# Arquitetura do Kubernetes

## Master

**API** – Responsável por receber e executar os novos comandos, manter a comunicação entre todos os componentes, gerenciar os recursos do cluster podendo através dela criar um pod, editar um replica set, ler os dados de um deployment, deletar um volume, tudo isso sempre através da API.

Quem expõem a API do Kubernetes é o **kube-apiserver**, ele é o front-end para a camada de gerenciamento do Kubernetes.

A forma de se comunicar com a API é através da ferramenta **kubectl**, é através dela que pode-se criar, ler, atualizar e remover recursos do cluster, isso de duas maneiras.

1 – declarativa (arquivos yaml ou json), 2 – imperativa (executar comandos).

**Controller Manager** – Responsável por manter e atualizar o estado desejado.

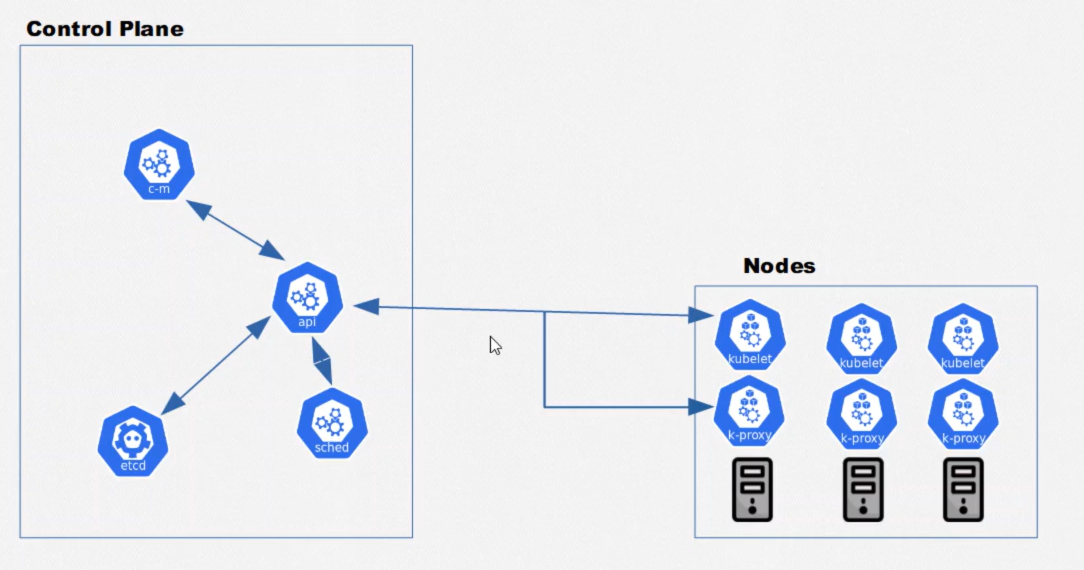
**kube-scheduler** – É um componente da camada de gerenciamento que observa os pods recém criados sem nenhum nó atribuído e seleciona o nó para executá-los, para isso diversos fatores são levados em consideração desde disponibilidade de hardware dos nós até políticas e restrições.

**ETCD** – Responsável por armazenar todos os dados do cluster K8S em um banco de dados chave: valor.

## Node

**Kubelet –** Responsável pela execução dos **PODs** dentro dos **NODEs**.

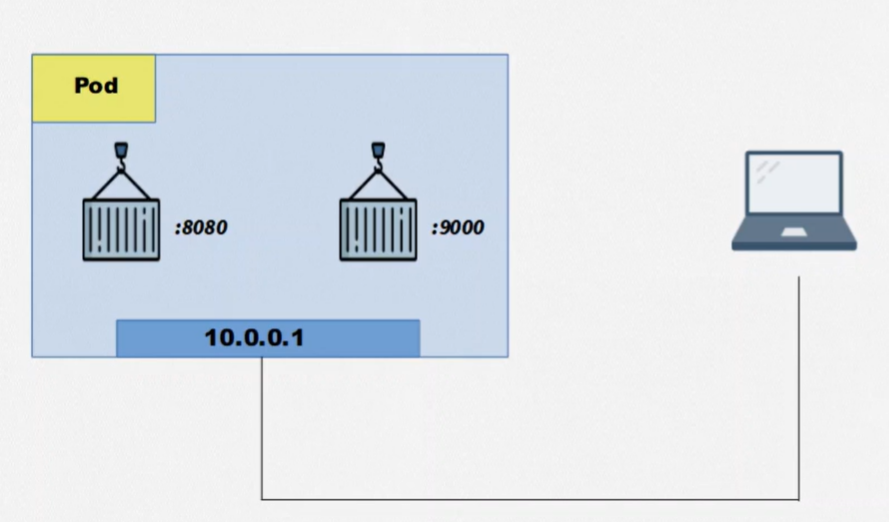
**Kube-proxy –** Responsável pela comunicação dos **NODEs** dentro do **Cluster**.



# PODs

Um POD pode ter um ou mais containers e quando um POD é inicializado ele recebe um IP.

Cada container dentro do POD pode iniciar a aplicação em uma porta desde que **sejam portas diferentes**.



PODs são efêmeros, eles foram feitos para morrer. No exemplo acima onde existem dois containers dentro do mesmo POD, enquanto pelo menos um container estiver saudável o POD se manterá em pé, quando os dois containers estiverem com problema / falhas o POD será morto e outro será inicializado (com um IP diferente).

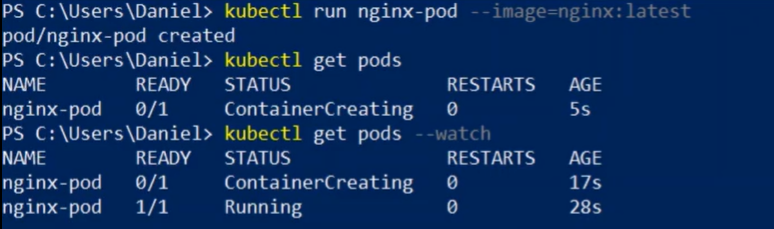
**Qual a vantagem de dois containers estarem no mesmo POD?**

Uma das vantagens é que eles compartilham o mesmo IP (10.0.0.1), sendo assim a comunicação entre eles é via localhost.

## Criando um POD

**kubectl run nginx-pod --image=nginx:latest**

kubectl run **nome-pod** --image=**imagem:versao**



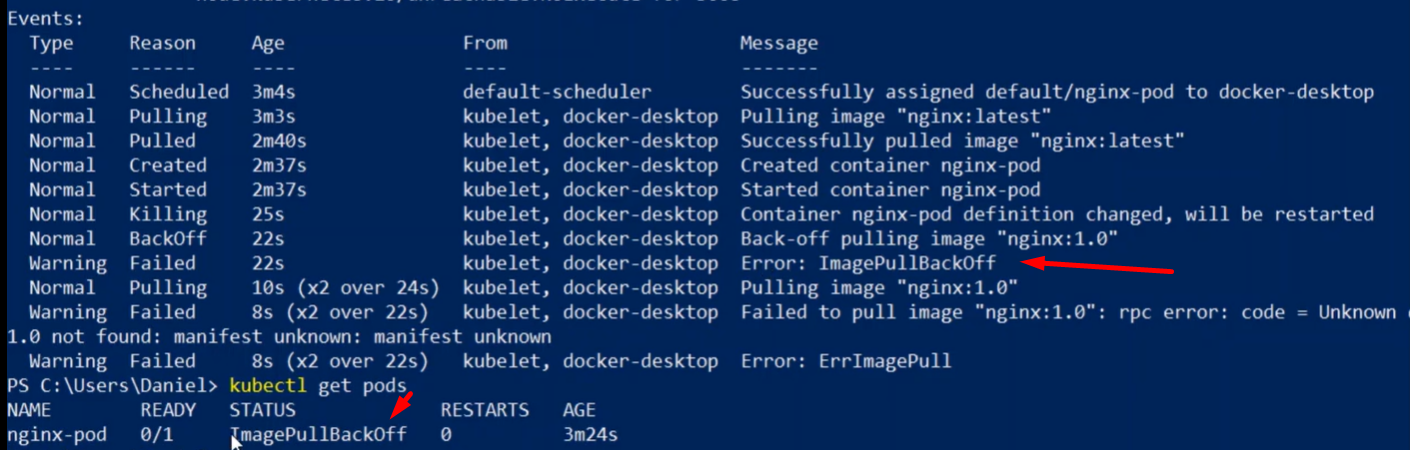
Obs: Quando um POD é iniciado o **scheduler** definirá em qual Node o POD será iniciado e as imagens que estarão dentro do Node são baixadas localmente e não para todos os Nodes do cluster.

Após a criação do container, quando o STATUS ainda estava ContainerCreating pode-se executar o comando **kubectl get pods --watch** isso irá mostrar em tela cada alteração de status até o container ser criado.

Caso queira editar algo nas configurações do POD existe uma maneira mais trivial que é,

**kubectl edit pod nginx-pod**

### Erro ImagePullBackOff



Esse erro ImagePullBackOff geralmente ocorre quando não é possível baixar a imagem que o POD irá utilizar. Repare na mensagem de erro: “Failed to pull imagem nginx:1.0” – “Falha para baixar a imagem nginx:1.0”

## Criando POD de maneira declarativa.

apiVersion: v1

kind: Pod

metadata:

name: primeiro-pod-declarativo

spec:

containers:

- name: nginx-container

image: nginx-latest  
 ports:  
 - containerPort: 80

Criar PODs (ou recursos) de maneira declarativa significa especificar o que será criado em um arquivo que pode ser yaml ou json para que assim o kubectl consiga enviar o comando e a API interpretar.   
**kubectl apply -f ./primeiro-pod-declarativo.yaml**

## Excluindo PODs

Para excluir um POD é possível executar de algumas maneiras.

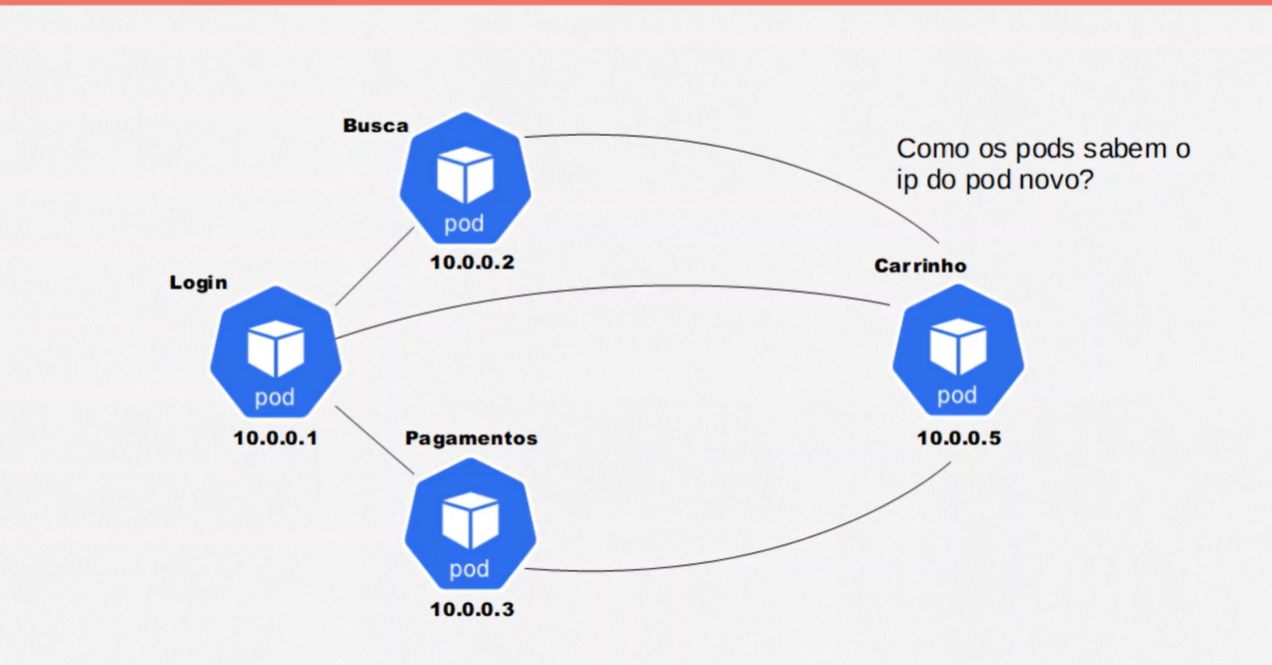
1 - kubectl delete pod **nome\_do\_pod**

2 – Da mesma maneira que pode-se criar PODs de maneira declarativa, também é possível remover.

kubectl **delete** -f ./primeiro-pod-declarativo.yaml, dessa o comando irá verificar o nome do POD definido no arquivo primeiro-pod-declarativo.yaml e deletar.

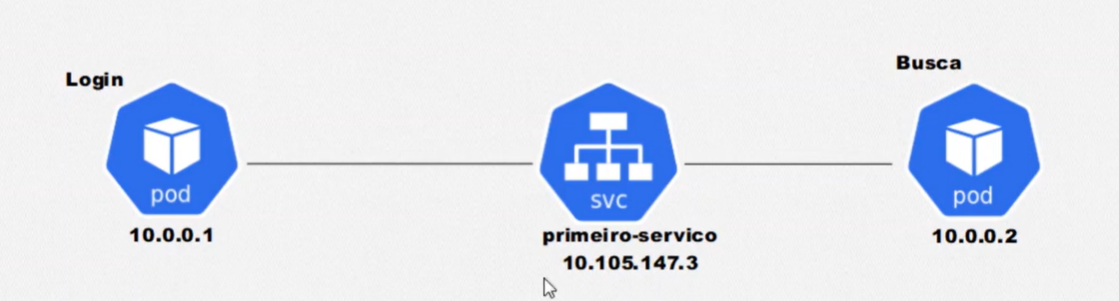
# SERVICES

Como os PODs sabem o IP dos outros PODs?



Já que os PODs são efêmeros e sempre são recriados os IPs sempre mudam, então a solução para não sofrer com a mudança de IPs dos PODs é a criação de um service, o IP do service é único e ele estará “a frente” do POD, é o service que será chamado e não o POD diretamente.

**CHAMADA -> SERVICE (SVC)-> POD**



Vantagens dos services;

Fazem o balanceamento de carga.

Proveem IP's fixos para comunicação.

## TIPOS de Serviços

* **ClusterIP** – comunicação de diferentes PODs dentro de um cluster. Não é uma comunicação externa ao cluster.

Essa comunicação é feita em uma via de mão única, a comunicação é somente para um POD não quer dizer que aquele POD poderá se comunicar com os outros.

* NodePort
* LoadBalance

# COMANDOS

kubectl run nginx-pod --image=nginx:latest

kubectl get pods –watch

kubectl edit pod nginx-pod

kubectl describe pod nginx-pod

kubectl run, kubectl describe, kubectl edit (criam, descrevem e editam pods, respectivamente).

kubectl apply -f ./primeiro-pod.yaml # criar um pod de maneira declarativa.

kubectl delete -f ./primeiro-pod.yaml # deletar um pod de maneira declarativa.

kubectl delete pod nome\_do\_pod # deletar um pod através do nome

kubectl exec -it porta-noticias -- bash #kubectl exec -it **nome\_do\_pod** -- **comando**