vagrant@s2-master-1: ~

```
root@s2-master-1:~# kubectl create namespace dev
namespace/dev created
root@s2-master-1:~#
root@s2-master-1:~#
root@s2-master-1:~# kubectl -n dev run webapp --image=nginx:alpine
pod/webapp created
root@s2-master-1:~#
root@s2-master-1:~#
root@s2-master-1:~# kubectl get pod
No resources found in default namespace.
root@s2-master-1:~#
root@s2-master-1:~#
root@s2-master-1:~# kubectl -n dev get pod
NAME     READY   STATUS    RESTARTS   AGE
webapp   1/1     Running   0          13s
root@s2-master-1:~#
```

# Alterando o *namespace* padrão

vagrant@s2-master-1: ~

```
root@s2-master-1:~# kubectl config set-context --current --namespace=dev
Context "kubernetes-admin@kubernetes" modified.
root@s2-master-1:~#
root@s2-master-1:~#
root@s2-master-1:~# kubectl get pod
NAME      READY   STATUS    RESTARTS   AGE
webapp    1/1     Running   0          4m
root@s2-master-1:~# 3~
```

# Alterando o *namespace* padrão

GitHub - ahmetb/kubectx: Faster way to switch be...

Faster way to switch between clusters and namespaces in kubectl - GitHub - ahmetb/kubectx: Faster way to switch

https://github.com/ahmetb/kubectx

ahmetb/**kubectx**

Faster way to switch between clusters and namespaces in kubectl

60 Contributors | 20 Used by | 4 Discussions | 16k Stars | 1k Forks

https://github.com/ahmetb/kubectx

# Interação entre *namespaces* e nomes DNS

```
> k create ns teste
namespace/teste created
 ~/ads19
> kubens teste
Context "kubernetes-admin@kubernetes" modified.
Active namespace is "teste".
 ~/ads19
> k get all
No resources found in teste namespace.
 ~/ads19
> k run curl --image=curlimages/curl -- sleep 360
pod/curl created
 ~/ads19
> k exec curl -- curl -s svc-app:8080
command terminated with exit code 6
 ~/ads19
> k exec curl -- ping -c 1 svc-app
ping: bad address 'svc-app'
command terminated with exit code 1
```

# Interação entre *namespaces* e nomes DNS

```
> k -n dev exec curl -- nslookup svc-app
Server:         10.96.0.10
Address:        10.96.0.10:53

** server can't find svc-app.cluster.local: NXDOMAIN

** server can't find svc-app.cluster.local: NXDOMAIN

Name:   svc-app.dev.svc.cluster.local
Address: 10.108.17.48

** server can't find svc-app.svc.cluster.local: NXDOMAIN


** server can't find svc-app.svc.cluster.local: NXDOMAIN
```

```
> k exec curl -- curl -s svc-app.dev:8080 | grep title
<title>Welcome to nginx!</title>
```

# Técnicas de manipulação de objetos

| Management technique | Operates on | Recommended environment | Supported writers | Learning curve |
|---|---|---|---|---|
| Imperative commands | Live objects | Development projects | 1+ | Lowest |
| Imperative object configuration | Individual files | Production projects | 1 | Moderate |
| Declarative object configuration | Directories of files | Production projects | 1+ | Highest |

https://kubernetes.io/docs/concepts/overview/working-with-objects/object-management/

# Exemplos

```
> k create deployment deploy-app --image=nginx:alpine
deployment.apps/deploy-app created
 ~/ads19                                                    kubernetes-admin@kubernetes/dev kube
> k get deployments.apps
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
deploy-app   1/1     1            1           16s
 ~/ads19                                                    kubernetes-admin@kubernetes/dev kube
> k delete deployments.apps deploy-app
deployment.apps "deploy-app" deleted
 ~/ads19                                                    kubernetes-admin@kubernetes/dev kube
> k create -f manifest/deploy-app.yaml
deployment.apps/deploy-app created
 ~/ads19
> vim manifest/deploy-app.yaml
 ~/ads19                                                    18s | kubernetes-admin@kubernetes/dev kube
> k create -f manifest/deploy-app.yaml
Error from server (AlreadyExists): error when creating "manifest/deploy-app.yaml": deployments.apps "deploy-app" already
 exists
```

# Exemplos

```
> k delete deployments.apps deploy-app
deployment.apps "deploy-app" deleted
 ~/ads19
> k apply -f manifest/deploy-app.yaml
deployment.apps/deploy-app created
 ~/ads19
> vim manifest/deploy-app.yaml
 ~/ads19
> k apply -f manifest/deploy-app.yaml
deployment.apps/deploy-app configured
```

# Exemplos de uso de comandos imperativos

```
# kubectl run squirtle --image=nginx:alpine
pod/squirtle created
```

```
# kubectl create ns fire
namespace/fire created
```

```
# kubectl -n fire create deploy charmander --image=fbscarel/myapp-redis --replicas=3
deployment.apps/charmander created
```

```
# kubectl create svc clusterip bulbasaur-svc --tcp=5432
service/bulbasaur-svc created
```

```
# kubectl -n fire create svc nodeport charmander --tcp=80 --node-port=31080
service/charmander created
```

```
# kubectl -n fire run db --image=redis:alpine --port=6379 --expose
service/db created
pod/db created
```

# Tarefa 2

As atividades práticas desta sessão podem ser obtidas em formato HTML via:

https://bit.ly/ads19-tarefas-s2

ESCOLA SUPERIOR DE REDES

# Arquitetura e conceitos do Kubernetes

RNP
ORGANIZAÇÃO SOCIAL DO MCTI
MINISTÉRIO DO TURISMO
MINISTÉRIO DA DEFESA
MINISTÉRIO DA SAÚDE
MINISTÉRIO DA EDUCAÇÃO
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES
PÁTRIA AMADA BRASIL
GOVERNO FEDERAL