Habilitando Aplicações Nativas de Nuvem

Introdução a Contêineres e Kubernetes

3. Kubernetes

Sergio Rio
www.sergiorio.tech

Refrescando nossa memória

- Na aula passada estudamos as definições de escopo do produto “Catálogo Telefônico” e as atividades requeridas para preparação e construção dos aplicativo utilizando contêineres.

Programa: Introdução a Contêineres e Kubernetes

1. Conceitos Básicos

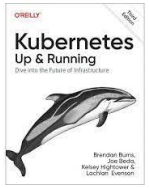
- ✓ Abstrações em Ciência da Computação
- ✓ Virtualização de Computadores
- ✓ MicroVMs e Unikernels

2. Contêineres

- Origem
- Fundamentos
- Criação e execução
- Registro e reuso
- Infraestrutura como Código
- Aplicativos Modernos
- Projeto

3. Kubernetes

- ➢ Origem
- ➢ Arquitetura
- ➢ Pods
- Abstrações de Recursos
- Descoberta de Serviços
- Serviços de Rede
- Instalação e administração básica
- Implantação de um caso de uso (exemplo)
- Projeto

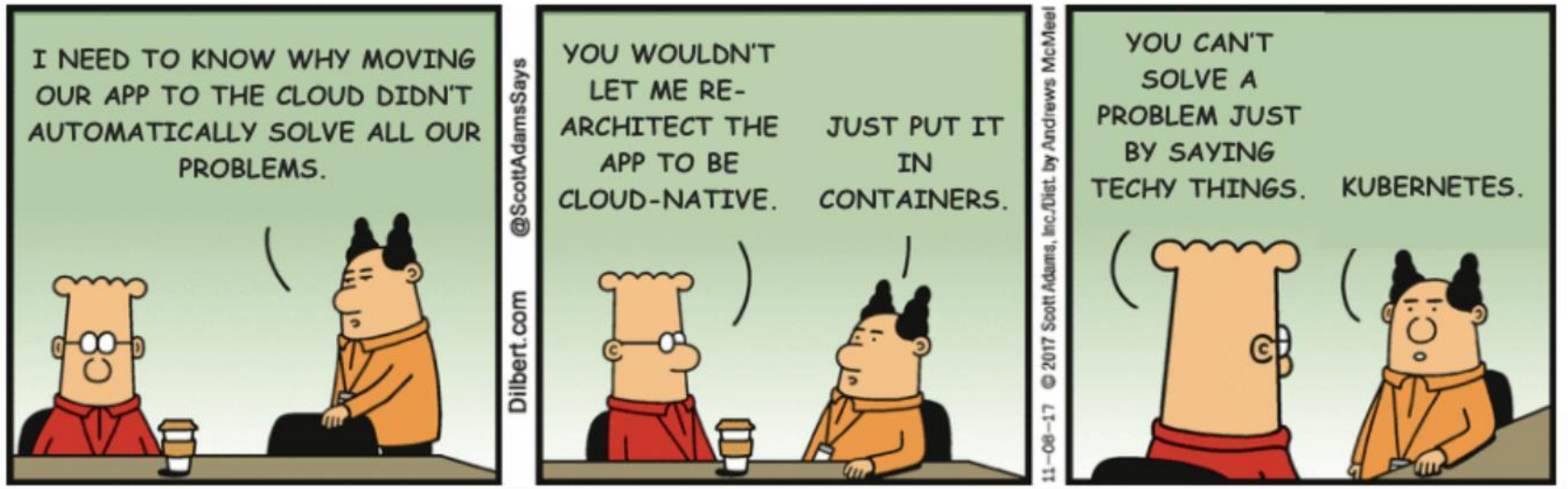


Kubernetes Up & Running
Brendan Burns, Joe Beda, Kelsey Hightower, and Lachlan Evenson
Cortesia da VMware Inc.

Kubernetes

Origem, Arquitetura e Pods

É sempre uma questão de princípios...

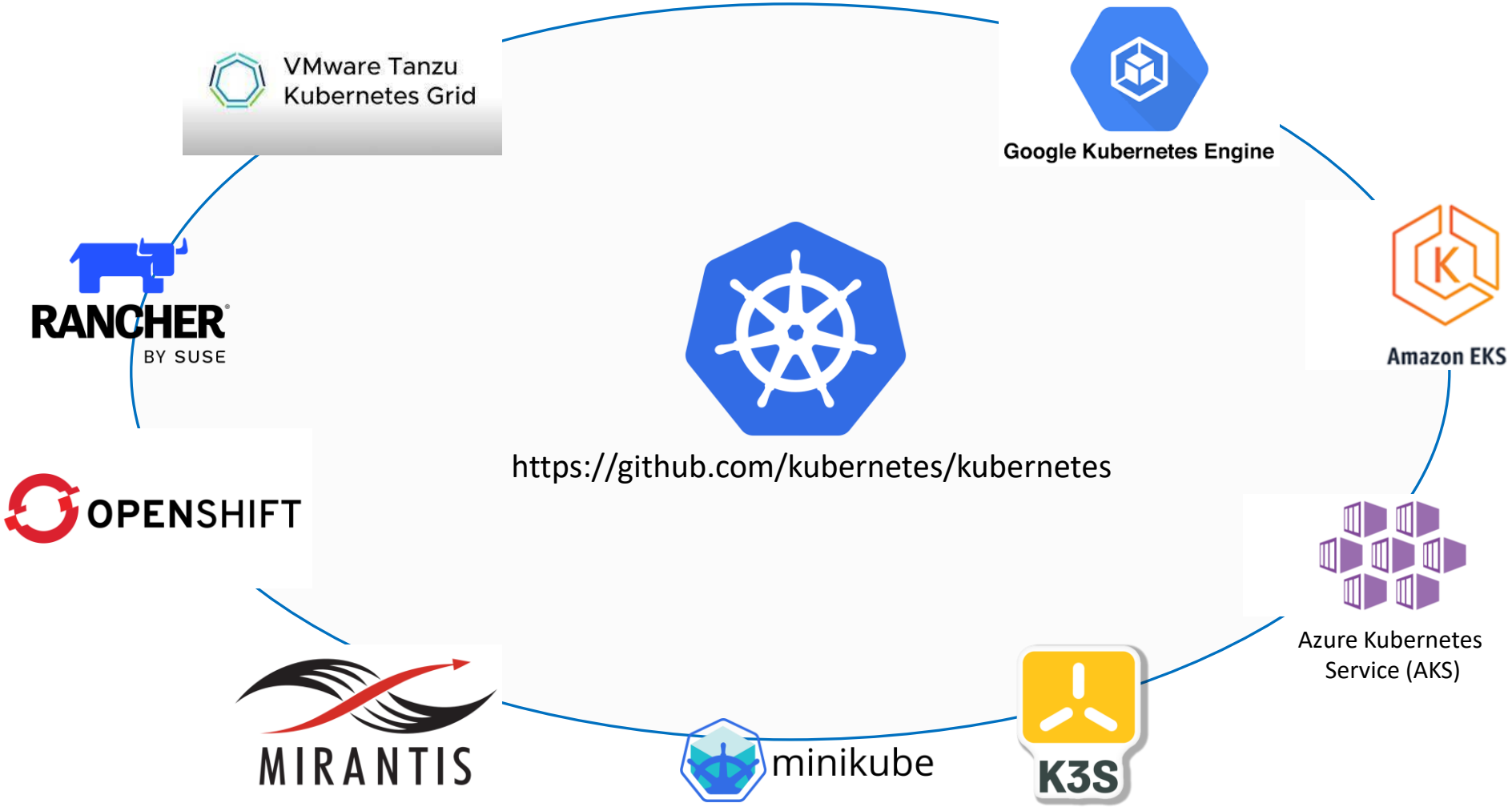


Kubernetes, “O Timoneiro”

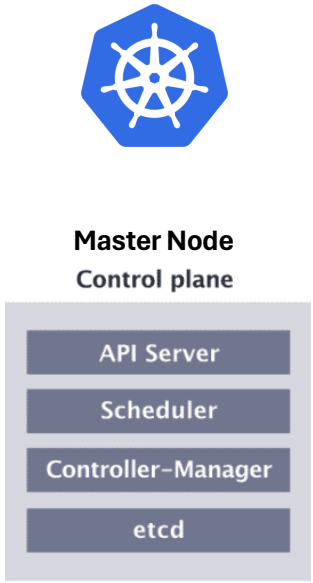
- O nome Kubernetes tem origem no Grego, significando timoneiro ou piloto. K8s é a abreviação derivada pela troca das oito letras "ubernete" por "8", se tornado K"8"s.
- Inicialmente um projeto interno da Google (“Borg”), posteriormente lançado como projeto de código aberto em 2014 e administrado desde 2018 pela CNCF, Cloud Native Computing Foundation.
- É uma plataforma de código aberto que automatiza a implantação, escalonamento e administração de aplicativos em contêineres.
- “É uma plataforma para construir plataformas”, Kelsey Hightower.

Source: https://kubernetes.io/

Ofertas de Kubernetes



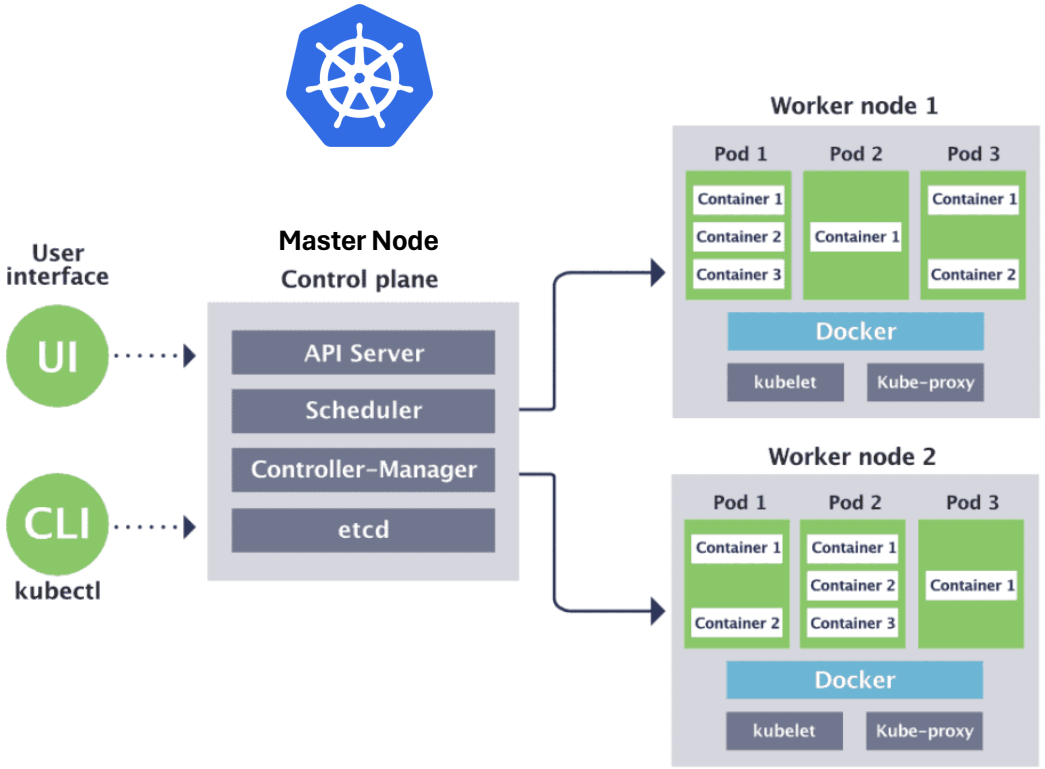
O “Timoneiro” em Ação! 1/2



- Agrupa contêineres que compõem um aplicativo em unidades logicas (Pods) para facilitar sua administração e descoberta.
- Administra o ciclo de vida dos contêineres incluindo “rollbacks” e pausas de implantação.
- Escalonamento automático (horizontal e vertical) conforme uso e demanda
- Modelo Declarativo
  - Desenvolvedores e administradores descrevem o estado requisitado.
  - K8s se encarrega de implantar o estado descrito de uma maneira dinâmica e de recuperá-lo em caso de falhas.
- Amplamente portável e extensível

Source: “How Kubernetes works” - https://www.cncf.io/blog/2019/08/19/how-kubernetes-works/

O “Timoneiro” em Ação! 2/2



- $ kubectl version
- $ kubectl get componentstatuses
- $ kubectl get nodes

Source: “How Kubernetes works” - https://www.cncf.io/blog/2019/08/19/how-kubernetes-works/

Pods

- *Pod* significa na Língua Inglesa coletivo de baleias. Termo usado pelos criadores do K8s para seguir a linha temática marítima do *Dockers* (“estivador” na Língua Portuguesa).
- *Pod* é uma abstração especial do K8s que agrupa um ou mais contêineres que compartilham um ou mais *namespaces*, como por exemplo uma rede. Nesse caso, os contêineres de um mesmo *pod* se comunicam através de *localhost (*mesma rede local*).*
- É a unidade mínima(atômica) de implementação de aplicativos em K8s. Todos os contêineres de um *pod* são executados em um mesmo nó do cluster.
- Aplicativos rodando em *pods* diferentes estão isolados: endereços IP, *hostnames*, etc. são diferentes.
- Os *pods* são desenhados para serem entidades relativamente efêmeras e descartáveis. Quando criados manualmente pelo *controller*, o novo *pod* será executado até sua conclusão ou seja deletado, ou *despejado* por falta de recursos naquele nó do cluster.
- Os nomes dos *pods* devem ser subdomínios DNS validos (ver RFC 1123).

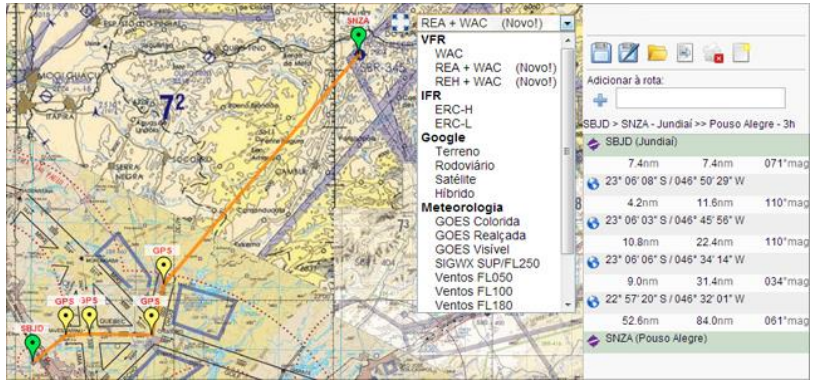
Kubernetes

Princípios

Principio K8s #1: Kube API declarativa ao invés de imperativa

Imperativo

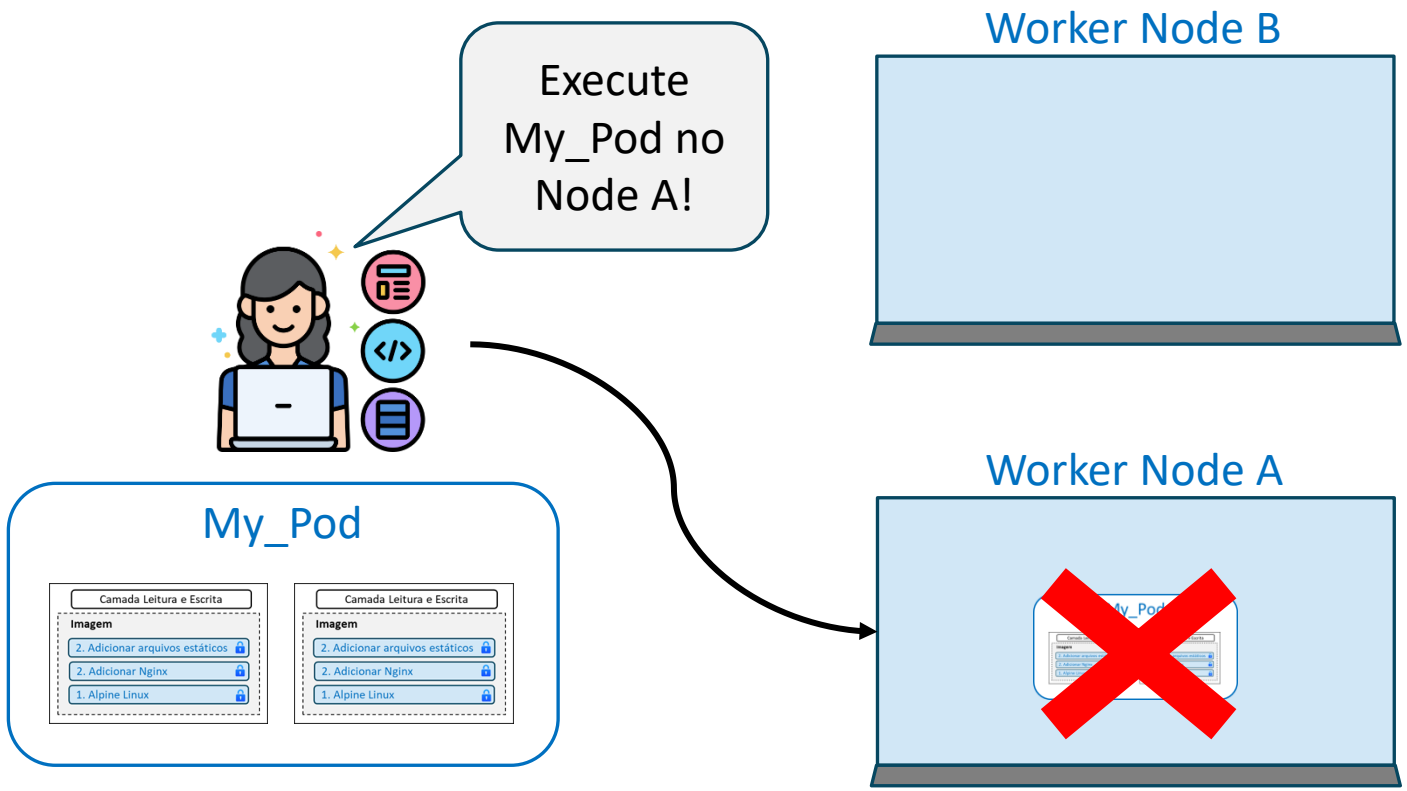
Pilotagem manual, todas as decisões e ações são tomadas pelos pilotos.



Declarativo

Pilotagem autônoma com base em especificações/declarações do estado desejado.

Principio K8s #1: Kube API declarativa ao invés de imperativa



E se:

- O *pod* ou contêineres crash!?
- O Node A crash!?
- E se o Node sofre um problema sério ainda que temporário?

Principio K8s #1: Kube API declarativa ao invés de imperativa

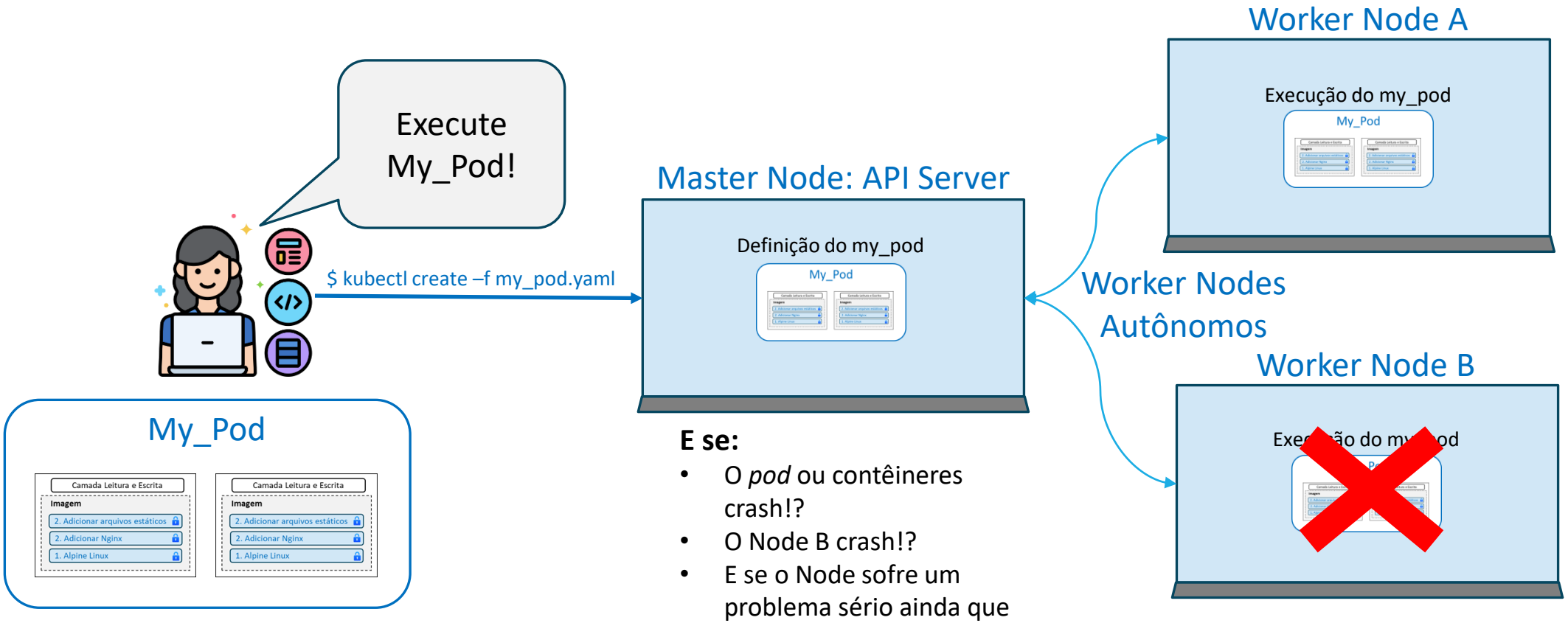
Antes

- **Você**: define a sequência exata de instruções para chegar ao estado desejado.
- **Sistema**: executa a sequência de instruções.
- **Você**: monitora o sistema e providencia novas instruções em caso de desvio.

Depois

- **Você**: define o estado desejado.
- **Sistema**: trabalha para atingir e manter o estado desejado.

Como implementar um workload?



E se:

- O *pod* ou contêineres crash!?
- O Node B crash!?
- E se o Node sofre um problema sério ainda que temporário?

Principio K8s #2: Não existem APIs internas e ocultas

O Control Plane do K8s é transparente!

Antes

- **Master**: define a sequência exata de instruções para o *Worker Node* chegar ao estado desejado.
- **Worker Node**: executa a sequência de instruções.
- **Master**: monitora o sistema e providencia novas instruções em caso de desvio no estado.

Depois

- **Master**: define o estado desejado do *Worker Node*.
- **Worker Node**: trabalha de maneira independente para atingir e manter o estado desejado.

E por que não existem APIs internas ocultas?

1. APIs declarativas proporcionam os mesmo benefícios aos *pods* ou aplicativos.
   - ✓Evita problemas com “eventos perdidos”.

2. Resulta em um sistema mais simples e robusto que pode facilmente recuperar-se de falhas:
   - ✓Nenhum ponto de falhas.
   - ✓Componentes mais simples no Master Node.

3. Permite ao K8s ser modular (composable) e expansível (extensible):
   - ✓Um componente default não atende as suas necessidades? Desabilite-o e o substitua com o seu próprio! Você pode desenvolver um novo *Scheduler* se quiser!
   - ✓Funcionalidade adicional ainda não disponível? Crie você mesmo e a adicione.

Kube API Data

- *K8s API* prove vários dados relevantes e interessantes para os *workloads* ou aplicativos:
  - *Secrets* – informações sensíveis (como senhas, certificados, etc.).
  - *ConfigMap* – configurações iniciais (como parâmetros de execução de aplicações ou scripts, etc.)
  - *DownwardAPI* – informações e especificações dos *pods* (como nomes, *namespaces, uids* do *pod* corrente)

Principio K8s #3: Encontrar os usuários onde eles estiverem

Antes

- Os aplicativos tem que ser adaptados, modificados ou até mesmo reconstruídos para serem compatíveis com o ambiente.

Depois

- Se um aplicativo consegue ler configurações de inicialização e dados sensíveis de um arquivo ou variáveis de ambiente ele não precisa ser modificado!

Principio K8s #4: Portabilidade de *workloads*

- Desacopla o desenvolvimento de um aplicativo de sistema distribuído da implementação do cluster.
- Faz do K8s uma verdadeira camada de abstração, como um *sistema operacional*.
- Como medir portabilidade?
  - PTO (*Portability Time Objective*) é o tempo máximo aceitável para mover um aplicativo e seus dados.

Atividades para a próxima aula

- Ler capítulos 3, 4, 5 e 6 do livro “Kubernetes Up and Running”
  - Deploying a Kubernetes Cluster
  - Common *kubectl* Commands
  - Pods
  - *Labels and Annotations*
- Laboratório tutorial Kubernetes *minikube:*
  1. Instalar e configurar *minikube* em instancia EC2 da *AWS:*
     - Setup Minikube on AWS EC2 Ubuntu
  2. Implementar um aplicativo na instancia *minikube*:
     - Hello Minikube