Introdução ao Kubernetes

Pedro Horchulhack

Agenda

1. Arquitetura geral
2. Atividade

Arquitetura geral

Arquitetura geral

- Entendemos que aplicações contêinerizadas são “empacotadas” para serem executadas de forma virtualizada
  - Distribuição fácil
  - Implantação fácil

- A gestão de contêineres é algo complicado principalmente quando a quantidade aumenta
  - O que fazer nesse cenário?

Arquitetura geral

- A Amazon Web Services (AWS) mudou o que entendemos como computação em nuvem
  - Google foi uma das empresas que tentou acompanhar o ritmo

- O Google já utilizava contêineres para provisionar suas soluções
  - Motor de busca
  - Gmail

- Deram início ao Kubernetes (K8S) e tinham dois objetivos
  - Abstrair a infraestrutura subjacente da AWS
  - Facilitar o processo de tirar/colocar aplicações na nuvem

Arquitetura geral

- De onde vem o nome Kubernetes?
  - Palavra grega κυβερνήτης (kyvernítis)
  - Timoneiro, piloto

Fonte: https://kubernetes.io

Arquitetura geral

- Podemos entender o Kubernetes como um orquestrador, um gestor de contêineres
  - Implanta sua aplicação
  - Escala para mais ou para menos dinamicamente, baseado na demanda
  - Se autocura quando ocorre algum erro (caso configurado corretamente)
  - Realiza lançamentos contínuos com *zero-downtime*
  - …

- E o que esse orquestrador tem a ver com Docker?
  - São tecnologias complementares

Arquitetura geral

– Podemos entender sua diferença dessa maneira:

Docker	Kubernetes
Construir aplicações para contêineres	Implantar contêineres
Executa contêineres	Gerencia contêineres
Operações de baixo nível	Operações de alto nível

Arquitetura geral

- O Kubernetes suporta outros motores de contêineres além do Docker
  - Containerd
  - Kata
  - Rkt
  - Runc
  - LXC

- Também existem outros orquestradores de contêineres:
  - Borg
  - OpenShift
  - Mesos
  - AWS ECS
  - …

Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

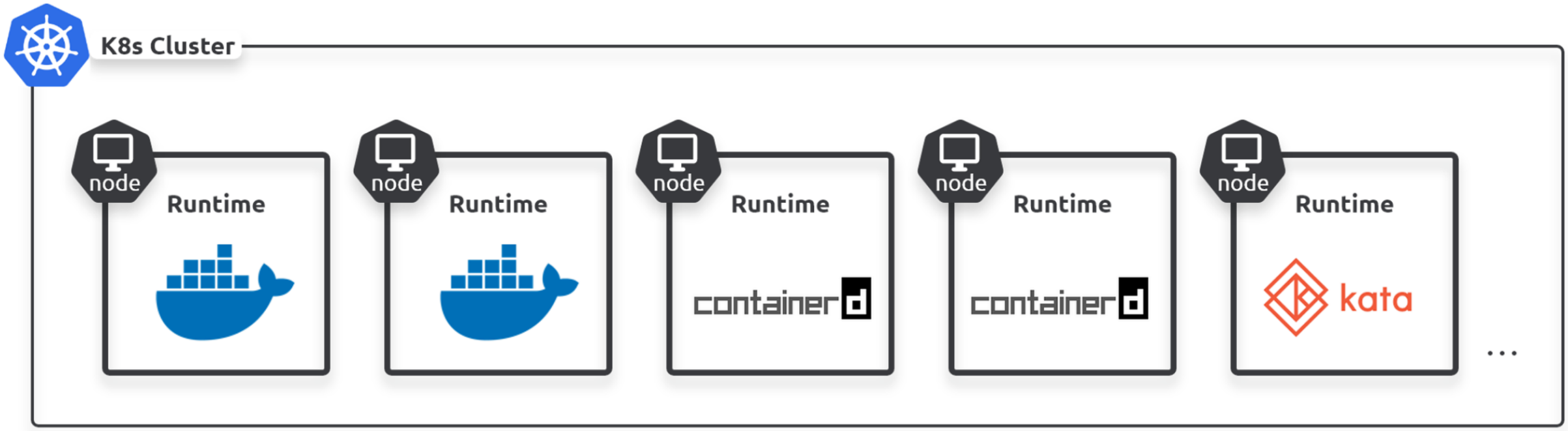
Arquitetura geral

- O orquestrador surgiu como uma plataforma para implantação de aplicações nativas para nuvem
  - Atua como um SO para a nuvem

- Ele irá **abstrair** os recursos do servidor e **escalonar** processos de aplicação

Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Arquitetura geral



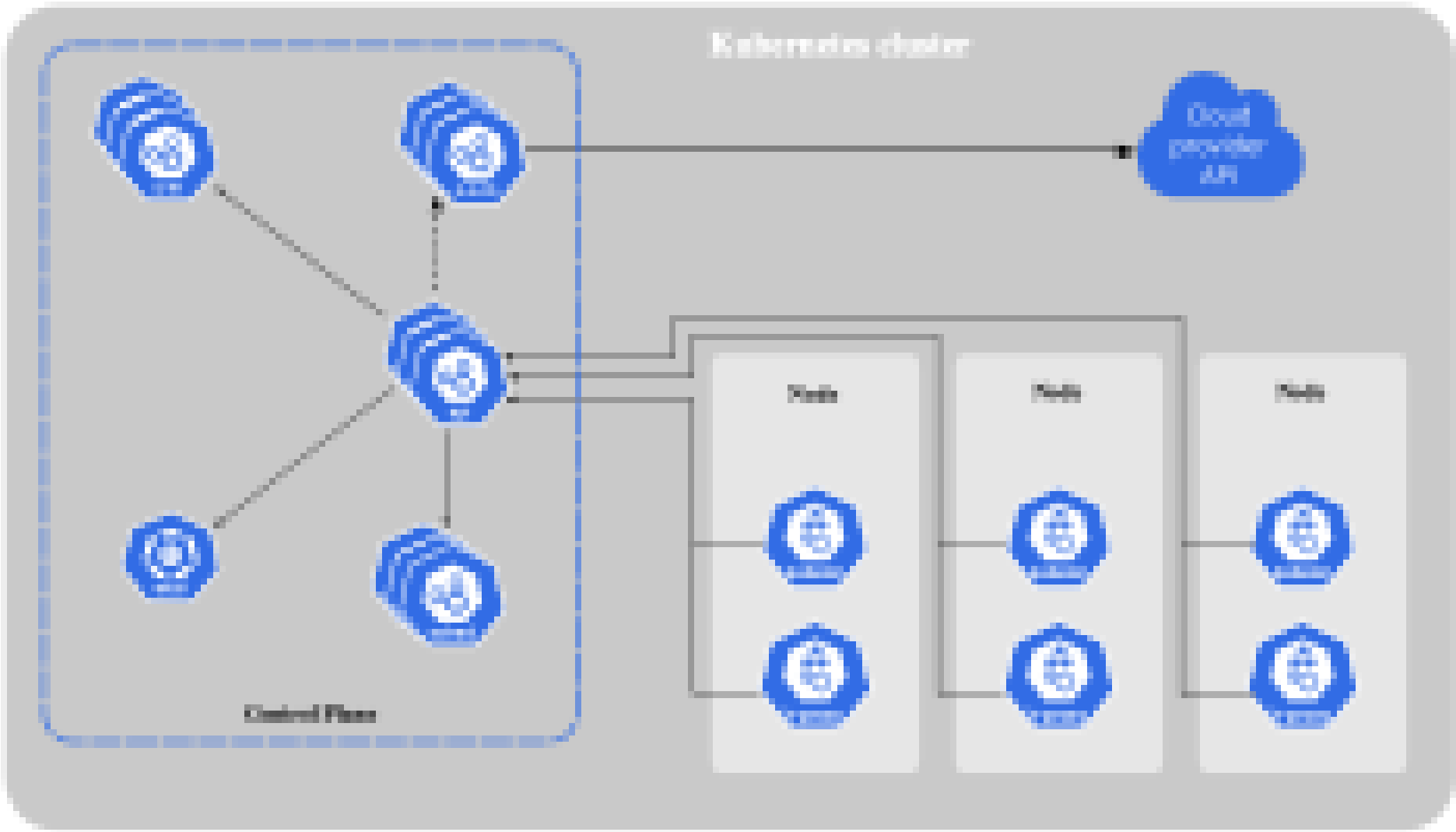
Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Arquitetura geral

- Um cluster Kubernetes corresponde a um conjunto de nós (nodes), controlados pelo **control plane**
  - Um node corresponde a uma máquina física será quem instanciará os contêineres
  - O control plane será responsável pela gestão dos recursos dos nodes e alocação de novas aplicações

- Na prática ocorre a sequência:
  - Projeta e escreve um microsserviço
  - Empacota cada microsserviço em uma imagem
  - Empacota cada contêiner da imagem em um Pod
  - Implanta em um cluster através de um objeto de controle de alto nível (Deployment, StatefulSet, DaemonSet, CronJob etc.)

Arquitetura geral



Fonte: https://kubernetes.io/pt-br/docs/concepts/overview/components/

Arquitetura geral – Control Plane

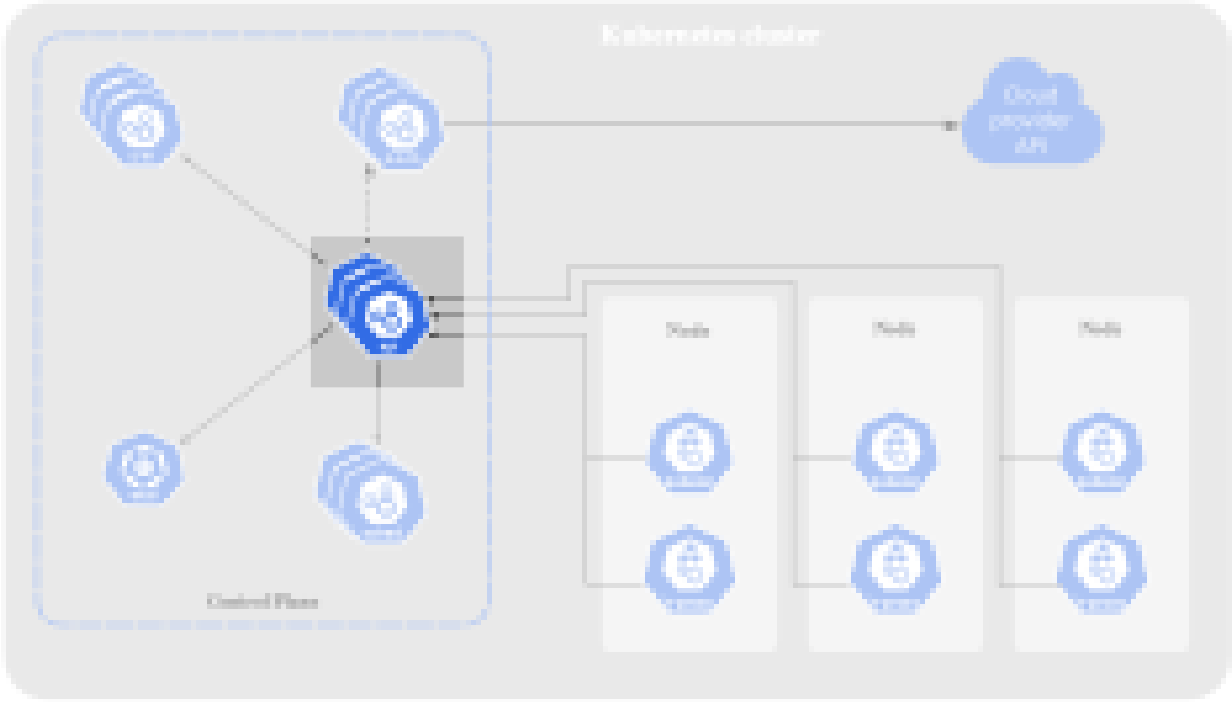
- Como já vimos, é o “cérebro” do cluster

- Comumente temos somente **um** control plane no cluster
  - Podemos configurar mais de um para garantir alta disponibilidade (geralmente entre 3 e 5)

- É uma boa prática não executar aplicações de usuários no control-plane
  - Deixa recursos disponíveis para o gestor trabalhar livremente

Arquitetura geral – API Server

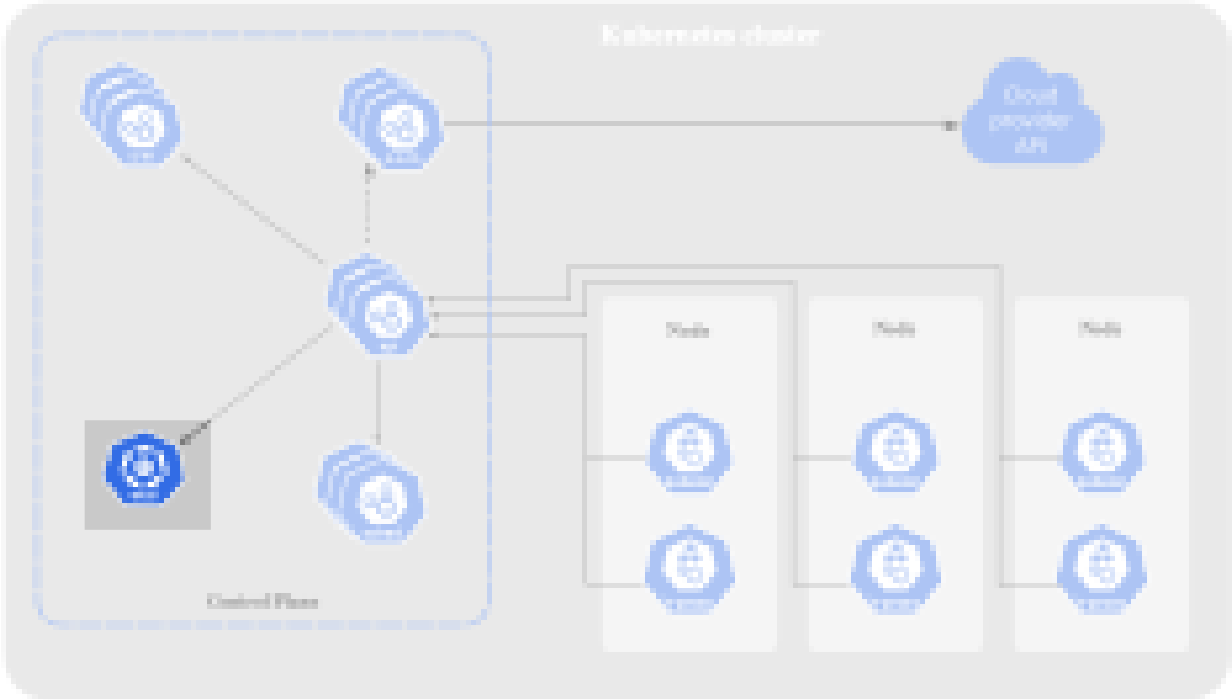
- É a central de comunicação do Kubernetes
  - Responsável por toda a comunicação entre todos os componentes, inclusive com o administrador do cluster
- É simplesmente uma API RESTful que envia uma requisição POST sobre HTTPS
- Faz verificações de autenticação e autorização
  - Depois de válidos, procede a validação do arquivo YAML enviado;
  - Agenda e escalona os recursos descritos no YAML aos nós



Fonte: https://kubernetes.io/pt-br/docs/concepts/overview/components/

Arquitetura geral – Armazenamento do cluster

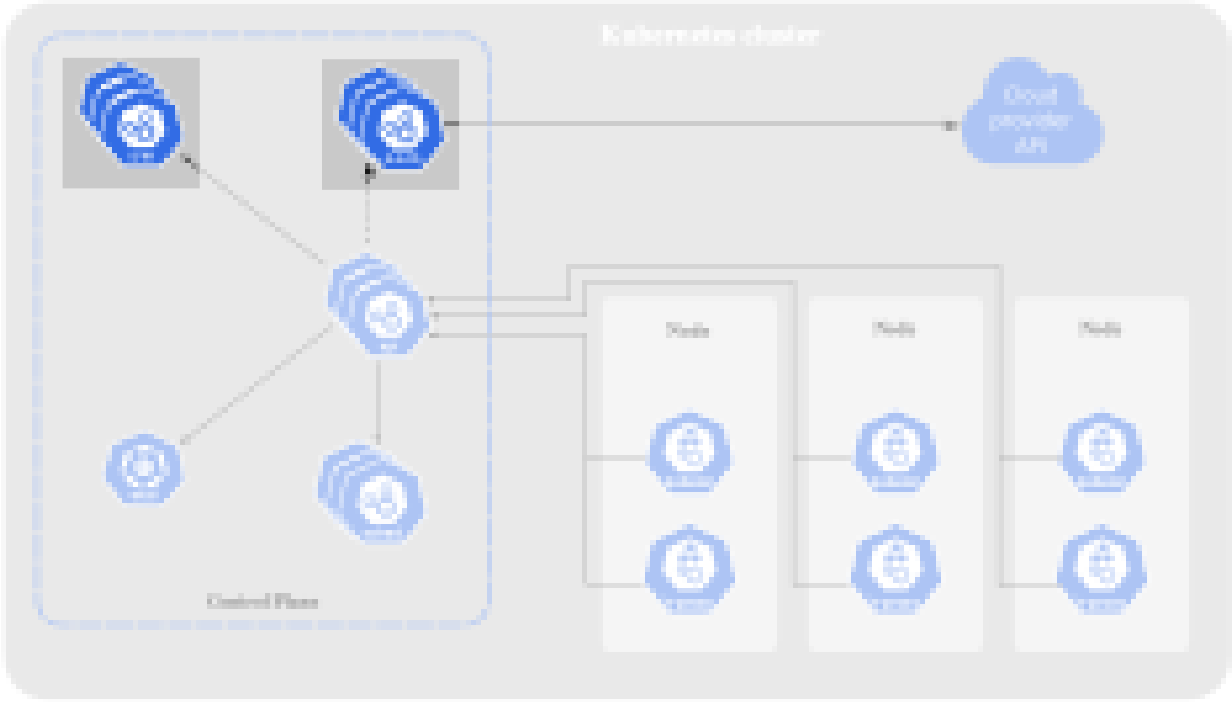
- É a única parte *stateful* e armazena toda a configuração do cluster
  - Logs
  - Aplicações rodando
  - Aplicações em espera
  - Configurações de rede
  - Configurações de nós
  - …
- Também é recomendado mais de um componente de armazenamento por cluster
- Utiliza o banco de dados distribuido `etcd`



Fonte: https://kubernetes.io/pt-br/docs/concepts/overview/components/

Arquitetura geral – Gerenciador de controladores

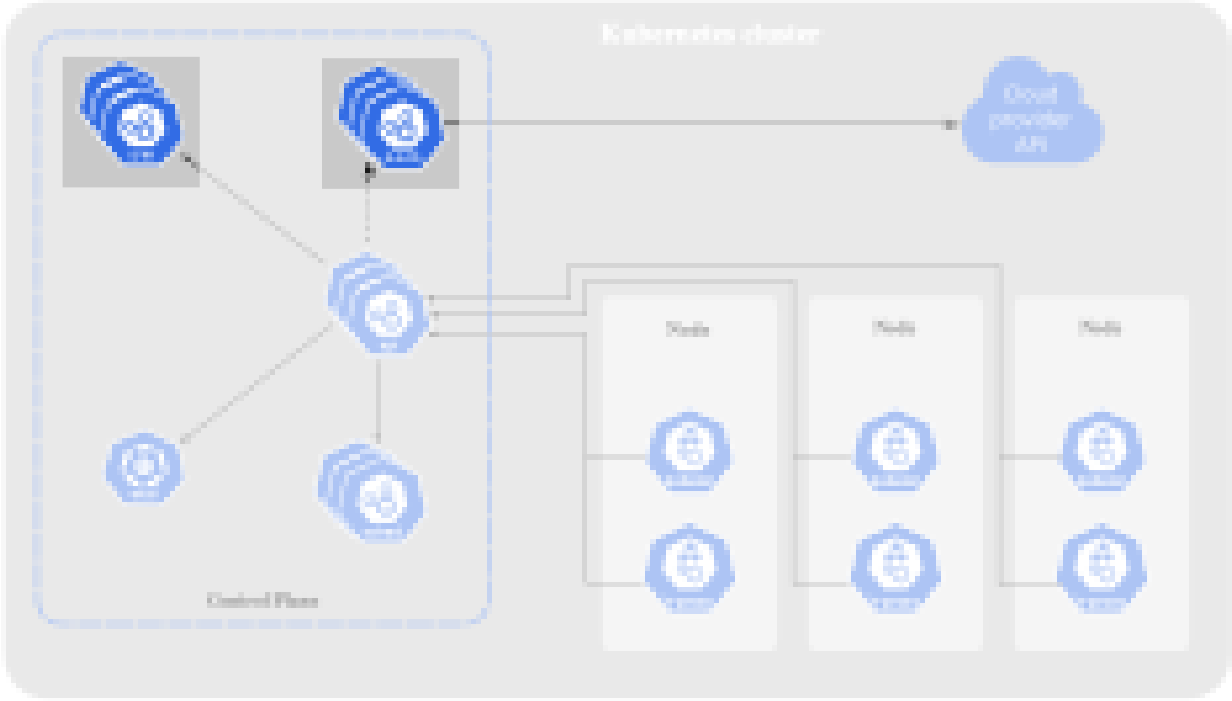
- Implementa todos os controladores de segundo plano que monitora os componentes do cluster e responde a eventos
  - É criado junto com a criação do cluster
- O que são controladores?
  - Deployment
  - StatefulSet
  - ReplicaSet
- Fornecem um comportamento específico para o *control plane* e *gerenciador de controladores* trabalharem



Fonte: https://kubernetes.io/pt-br/docs/concepts/overview/components/

Arquitetura geral – Gerenciador de controladores

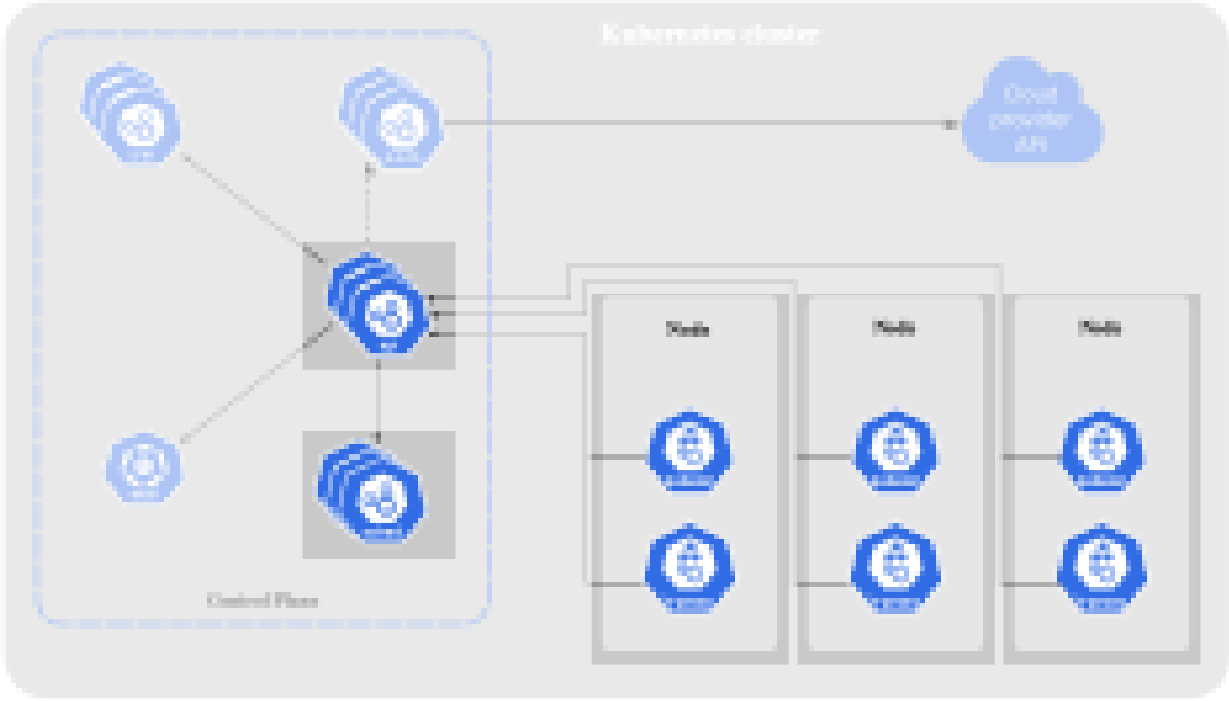
- Ele implementa um *watch-loop* que fica verificando e tentando resolver quatro estados:
  1. Obter estado desejado
  2. Observar o estado atual
  3. Analisar diferenças
  4. Reconciliar diferenças



Fonte: https://kubernetes.io/pt-br/docs/concepts/overview/components/

Arquitetura geral – Escalonador

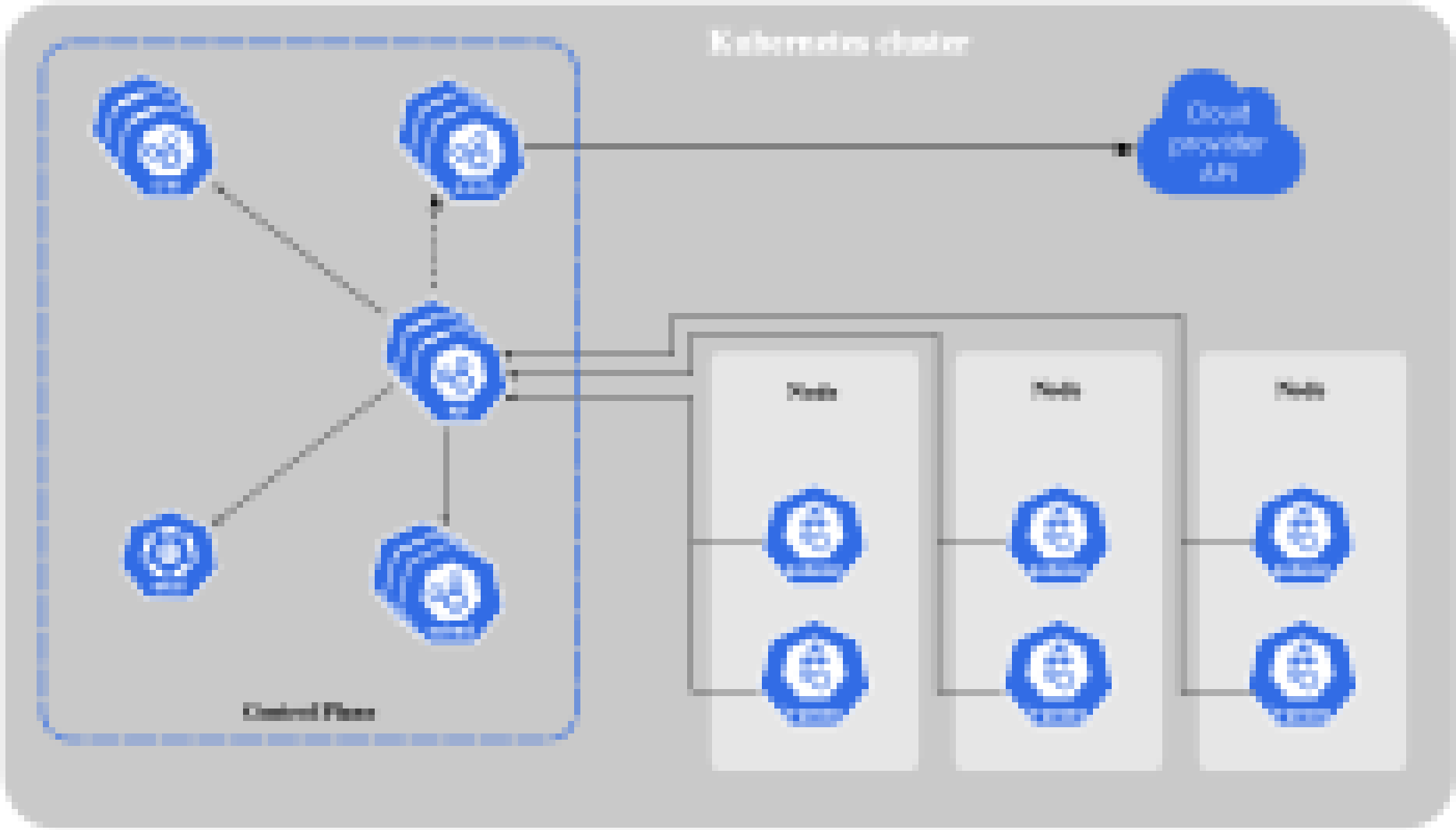
- Espera por novas tarefas e atribui elas ao melhor nó “saudável” com base em critérios internos
  - Realiza um filtro para os nós incapazes de executar uma nova tarefa naquele momento
  - Ranqueia os melhores nós para indicar o mais adequado para determinada tarefa
- Faz uma verificação de uma série de atributos:
  - Afinidade e anti-afinidade
  - Nós *tainted*
  - Especificações de rede (porta ou endereço disponível)
  - Recursos disponíveis
  - …



Caso o escalonador não encontre um nó, a tarefa fica em estado *pendente*!

Fonte: https://kubernetes.io/pt-br/docs/concepts/overview/components/

Arquitetura geral

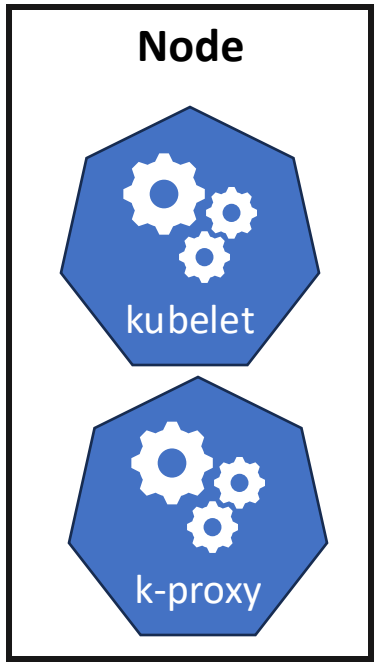


Fonte: https://kubernetes.io/pt-br/docs/concepts/overview/components/

Arquitetura geral

- Cada nó possui três componentes fundamentais para o Kubernetes executar corretamente
  - Kubelet
  - Runtime de contêineres (Docker, por exemplo)
  - Proxy de rede (kube-proxy)

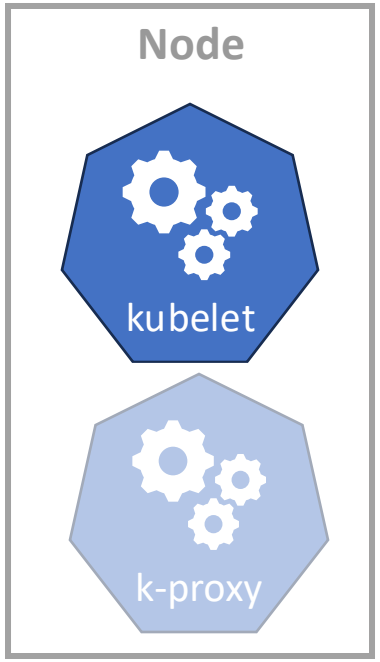
- Em alto nível, fazem:
  - Observação de novas tarefas através da API
  - Executa atribuições de tarefas
  - Reporta de volta a API



Arquitetura geral – Kubelet

- É o principal agente do Kubernetes e executa em todo nó do cluster
  - Aqui kubelet e nó (node) correspondem a mesma coisa, porém o kubelet é o componente que conversa diretamente com a API

- Além disso, ele monitora recursos com CPU, memória e armazenamento

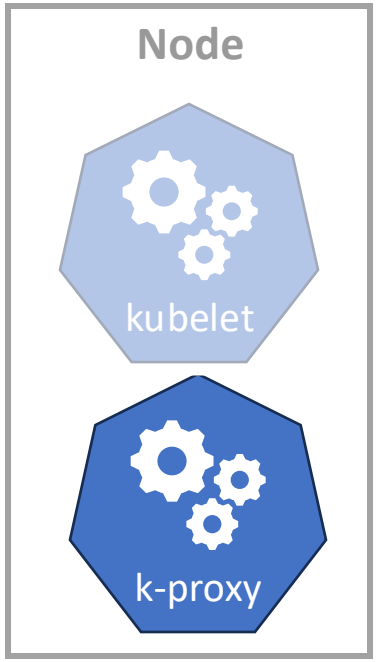


Arquitetura geral – Kube-proxy

- Como o objetivo do Kubernetes é a abstração de recursos, todos os recursos são enxergados como uma coisa só

- Toda comunicação entre API e nós é realizada através desse kube-proxy

- É responsável pela gestão da rede
  - IP (nós)
  - IPTables local
  - Roteamento
  - Load-balancing



Arquitetura geral – Kubernetes DNS

- O Kubernetes também possui um DNS para publicar novas aplicações
  - Cada nó possui um endereço IP *hard-coded*

- O objetivo desse componente é garantir que cada contêiner e Pod possam localizar e utilizar os IPs

- O registro de serviços é automático, ou seja, não precisamos registrar manualmente nossa aplicação ou configurá-la para descobrir/ser descoberta na rede do Kubernetes

Arquitetura geral – Empacotando para o K8S

- De maneira semelhante a criação de imagens Docker declarativas, no Kubernetes é **tudo declarativo**
  - Utilizamos arquivos YAML

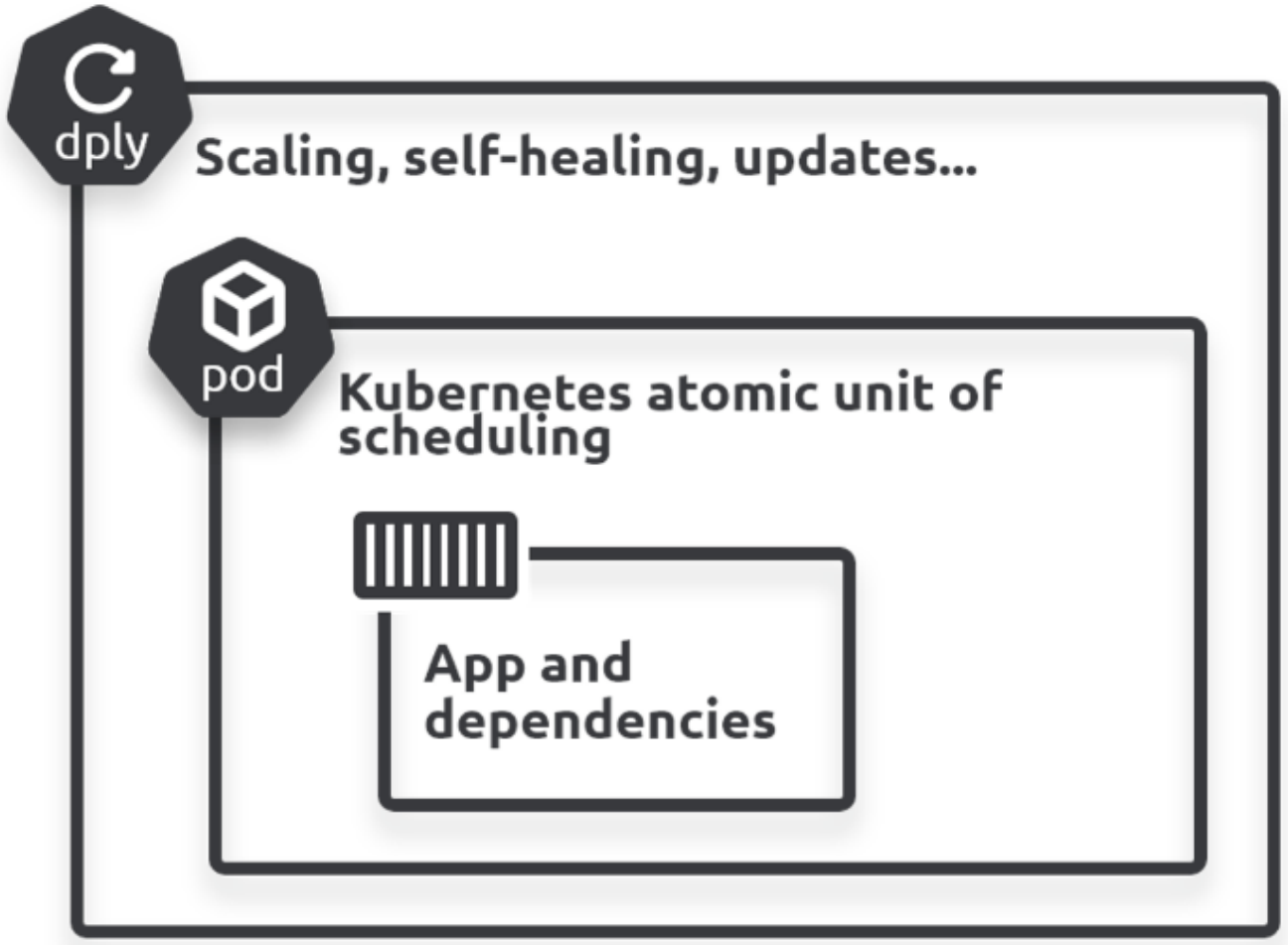
- Pod é a menor unidade de controle dentro do Kubernetes (Pod com a primeira letra maiúscula)
  - Atua como um envelopador de um contêiner para deixá-lo apto a executar no K8S
  - É a menor unidade escalonável no sistema

- O que os Pods tem a ver com contêineres?

Arquitetura geral – Empacotando para o K8S

- Envelopa contêineres para serem executados no Kubernetes

- O modelo mais simples é um contêiner por Pod
  - Podemos instanciar mais de um contêiner em um Pod
  - Contêineres (2 ou mais) no mesmo Pod executam no mesmo nó

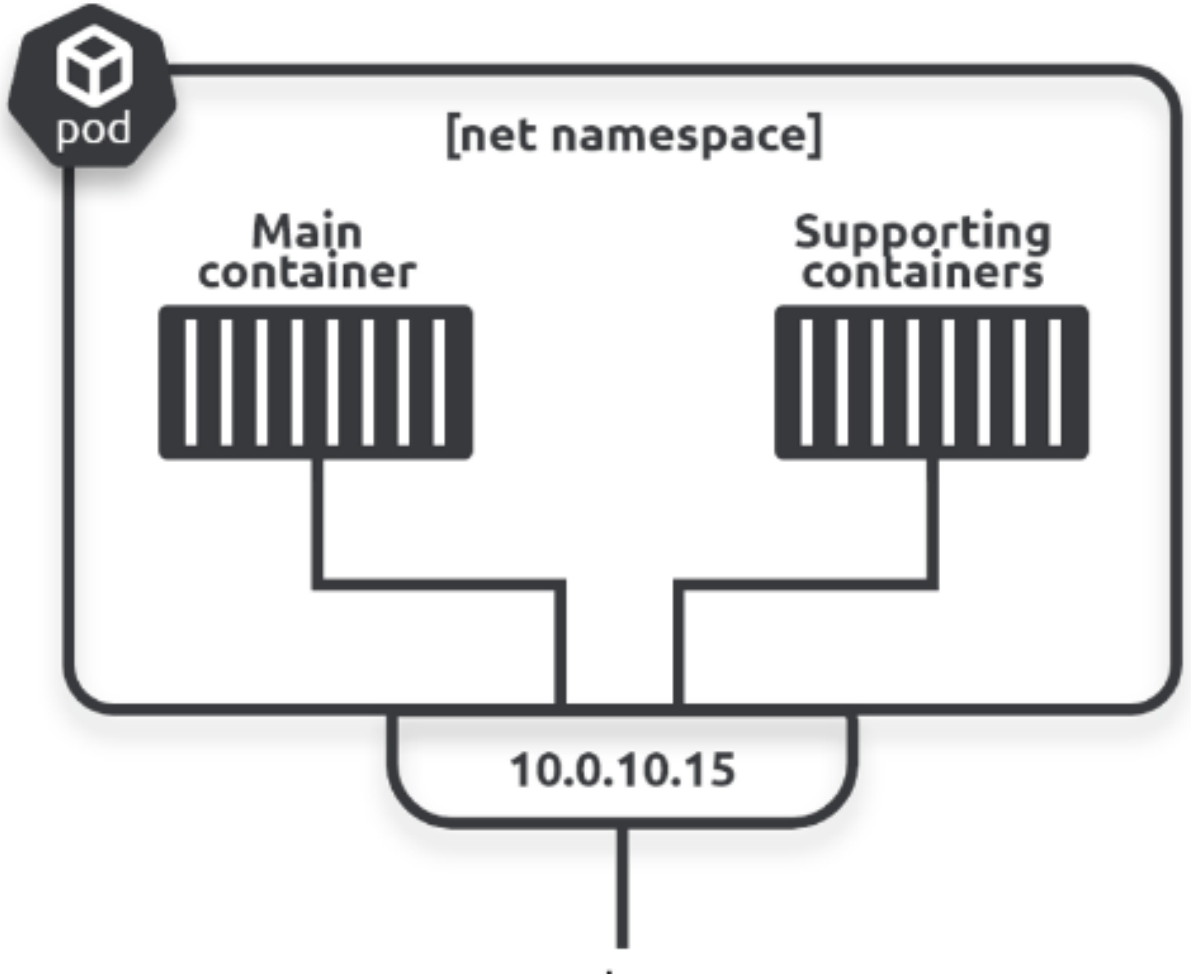


Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Arquitetura geral – Empacotando para o K8S

- Lembrando que Pods não executam aplicações, mas os contêineres dele

- Se tivermos mais de um contêiner executando no mesmo Pod eles compartilharão
  - Pilha de rede
  - Volumes
  - Namespaces
  - Memória compartilhada



Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Arquitetura geral – Empacotando para o K8S

- Pods possuem um ciclo de vida simples:
  - São criados (created/waiting)
  - Vivem (running)
  - Morrem (terminated)

- Se eles morrerem “do nada”, o Kubernetes irá criar um Pod novo, com um novo ID e IP
  - Isso implica no desenvolvimento das aplicações
  - O objetivo agora é desenvolver aplicações fracamente acopladas internamente

- Também são imutáveis. Não podemos alterá-lo uma vez executando.
  - Caso queiramos atualizá-lo, precisamos removê-lo e criá-lo novamente

Arquitetura geral – Comandos básicos

- kubectl é a ferramenta que nos comunicaremos com o Kubernetes

- Alguns comandos:

```
kubectl get nodes
kubectl get pods
kubectl describe pod <nome do pod>
kubectl apply -f <manifesto.[yaml | yml]>
kubectl delete -f <manifesto.[yaml | yml]>
kubectl version
kubectl logs -c <container> <pod>
kubectl top [pod | node]
```

Atividade

Atividade

- Vamos criar nosso cluster Kubernetes usando o minikube

```
sudo snap install microk8s --classic

sudo usermod -aG microk8s $USER
sudo mkdir -p ~/.kube
sudo chmod 0700 ~/.kube
su - $USER

microk8s start
microk8s status --wait-ready
microk8s enable dashboard

echo "alias kubectl='microk8s kubectl'" >> ~/.bashrc
```

Atividade

- Vamos criar nosso cluster Kubernetes usando o minikube

```
microk8s kubectl describe secret -n kube-system microk8s-dashboard-token

microk8s kubectl port-forward --address 0.0.0.0 -n kube-system service/kubernetes-dashboard 10443:443
```

Atividade

- Vamos criar um Pod simples com uma imagem Ubuntu

Referências

Referências

1. https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands
2. https://microk8s.io/docs/getting-started
3. https://microk8s.io/docs/addon-dashboard
4. https://www.aquasec.com/cloud-native-academy/container-platforms/container-engines/
5. https://gcore.com/blog/top-10-container-orchestration-tools/
6. https://kubernetes.io/pt-br/docs/concepts/overview/components/
7. https://kubernetes.io/pt-br/docs/concepts/overview/working-with-objects/
8. https://kubernetes.io/pt-br/docs/concepts/overview/

Referências

9. POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.
10. RICE, Liz. **Container security: fundamental technology concepts that protect containerized applications**. First edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2020.

Contato: p.horchulhack@pucpr.br