kubernetes

Sumário

O que é o Kubernetes?

O K8s é um projeto de código aberto que tem como objetivo orquestrar containers e automatizar a implantação de aplicações.

Atualmente mantido pela **Cloud Native Computing Foundation**, o Kubernetes gerencia os clusters que contêm os hosts que executam as aplicações Linux.

Esses clusters podem incluir hosts em nuvem, por isso, o Kubernetes é a plataforma ideal para hospedar aplicações *cloud-native* que exigem escalabilidade rápida, como a transmissão de dados em tempo real por meio do Apache Kafka.

Fazer o *deploy* de uma nova versão de uma aplicação é sempre um processo arriscado. Existem uma série de passos manuais ou semi-automatizados e, caso algo dê errado, fazer o *rollback* para a versão anterior é supercomplicado.

Agora imagine isso com uma aplicação composta por dezenas de microsserviços, cada um com um ciclo de vida diferente, datas de *release* diferentes, tecnologias diferentes.

Esse seria o pesadelo de qualquer time de desenvolvimento.

Por isso, ele elimina muitos processos manuais que uma aplicação em containers exige, facilitando e dando agilidade a projetos de microsserviços, por exemplo.

Como funciona o Kubernetes: Containers

Não tem muito mistério! Containers seguem basicamente a mesma lógica da sua contraparte literal.

Da mesma forma que agrupamos objetos que precisam ser enviados de um local para o outro em containers, também agrupamos nossos códigos em um container, que pode ser executado em diversos locais.

Dessa forma, podemos trabalhar com componentes menores, utilizando a arquitetura de microsserviços que, assim como o **Kubernetes**, está em alta.

Utilizar *microservices* e *containers* simplifica a vida do programador, pois dissipa codificações enormes em outras menores, impedindo que seu código vire um monstro.

Como funciona o Kubernetes: Aplicações cloud-native

C*loud-native* é o termo utilizado para classificar aplicações projetadas para tirar máximo proveito de ambientes em nuvem, seja ela privada ou pública.

Essas aplicações se baseiam na **arquitetura de microsserviços** e incorporam práticas que possibilitam a automação de todo o ciclo de vida da aplicação.

Existem aplicações conhecidas como **monolíticas**, onde todas as partes da aplicação vivem juntas (causando forte acoplamento).

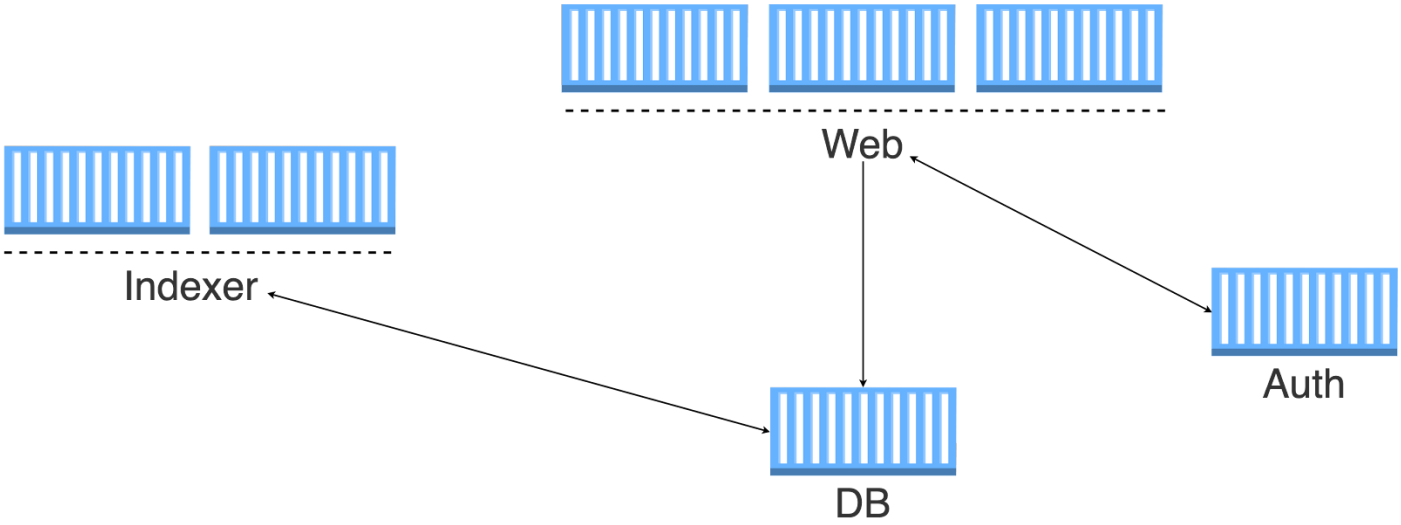
A arquitetura de microsserviços sugere a ideia de que aplicações devem ser compostas por partes menores e independentes chamadas de serviços (resultando em fraco acoplamento).

A ideia é que cada serviço seja especializado e ofereça uma API para se comunicar com outros serviços.

Essa característica possibilita, por exemplo, que diferentes times assumam diferentes partes de uma mesma aplicação.

Outra vantagem é que um mesmo serviço pode ser utilizado por múltiplas aplicações sem nenhum esforço extra.

Características das aplicações cloud-native



No caso de aplicações *cloud-native*, uma das principais características é que elas utilizam contêineres para encapsular cada microsserviço.

Como esses contêineres possuem o serviço e todas as suas dependências, eles se tornam independentes da infraestrutura, podendo ser facilmente migrados de uma *cloud* para outra, por exemplo.

Outro ponto importante é que a **utilização de contêineres** facilita muito questões como escalabilidade e *deploy* de novas versões.

No exemplo da imagem acima, se 3 contêineres da interface web não forem suficientes, basta iniciar mais um.

Saiu uma nova versão? Basta substituir os contêineres pela nova versão. A nova versão tem um bug crítico? Só substituir de volta pelo contêiner da versão anterior.

Esse tipo de flexibilidade traz diversas vantagens, mas também cria novos desafios. Quando uma aplicação é composta por diversas partes pequenas, gerenciar tudo isso de forma manual pode se tornar bem complexo.

E é para ajudar nessa parte que existem orquestradores como o Kubernetes.
Microservices e o perigo das “modinhas”

Funcionalidades do Kubernetes

Para ajudar a resolver o problema citado no começo do texto, o **Kubernetes** oferece uma série de funcionalidades. No entanto, antes de entrarmos em detalhes, é importante entendermos um conceito central do K8s: **o estado da aplicação**.

A ideia por trás desse conceito é que existem dois tipos de estado de uma aplicação: **o atual e o desejado.**

O estado atual da aplicação descreve a realidade. Por exemplo, quantas réplicas de um determinado serviço estão em execução, qual a versão em produção de cada serviço e por aí vai.

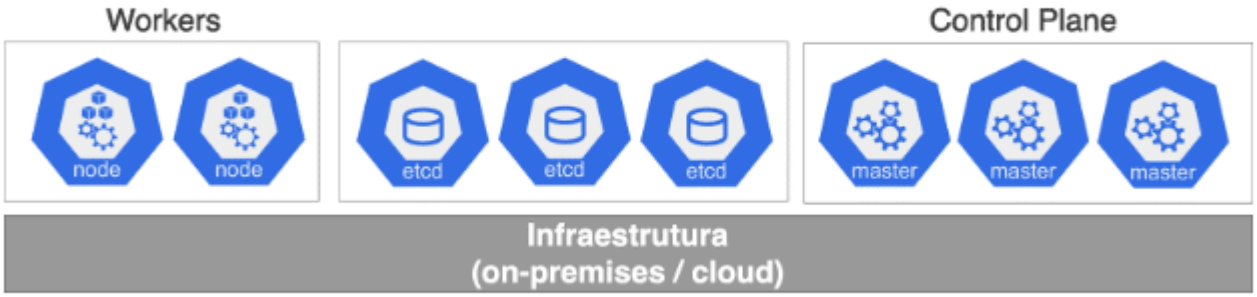
Já o estado desejado descreve como o time ou a pessoa responsável pela aplicação deseja que ela esteja naquele momento.

O Kubernetes implementa uma série de *loops* que ficam constantemente verificando se o estado atual é igual ao estado desejado. Esse papel é desempenhado pelos chamados *Controllers*.

Quando um *controller* identifica que o estado atual é diferente do estado desejado, ele aciona outros componentes do sistema para fazer novamente com que o estado atual se iguale ao estado desejado.

Todo esse processo de monitoramento e gestão do estado da aplicação, sem contar a execução da aplicação em si, exige uma série de componentes. É por isso que a arquitetura de um ambiente Kubernetes é baseada em um cluster de máquinas.

A arquitetura Kubernetes: Como funciona?



O Kubernetes é composto por uma série de componentes, cada um com um propósito diferente. Para garantir que exista uma separação de responsabilidades e que o sistema seja resiliente, o **K8s** utiliza um cluster de máquinas para ser executado.

As máquinas de um cluster são separadas em três tipos:
Node

O primeiro tipo é chamado de *Node*. O papel de um *Node* é executar os contêineres que encapsulam as aplicações sendo gerenciadas pelo K8s.

Quando você faz o *deploy* de uma aplicação em um cluster K8s, essa aplicação vai ser executada em um dos *Nodes* do cluster. O conjunto de *Nodes* forma o que chamamos de *Workers*.
etcd

O segundo tipo de nó é o *etcd*. O *etcd* é, na verdade, o nome da **base de dados distribuída** que é utilizada para armazenar tudo o que está acontecendo dentro do cluster, incluindo o estado da aplicação.

Em ambientes de produção, um bom gerenciamento desses nós é essencial para garantir que o *cluster* esteja sempre disponível.
Master

Finalmente, o último tipo de nó é o que chamamos de *Master*. É nesse tipo de nó que os principais componentes do Kubernetes são executados, como o *Scheduler*, o qual tem a responsabilidade de controlar a alocação de recursos no cluster.

O conjunto de nós *Master* forma o que pode ser considerado o cérebro de um cluster Kubernetes: o *Control Plane*.

Control Plane

O *Control Plane* do Kubernetes pode ser considerado o cérebro de um cluster. Ele é responsável por gerenciar os principais componentes do sistema e garantir que tudo está funcionando de acordo com o estado desejado da aplicação.

Para facilitar a representação desse estado, o K8s trabalha com uma abstração chamada de *Object*. Um *Object* representa parte do estado da aplicação e quando o seu estado atual não é o estado desejado, mudanças são aplicadas para que os dois estados se igualem novamente.

Existem diversos tipos de *Objects* em um ambiente Kubernetes, mas alguns deles são essenciais para entendermos como um cluster funciona.

Pod

Na seção sobre aplicações *cloud-native*, uma das principais características é que essas aplicações utilizam contêineres para encapsular seus microsserviços. No entanto, quando falamos sobre aplicações sendo executadas em um cluster Kubernetes, não falamos sobre contêineres diretamente, mas sim sobre *Pods*.

Pods são a unidade básica de um um cluster **K8s**. Elas encapsulam um ou mais contêineres de uma aplicação e representam um processo dentro do cluster. Quando fazemos o *deploy* de uma aplicação no K8s, estamos criando uma ou mais *Pods*.

No entanto, *Pods* são efêmeras, ou seja, elas são criadas e destruídas de acordo com as necessidades do cluster.

Para garantir que o acesso a um microsserviço esteja sempre disponível, existe um *Object* chamado *Service* que encapsula uma ou mais *Pods* e é capaz de encontrá-las dinamicamente em qualquer *Node* do cluster.
Deployment

Esse tipo de *Object* oferece uma série de funcionalidades que automatizam todos aqueles passos que descrevemos de um cenário típico de desenvolvimento de software, com *deploys* manuais ou semi-automatizados de uma aplicação.

Utilizando *Deployments*, nós podemos descrever qual o estado desejado da nossa aplicação e um *Deployment controller* vai se encarregar de transformar o estado atual no estado desejado, caso eles sejam diferentes.

E por falar em descrever o estado desejado da nossa aplicação, é hora de entendermos como isso é feito em um ambiente **Kubernetes**.

Aprendendo kubernetes: como criar aplicações?

Quando estamos utilizando um cluster Kubernetes, existem duas formas de aplicarmos mudanças ao estado atual de uma aplicação, ou seja, de mudarmos sua configuração.

A abordagem tradicional e que talvez você esteja mais acostumado é o chamada de **Configuração Imperativa**, onde dizemos como cada mudança deve ser feita.

Por exemplo, imagine que você queira mudar o número de réplicas de uma determinada *Pod* de 3 para 4.

Na abordagem imperativa, você enviaria comandos diretamente para a **API** do K8s dizendo que você quer alterar o número de réplicas de 3 para 4. Mas como fazer isso em uma aplicação com dezenas de microsserviços?

E se, enquanto você estava alterando cada um deles, algo aconteceu e você só teve tempo de aplicar as mudanças em metade das *Pods*. Quais são as implicações que uma mudança como essa poderia causar? Se algo começar a dar errado, como os membros do seu time vão saber o que já foi alterado e o que ainda não foi?

Talvez você tenha passado um parâmetro errado em um dos comandos e agora a aplicação está fora do ar.

Configuração declarativa

É para evitar esse tipo de problema que o Kubernetes suporta o que é chamado de **Configuração Declarativa**.

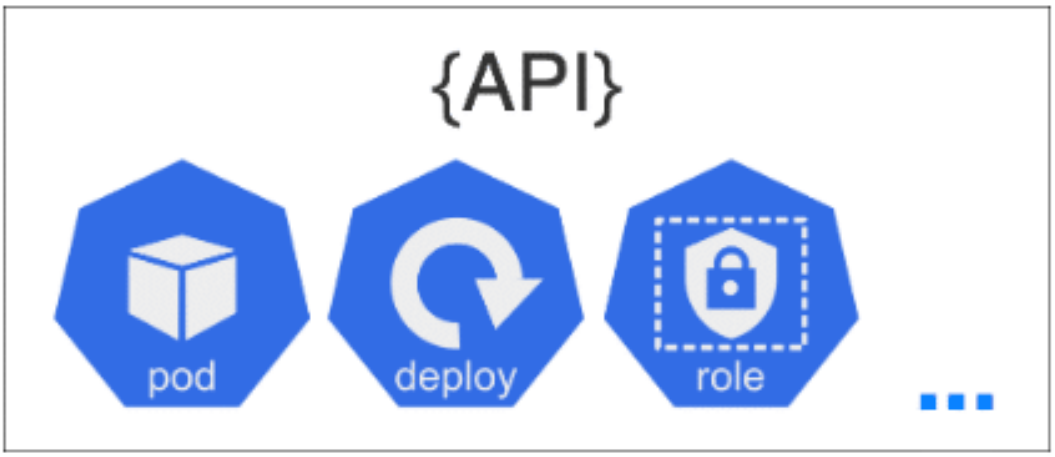
Em uma abordagem declarativa, nós não dizemos como uma mudança deve ser feita, mas apenas qual mudança deve ser feita.

O sistema, no nosso caso o *Control Plane* do K8s, vai decidir qual é a melhor forma de aplicar aquela mudança e tornar o estado atual da aplicação igual ao estado desejado.

Se nós fossemos fazer a mesma mudança do exemplo anterior, alterar o número de réplicas de uma *Pod* de 3 para 4 de uma forma declarativa, bastaria alterarmos o valor do campo “replicas” no exemplo abaixo e enviarmos esse arquivo YAML para a API do K8s.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # só precisamos alterar o valor do campo abaixo
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```

Entendendo a API do Kubernetes



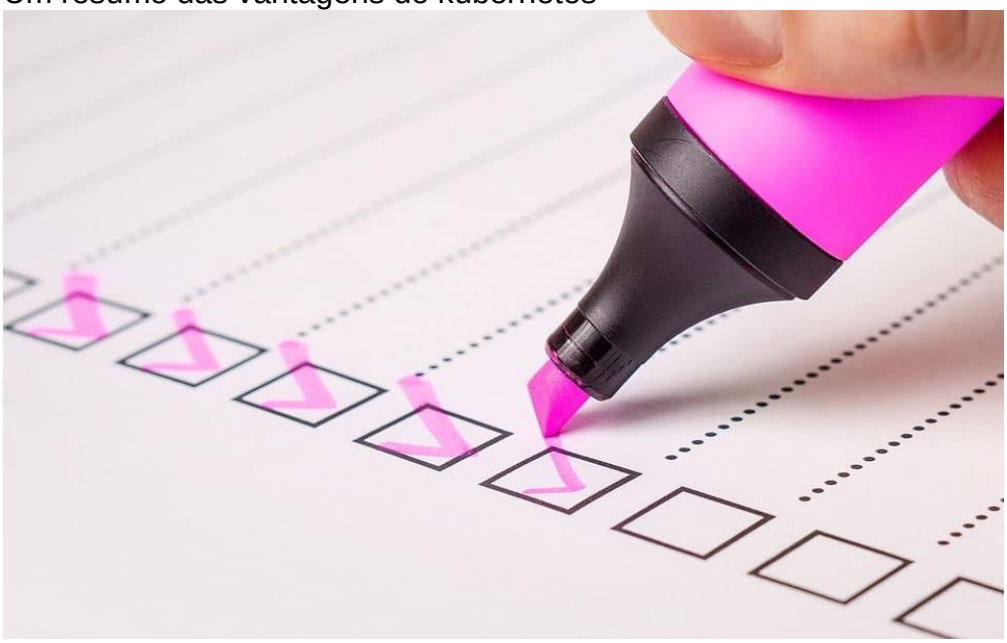
A API do Kubernetes é um dos principais elementos do *Control Plane*. É através dela que conseguimos interagir com todos componentes de um cluster K8s, seja pela linha de comando ou pela interface web.

Além disso, é a API quem define os diferentes *Objects* que fazem parte do ecossistema do K8s.

Quando enviamos uma alteração de estado, seja de forma imperativa ou declarativa, a API cria o que é chamado de *Record of Intent* (Registro de Intenção). Dependendo do *Object* que está sendo alterado nesse *Record of Intent*, um *Controller* específico vai detectar que o estado desejado foi alterado e vai reagir para aplicar as mudanças necessárias.

Como existem diversos tipos de *Objects* no contexto do Kubernetes, a API pode parecer complexa quando vista pela primeira vez. Para facilitar a gestão e evolução da API, os *Objects* foram agrupados em diferentes categorias, como *Core*, *Apps* e *Storage*.

Esses grupos são compostos por desenvolvedores da comunidade Kubernetes e são eles quem decidem como cada categoria vai evoluir. Isso mostra a natureza open source do projeto e como os usuários do sistema tem influência direta na sua evolução.
Um resumo das vantagens do kubernetes



O Kubernetes é uma ferramenta superpoderosa, mas que pode parecer bem complexa em um primeiro momento.

Existem diversos conceitos e componentes envolvidos no funcionamento de um cluster, mas são eles que fazem com o que o K8s seja o orquestrador mais utilizado no momento.

Para recapitularmos os principais pontos que vimos sobre o Kubernetes e organizar o nosso raciocínio, aqui vai um resumo:

- O K8s oferece uma plataforma completa para aplicações conhecidas como *cloud-native*;
- Os seus componentes são executados em um cluster composto por três tipos de nós: *Node*, *etcd* e *Master*;
- O conjunto de todos os nós *Master* forma o chamado *Control Plane*, o qual é responsável por controlar tudo o que acontece dentro do cluster e monitorar o estado da aplicação;
- Ele utiliza abstrações, como *Pods* e *Deployments*, chamadas de *Objects* para representar diferentes aspectos do estado de uma aplicação;
- Esse estado pode ser alterado de duas formas: imperativa ou declarativa. A forma declarativa é considerada a mais ideal e utiliza arquivos no formato YAML que são enviados para a API;
- A API do Kubernetes é a porta de entrada de um cluster, sendo utilizada tanto pela linha de comando quando pela interface web.

Agora que você já tem uma noção do que é o Kubernetes e da sua arquitetura, é hora de começar a aprender da melhor forma possível: colocando a mão na massa!

O que é um cluster Kubernetes e como construir um?

A melhor forma de aprender como o Kubernetes funciona é usando ele na prática. Para isso, existem 3 formas rápidas de você criar o seu primeiro cluster.

A primeira, e mais fácil, é utilizando o site: **Play With Kubernetes**.

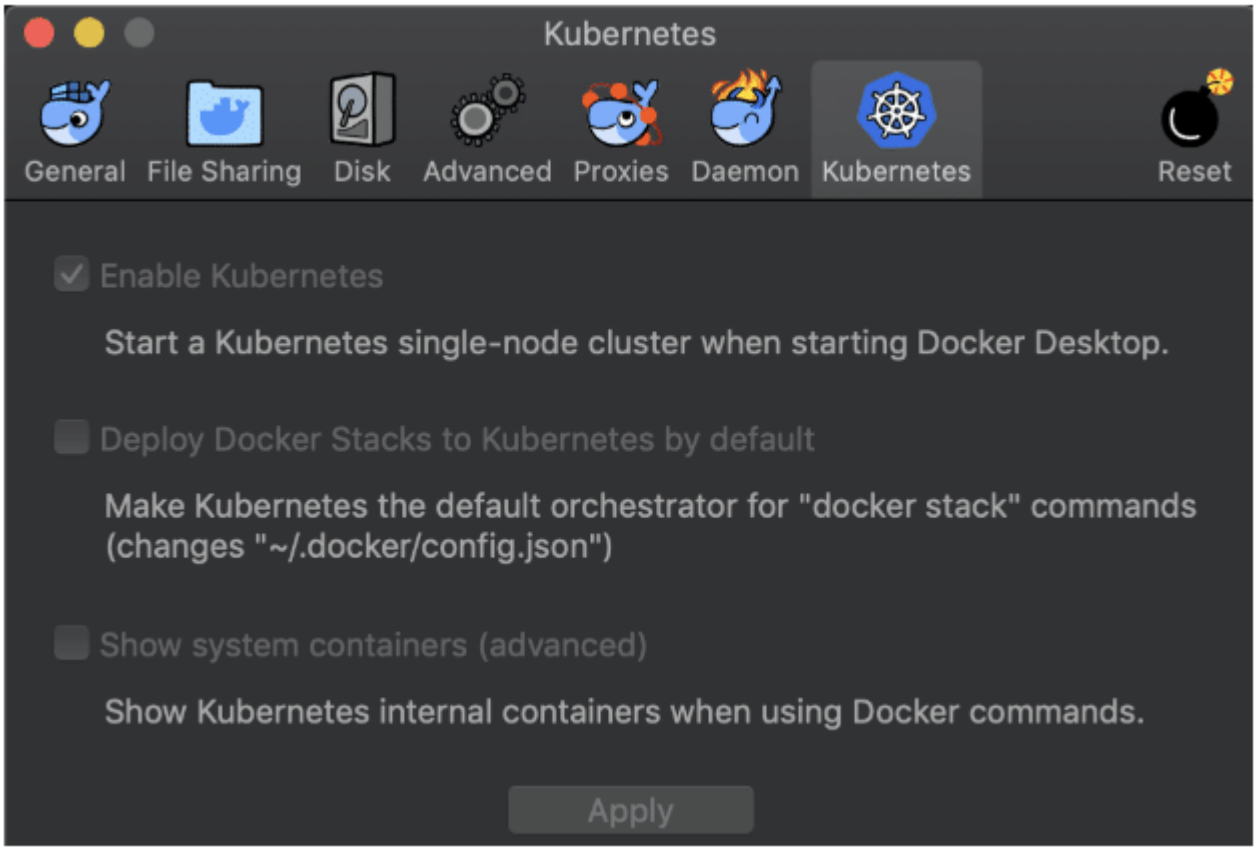
O *Play with K8s* oferece um cluster Kubernetes que você pode acessar através do seu browser. A ideia é que você tenha um ambiente de laboratório para brincar por até 4 horas. Depois disso, o ambiente é destruído, mas você sempre pode criar um novo.

Outra opção é utilizar a função Kubernetes disponível no **Docker**.

Com o avanço do Kubernetes, o Docker criou uma opção que possibilita a criação de um cluster K8s na sua máquina.

Para isso, basta instalar o Docker seguindo as **instruções para o seu sistema operacional**, clicar com o botão direito no ícone do Docker e selecionar Preferências.

Nesse menu você deve achar a opção chamada Kubernetes, como na imagem abaixo.



Finalmente, a terceira opção é utilizar os serviços oferecidos pelos provedores de nuvem pública.

Com a adoção do Kubernetes crescendo a cada dia, era natural que eles não ficariam de fora.

Atualmente, os principais serviços são:

- **Amazon EKS** da AWS;
- **Kubernetes Engine** do Google Cloud Platform; e
- **AKS** da Azure (Microsoft).

Essas três opções oferecem um cluster gerenciado, livrando você de se preocupar com diversos aspectos, como quantidade de nós *etcd* e *Master*.

E aí estão, três formas de você criar o seu primeiro cluster e começar a aprender Kubernetes na prática.