

# Série de Certificação GCP: 3.2 Implantando e implementando recursos do Kubernetes Engine



Prashanta Paudel

25 de outubro de 2018 · 28 min de leitura

O Kubernetes é um tema quente na computação em nuvem no momento. As empresas estão usando contêineres para implantação em vez de VMs da maneira tradicional. É um bom momento para os kubernetes, então vamos aprendê-lo corretamente.

*O Kubernetes é uma ferramenta de gerenciamento de contêineres de código aberto. Ele faz a tarefa de implantação, dimensionamento e balanceamento de carga de contêineres.*

**O Kubernetes não é uma plataforma de contêineres**, mas uma solução de gerenciamento de vários contêineres.

Como o kubernetes é uma ferramenta de gerenciamento de contêineres, é bom saber o que é um contêiner e o que é um estivador?

## Recipientes

Os contêineres são uma maneira melhor de desenvolver e implantar o aplicativo.



Recipientes-simbólicos

Um contêiner é um software empacotado em Unidades padronizadas para desenvolvimento, expedição e implantação.

Um container é uma unidade padrão de software que empacota o código e todas as suas dependências para que o aplicativo seja executado de maneira rápida e confiável em um único ambiente de computação.

Uma imagem de contêiner é um pacote de software leve, autônomo e executável, que inclui tudo o que é necessário para executar um aplicativo: código, tempo de execução, ferramentas do sistema, bibliotecas do sistema e configurações.

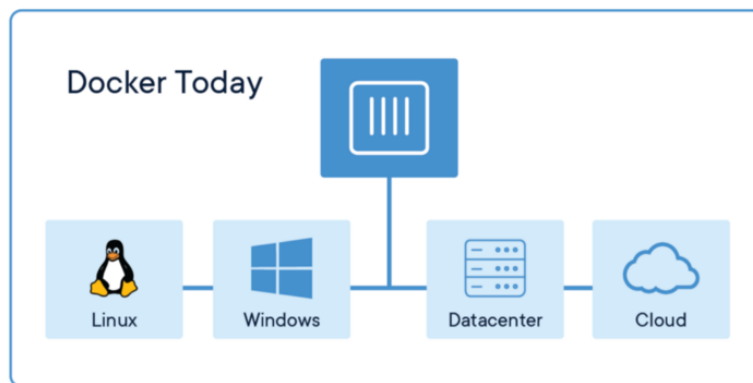


Recipiente

Antes de os contêineres serem implantados, eles são apenas imagens de contêiner. As imagens de contêiner se tornam contêineres quando são executadas no mecanismo do Docker.

O software com contêineres sempre será executado mesmo independentemente da infraestrutura em que eles residem.

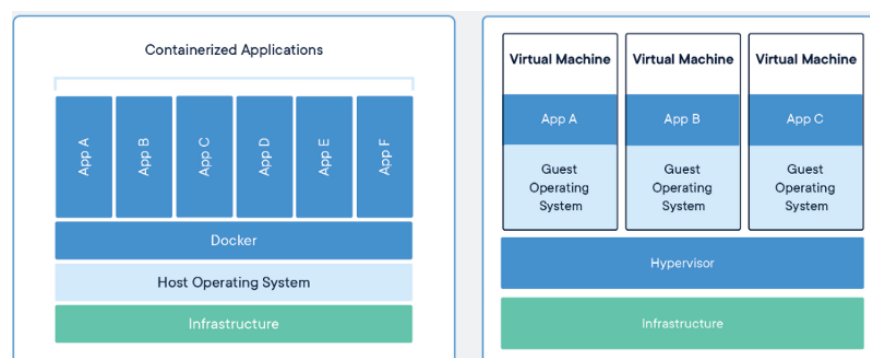
A tecnologia de contêineres Docker foi lançada em 2013 como um mecanismo Docker de código aberto. Agora ele é usado em todos os lugares, do Linux, do Windows às nuvens.



contêineres docker hoje

A tecnologia Docker é voltada para desenvolvedores e operadores de sistemas, pois separa dependências de aplicativos e infraestrutura.

## Contêineres e VMs



referência: <https://www.docker.com/resources/what-container>

CONTAINERS	VIRTUAL MACHINES
Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.	Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, the application, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

referência: <https://www.docker.com/resources/what-container>

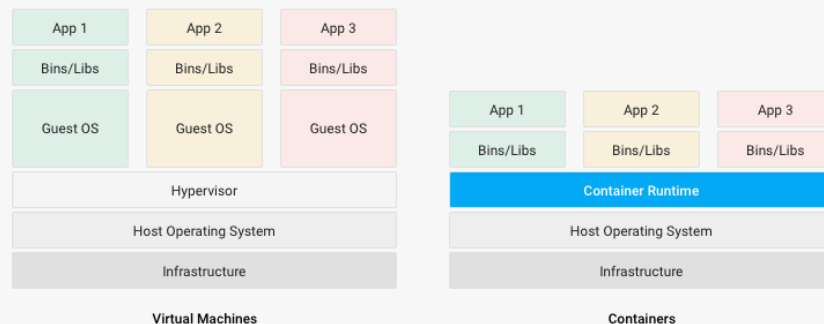
A principal diferença entre contêineres e VMs é que as máquinas virtuais virtualizam o hardware e o sistema operacional, enquanto os contêineres virtualizam apenas o kernel do sistema operacional.

Na verdade, poucos de vocês sabem disso, mas a maioria de vocês tem usado contêineres há anos. O Google tem sua própria tecnologia de contêineres de código aberto, Imctfy (Let Me Contain That For You). Toda vez que você usa algumas das funcionalidades do Google - Pesquisa, Gmail, Google Docs, o que for -, você recebe um novo contêiner.

## Containers 101: What are containers?

Containers offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run. This decoupling allows container-based applications to be deployed easily and consistently, regardless of whether the target environment is a private data center, the public cloud, or even a developer's personal laptop. Containerization provides a clean separation of concerns, as developers focus on their application logic and dependencies, while IT operations teams can focus on deployment and management without bothering with application details such as specific software versions and configurations specific to the app.

For those coming from virtualized environments, containers are often compared with virtual machines (VMs). You might already be familiar with VMs: a guest operating system such as Linux or Windows runs on top of a host operating system with virtualized access to the underlying hardware. Like virtual machines, containers allow you to package your application together with libraries and other dependencies, providing isolated environments for running your software services. As you'll see below however, the similarities end here as containers offer a far more lightweight unit for developers and IT Ops teams to work with, carrying a myriad of benefits.



referência: <https://cloud.google.com/containers/>

## Why Containers?

Instead of virtualizing the hardware stack as with the virtual machines approach, containers virtualize at the operating system level, with multiple containers running atop the OS kernel directly. This means that containers are far more lightweight: they share the OS kernel, start much faster, and use a fraction of the memory compared to booting an entire OS.

There are many container formats available. Docker is a popular, open-source container format that is supported on Google Cloud Platform and by Google Kubernetes Engine.

### Consistent Environment

Containers give developers the ability to create predictable environments that are isolated from other applications. Containers can also include software dependencies needed by the application, such as specific versions of programming language runtimes and other software libraries. From the developer's perspective, all this is guaranteed to be consistent no matter where the application is ultimately deployed. All this translates to productivity: developers and IT Ops teams spend less time debugging and diagnosing differences in environments, and more time shipping new functionality for users. And it means fewer bugs since developers can now make assumptions in dev and test environments they can be sure will hold true in production.

### Run Anywhere

Containers are able to run virtually anywhere, greatly easing development and deployment: on Linux, Windows, and Mac operating systems; on virtual machines or bare metal; on a developer's machine or in data centers on-premises; and of course, in the public cloud. The widespread popularity of the [Docker image format](#) for containers further helps with portability. Wherever you want to run your software, you can use containers.

### Isolation

Containers virtualize CPU, memory, storage, and network resources at the OS-level, providing developers with a sandboxed view of the OS logically isolated from other applications.

### Why Sandbox anyway?

Containers silo applications from each other unless you explicitly connect them. That means you don't have to worry about conflicting dependencies or resource contention — you set explicit resource limits for each service. Importantly, it's an additional layer of security since your applications aren't running directly on the host operating system.

referência: <https://cloud.google.com/containers/>

## From Code to Applications

Containers allow you to package your application and its dependencies together into one succinct manifest that can be version controlled, allowing for easy replication of your application across developers on your team and machines in your cluster.

Just as how software libraries package bits of code together, allowing developers to abstract away logic like user authentication and session management, containers allow your application as a whole to be packaged, abstracting away the operating system, the machine, and even the code itself. Combined with a service-based architecture, the entire unit that developers are asked to reason about becomes much smaller, leading to greater agility and productivity. All this eases development, testing, deployment, and overall management of your applications.

### Monolithic to Service Based Architecture

Containers work best for service based architectures. Opposed to monolithic architectures, where every pieces of the application is intertwined — from IO to data processing to rendering — service based architectures separate these into separate components. Separation and division of labor allows your services to continue running even if others are failing, keeping your application as a whole more reliable.

Componentization also allows you to develop faster and more reliably; smaller codebases are easier to maintain and since the services are separate, it's easy to test specific inputs for outputs.

Containers are perfect for service based applications since you can health check each container, limit each service to specific resources and start and stop them independently of each other.

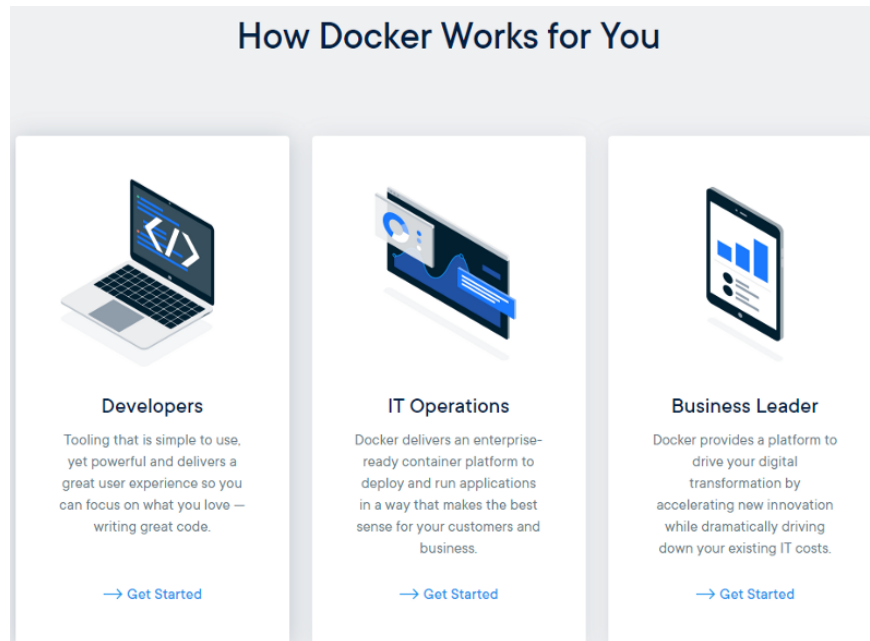
And since containers abstract the code away, containers allow you to treat separate services as black boxes, further decreasing the space a developer needs to be concerned with. When developers work on services that depends on another, they can easily start up a container for that specific service without having to waste time setting up the correct environment and troubleshooting beforehand.

referência: referência: <https://cloud.google.com/containers/>

## Docker

O Docker fornece liberdade de infraestrutura para que desenvolvedores e TI criem, gerenciem e protejam aplicativos essenciais aos negócios sem o medo da tecnologia ou do bloqueio de infraestrutura.

*O Docker também é uma organização que desenvolveu a tecnologia de contêineres docker.*



referência: <https://www.docker.com/why-docker>

Os contêineres Docker são a tecnologia de capacitação em nuvem de crescimento mais rápido e impulsionam uma nova era de computação e arquitetura de aplicativos com sua abordagem leve de agrupar aplicativos e dependências em pacotes de aplicativos isolados, mas altamente portáteis.

Os contêineres, por si só, não são suficientes para fornecer valor na escala de sua empresa e não abordam diretamente as necessidades de conformidade, segurança e operacionais de sua organização.

Por que usar contêineres Docker?

- As VMs são volumosas em requisitos de sistema e precisam de mais recursos, mas os contêineres não exigem grande parte dos recursos.
- Os contêineres do Docker suportam a integração contínua / implantação contínua (CI / CD) que incentiva os desenvolvedores a integrar seu código no repositório compartilhado no início.
- Os contêineres do Docker são fáceis de implantar na nuvem.

- Ele incorpora facilmente na maioria dos aplicativos DevOps, incluindo Puppet, Chef, Vagrant e Ansible.
- Jenkins um programa de CI / CD de código aberto, para automatizar a entrega de novos softwares em contêineres.
- 10 ou 100 de contêineres para balanceamento de carga do tráfego e garantia de alta disponibilidade.

## Orquestração de Contêineres

A orquestração de contêineres é o processo de gerenciamento de contêineres na nuvem.

As principais ferramentas de orquestração de contêineres são

- Enxame de Docker
- Kubernetes
- Mesosfera

This embedded content is from a site that does not comply with the Do Not Track (DNT) setting now enabled on your browser.

Please note, if you click through and view it anyway, you may be tracked by the website hosting the embed.

**[Learn More about Medium's DNT policy](#)**

## DRAWBACKS

Isso, por sua vez, significa que uma coisa que os hipervisores da VM podem fazer que o contêiner não pode é usar sistemas operacionais ou kernels diferentes. Portanto, por exemplo, você pode usar o Microsoft Azure para executar as duas instâncias do Windows Server 2012 e do SUSE Linux Enterprise Server ao mesmo tempo. Com o Docker, todos os contêineres devem usar o mesmo sistema operacional e kernel.



Por outro lado, se tudo o que você deseja fazer é obter a maioria das instâncias de aplicativos do servidor em execução na menor quantidade de hardware, não se preocuparia com a execução de várias VMs do sistema operacional. Se várias cópias do mesmo aplicativo forem o que você deseja, você vai adorar os contêineres.

## Kubernetes



referência: Wikipedia

O Kubernetes é um sistema de orquestração de contêineres de software livre para automatizar a implantação, o dimensionamento e o gerenciamento de aplicativos em contêiner.

Foi originalmente criado pelo Google e agora é mantido pela Cloud Native Computing Foundation. O objetivo é fornecer uma “plataforma para automatizar a implantação, o dimensionamento e as operações de contêineres de aplicativos no cluster de hosts.

### História

A Kubernetes foi fundada por Joe Beda, Brendan Burns e Craig McLuckie, e rapidamente se juntou a outros engenheiros do Google, incluindo Brian Grant e Tim Hockin, e foi **anunciada pela primeira vez pelo Google em meados de 2014**. Seu desenvolvimento e design são fortemente **influenciados pelo sistema Borg do Google** e por muitos dos principais colaboradores do projeto anteriormente desenvolvido em Borg. O codinome original do Kubernetes no Google era o Projeto Sete, uma referência ao personagem de Star Trek, Seven

of Nine, que é um Borg 'mais amigável'. Os sete raios da roda do logotipo do Kubernetes são um aceno para esse codinome.

**O Kubernetes v1.0 foi lançado em 21 de julho de 2015.** Junto com a versão v1.0 do Kubernetes, o **Google fez uma parceria com a Linux Foundation para formar a Cloud Native Computing Foundation (CNCF)** e ofereceu o Kubernetes como uma tecnologia de sementes. Em 6 de março de 2018, o Projeto Kubernetes alcançou o nono lugar em commits no GitHub, e o segundo lugar em autores e edições, logo abaixo do Linux Project.

## O que o Kubernetes pode fazer por você?

Com os serviços da Web modernos, os usuários esperam que os aplicativos estejam disponíveis 24 horas por dia, 7 dias por semana, e os desenvolvedores esperam implantar novas versões desses aplicativos várias vezes ao dia. A containerização ajuda a empacotar o software para atender a essas metas, permitindo que os aplicativos sejam lançados e atualizados de maneira fácil e rápida, sem tempo de inatividade. O Kubernetes ajuda você a garantir que esses aplicativos em contêiner sejam executados onde e quando quiser, além de ajudá-los a encontrar os recursos e as ferramentas de que precisam para trabalhar. O Kubernetes é uma plataforma de código aberto pronta para produção, projetada com a experiência acumulada do Google em orquestração de contêineres, combinada com as melhores ideias da comunidade.

## Kubernetes Clusters

**O Kubernetes coordena um cluster altamente disponível de computadores conectados para funcionar como uma única unidade.** As abstrações no Kubernetes permitem que você implante aplicativos em contêiner em um cluster sem vinculá-los especificamente a máquinas individuais. Para usar esse novo modelo de implantação, os aplicativos precisam ser empacotados de forma que os desacople de hosts individuais: eles precisam ser contêinerizados. Os aplicativos em contêiner são mais flexíveis e disponíveis do que nos modelos de implantação anteriores, nos quais os aplicativos foram instalados diretamente em máquinas específicas como pacotes profundamente integrados ao host. **O Kubernetes automatiza a distribuição e o agendamento de contêineres de aplicativos em um**

**cluster de maneira mais eficiente.** O Kubernetes é uma plataforma de código aberto e está pronto para produção.

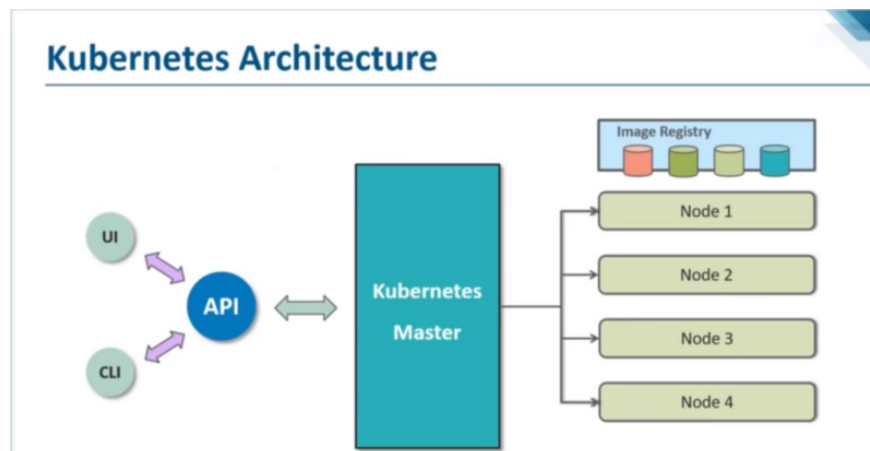
Um cluster do Kubernetes consiste em dois tipos de recursos:

- O **mestre** coordena o cluster
- **Nós** são os trabalhadores que executam aplicativos

**O mestre é responsável por gerenciar o cluster.** O mestre coordena todas as atividades em seu cluster, como agendar aplicativos, manter o estado desejado dos aplicativos, dimensionar aplicativos e implantar novas atualizações.

**Um nó é uma VM ou um computador físico que serve como uma máquina de trabalho em um cluster do Kubernetes.** Cada nó tem um Kubelet, que é um agente para gerenciar o nó e se comunicar com o mestre do Kubernetes. O nó também deve ter ferramentas para manipular operações de contêiner, como Docker ou rkt. Um cluster do Kubernetes que manipula o tráfego de produção deve ter um mínimo de três nós.

*Os mestres gerenciam o cluster e os nós são usados para hospedar os aplicativos em execução.*



referência: <https://dzone.com/articles/what-is-kubernetes-container-orchestration-tool>

Ao implantar aplicativos no Kubernetes, você diz ao mestre para iniciar os contêineres do aplicativo. O mestre agenda os contêineres para serem executados nos nós do cluster. **Os nós se comunicam com o mestre usando a API do Kubernetes**, que o mestre expõe. Os

usuários finais também podem usar a API do Kubernetes diretamente para interagir com o cluster.

### Kubernetes: Production-Grade Container Orchestration

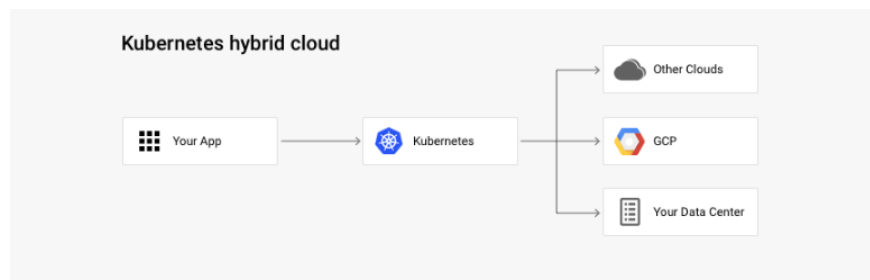
We've had so much success with our internal cluster management system [Borg](#) that we've taken what we've learned and put it into the open source project [Kubernetes](#). Now you and other companies can benefit from our decades of experience. Also known as "k8s," Kubernetes provides automated container orchestration – management of your machines and services for you – improving your reliability and reducing the time and resources you need to spend on DevOps, not to mention relief from the stress attached to these tasks.

Kubernetes makes everything associated with deploying and managing your application easier. Kubernetes automates rollouts and rollbacks, monitoring the health of your services to prevent bad rollouts before things go bad. It also continuously runs health checks against your services, restarting containers that fail or have stalled, and only advertising services to clients when it has confirmed they've started up successfully. Additionally, Kubernetes will automatically scale your services up or down based off of utilization, ensuring you're only running what you need, when you need it. Like containers, Kubernetes allows you to declaratively manage your cluster, allowing your setup to be version controlled and easily replicated.

#### Kubernetes Features

- ✓ Automated rollouts and roll backs
- ✓ Service health monitoring
- ✓ Automatic scaling of services
- ✓ Declarative management
- ✓ Deploy anywhere, including hybrid deployments

Perhaps most importantly, Kubernetes is built to be used anywhere, allowing you to orchestrate across on-site deployments to public clouds to hybrid deployments in between. This enables your infrastructure to reach your users where they're at, your applications to have higher availability, and your company to balance your security and cost concerns, all tailored to your specific needs.



## Your Cluster on Google

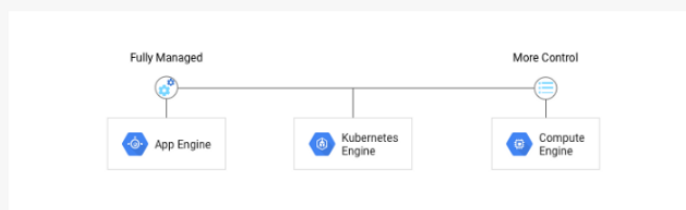
Of course, Kubernetes runs best on Google Cloud Platform. [Google Kubernetes Engine](#) is the premier managed Kubernetes solution that gets you quickly set up and production-ready.

Kubernetes Engine is fully managed by Google reliability engineers, the ones who know containers the best, ensuring your cluster is highly available and up-to-date. It integrates seamlessly with all GCP services, such as [Stackdriver](#) monitoring, diagnostics, and logging; [Identity and Access Management](#); and Google's best-in-class [networking](#) infrastructure.

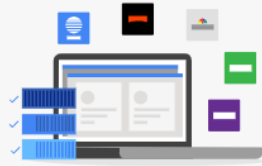
### Kubernetes Engine Features

- ✓ Managed open-source Kubernetes
- ✓ 99.5% SLA, and high availability with integrated multi-zone deployments
- ✓ Seamless integration of other GCP services
- ✓ Industry leading price per performance
- ✓ Flexible & interoperable with your on-premises clusters or other cloud providers
- ✓ Google-grade managed-infrastructure

But we love to give you options. Google Cloud Platform offers you a full spectrum for running your containers. From fully managed platform-as-a-service with [Google App Engine Flexible Environment](#) to cluster management with Kubernetes Engine to roll-it-yourself infrastructure on world-class price-to-performance [Google Compute Engine](#), you can find your ideal solution for running containers on Google Cloud Platform.



referência: <https://cloud.google.com/containers/>

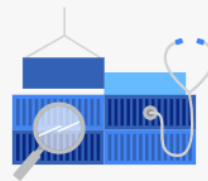


## Deploy a Wide Variety of Applications

Kubernetes Engine enables rapid application development and iteration by making it easy to deploy, update, and manage your applications and services. Kubernetes Engine isn't just for stateless applications either; you can attach persistent storage, and even run a database in your cluster. Simply describe the compute, memory, and storage resources your application containers require, and Kubernetes Engine provisions and manages the underlying cloud resources automatically. Support for hardware accelerators makes it easy to run Machine Learning, General Purpose GPU, High-Performance Computing, and other workloads that benefit from specialized hardware accelerators.

## Operate Seamlessly with High Availability

Control your environment from the built-in Kubernetes Engine dashboard in Google Cloud console. Use routine health checks to detect and replace hung, or crashed, applications inside your deployments. Container replication strategies, monitoring, and automated repairs help ensure that your services are highly available and offer a seamless experience to your users. Google Site Reliability Engineers (SREs) constantly monitor your cluster and its compute, networking, and storage resources so you don't have to, giving you back time to focus on your applications.



<https://cloud.google.com/kubernetes-engine/>

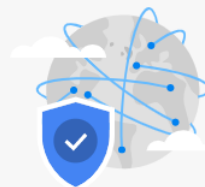



## Scale Effortlessly to Meet Demand

Go from a single machine to thousands: Kubernetes Engine autoscaling allows you to handle increased user demand for your services, keeping them available when it matters most. Then, scale back in the quiet periods to save money, or schedule low-priority batch jobs to use up spare cycles. Kubernetes Engine helps you get the most out of your resource pool.

## Run Securely on Google's Network

Connect to and isolate clusters no matter where you are with fine-grained network policies using Global Virtual Private Cloud (VPC) in Google Cloud. Use public services behind a single global anycast IP address for seamless load balancing. Protect against DOS and other types of edge attacks on your containers.





### Move Freely between On-premises and Clouds

Kubernetes Engine runs Certified Kubernetes ensuring portability across clouds and on-premises. There's no vendor lock-in: you're free to take your applications out of Kubernetes Engine and run them anywhere Kubernetes is supported, including on your own on-premises servers. You can tailor integrations such as monitoring, logging, and CI/CD using Google Cloud Platform (GCP) and third party solutions in the ecosystem.

## KUBERNETES ENGINE FEATURES

Run containerized applications on production-ready Kubernetes, managed by Google Cloud.

<b>Identity &amp; Access Management</b> Control access in the cluster with your Google accounts and role permissions.	<b>Stateful Application Support</b> Kubernetes Engine isn't just for 12-factor apps. You can attach persistent storage to containers, and even host complete databases.
<b>Hybrid Networking</b> Reserve an IP address range for your cluster, allowing your cluster IPs to coexist with private network IPs via <a href="#">Google Cloud VPN</a> .	<b>Docker Image Support</b> Kubernetes Engine supports the common Docker container format.
<b>Security and Compliance</b> Kubernetes Engine is backed by Google security team of over 750 experts and is both HIPAA and PCI DSS 3.1 compliant.	<b>Fully Managed</b> Kubernetes Engine clusters are fully managed by Google Site Reliability Engineers (SREs), ensuring your cluster is available and up-to-date.
<b>Integrated Logging &amp; Monitoring</b> Enable <a href="#">Stackdriver Logging</a> and <a href="#">Stackdriver Monitoring</a> with simple checkbox configurations, making it easy to gain insight into how your application is running.	<b>OS Built for Containers</b> Kubernetes Engine runs on <a href="#">Container-Optimized OS</a> , a hardened OS built and managed by Google.
<b>Auto Scale</b> Automatically scale your application deployment up and down based on resource utilization (CPU, memory).	<b>Private Container Registry</b> Integrating with <a href="#">Google Container Registry</a> makes it easy to store and access your private Docker images.
<b>Auto Upgrade</b> Automatically keep your cluster up to date with the latest release version of Kubernetes. Kubernetes release updates are quickly made available within Kubernetes Engine.	<b>Fast Consistent Builds</b> Use <a href="#">Google Cloud Build</a> to reliably deploy your containers on Kubernetes Engine without needing to setup authentication.
<b>Auto Repair</b> When auto repair is enabled, if a node fails a health check Kubernetes Engine initiates a repair process for that node.	<b>Workload Portability, on-premises and cloud</b> Kubernetes Engine runs Certified Kubernetes, enabling workload portability to other Kubernetes platforms across clouds and on-premises.
<b>Resource Limits</b> Kubernetes allows you to specify how much CPU and memory (RAM) each Container needs, which is used to better organize workloads within your cluster.	<b>GPU support<sup>BETA</sup></b> Kubernetes Engine supports GPU and makes it easy to run ML, GPGPU, HPC, and other workloads that benefit from specialized hardware accelerators.

Referência: <https://cloud.google.com/kubernetes-engine/>

## Pequeno tutorial no CLI para implantação de kubernetes

Bem-vindo ao CI / CD com o emulador de terminal tutorial GKE!

Neste tutorial, você criará um cluster do Kubernetes no GKE, criará uma implantação do Google Container Registry, aumentará a implantação e atualizará o código base.

### ETAPA 1: CRIAR UM CLUSTER

Digite o seguinte comando para criar um cluster:

```
gcloud container clusters create myCluster
$ gcloud container clusters create mycluster
Criando cluster mycluster... done.
Criado [ https://container.googleapis.com/v1/projects/kubernetes-terminal-simulator/zones/us-west1-a/clusters/mycluster].
Entrada kubeconfig gerada para mycluster.
NOME DA ZONA MASTER_VERSION MASTER_IP MACHINE_TYPE
NODE_VERSION N
UM_NODES STATUS
mycluster us-west1-a 1.7.8-gke.0 198.51.100.41 n1-standard-1 1.7.8-gke.0 3 RUNNING
```

## PASSO 2: GIRAR UMA INSTALAÇÃO A PARTIR DO REGISTRO DE RECIPIENTES DO GOOGLE.

Agora vamos extrair e executar uma implantação do Kubernetes do Google Container Register

```
:
app de execução do kubectl - image gcr.io/google-samples/hello-app:1.0
$ app de execução do kubectl - image gcr.io/google-samples/hello-app:1.0
"aplicativo" de implantação criado
```

## PASSO 3: DEPLOYMENT DE ESCALA.

Devemos aumentar a implantação em três pods para redundância: aplicativo de implantação de escala kubectl - aplicativo de implantação de escala

```
$ kubectl de réplicas 3 - aplicativo de implantação de réplicas 3 dimensionado
```

## PASSO 4: PORTAS ABERTAS.

Agora que o site está implantado, estamos prontos para abri-lo para o mundo:

```
kubectl expõe aplicativo de implementação - porta 80 - tipo = LoadBalancer
$ kubectl expõe aplicativo de implantação - porta 80 - tipo = Serviço de balanceamento de carga "app" exposto
```

## ETAPA 5: CONFIRMAR DEPLOYMENT.



Confirme a implantação do serviço:

kubectl obtenha o aplicativo de serviço

\$ kubectl obtenha o aplicativo de serviço

NOME TIPO CLUSTER-IP PORTA (S) EXTERNO-IP AGE

aplicativo undefined 203.0.113.51 203.0.113.16 80 - tipo: 31834 /

TCP 1m

PASSO 6: CONFIRME DISPONIBILIDADE.

Confirme se o serviço está implantado e disponível:

curl <http://203.0.113.16:80--type>

\$ curl <http://203.0.113.16:80--type>

Hello, world!

Versão: 1.0.0

Nome do host: app-970732273-p6j37

PASSO 7: CÓDIGO DE ATUALIZAÇÃO.

Vamos imaginar que o site foi atualizado e precisamos implantar a base de código de atualização d para produção - é fácil, basta digitar o seguinte comando:

kubectl definir implantação de imagem aplicativo app = gcr.io /

google-samples / hello-app: 2.0

\$ kubectl definir aplicativo de implantação de imagem app = gcr.io /

google-samples / hello-app: imagem de

implantação de "aplicativo" 2.0 atualizada

ETAPA 8: CONFIRMAR ATUALIZAÇÃO.

Parabéns, o seu novo site está em funcionamento e automaticamente implementado em

todos os pods em execução! Vamos confirmar:

curl <http://203.0.113.16:80--type>

\$ curl <http://203.0.113.16:80--type>

Olá, mundo!

Versão: 2.0.0

Nome do host: app-970732273-hnhzc

TUTORIAL COMPLETE!

## Visão Geral do GKE

Googles O kubernetes Engine fornece um ambiente gerenciado para implantar, gerenciar e escalonar seu aplicativo containerizado usando a

infraestrutura do google

O ambiente geralmente consiste em instâncias de mecanismos de computação do Google agrupadas para formar um cluster.

## Orquestração de Cluster com GKE

Os clusters GKE são ativados pelo sistema de gerenciamento de clusters de código aberto do Kubernetes. O Kubernetes fornece os mecanismos pelos quais você interage com seu cluster. Você usa os comandos e recursos do Kubernetes para implantar e gerenciar seus aplicativos, executar tarefas administrativas e definir políticas, além de monitorar a integridade de suas cargas de trabalho implementadas.

## Kubernetes no Google Cloud Platform

Quando você executa um cluster GKE, também obtém os benefícios dos recursos avançados de gerenciamento de cluster que o Google Cloud Platform oferece. Esses incluem:

- Balanceamento de carga do Google Cloud Platform para instâncias do Compute Engine
- Conjuntos de nós para designar subconjuntos de nós dentro de um cluster para flexibilidade adicional
- Escalonamento automático da contagem de instâncias do nó do seu cluster
- Atualizações automáticas para o software do nó do seu cluster
- Reparo automático do nó para manter a integridade e a disponibilidade do nó
- Registrando e monitorando com o Stackdriver para visibilidade em seu cluster

## Versões e recursos do Kubernetes

Os mestres de cluster GKE são atualizados automaticamente para executar novas versões do Kubernetes, pois essas versões se tornam estáveis, para que você possa aproveitar os recursos mais recentes do projeto Kubernetes de código aberto.

## Cargas de trabalho GKE

O GKE funciona com aplicativos contêinerizados: aplicativos empacotados em instâncias de espaço de usuário isoladas e independentes de hardware, por exemplo, usando o Docker. No GKE e no Kubernetes, esses contêineres, seja para aplicativos ou tarefas em lote, são chamados coletivamente de *cargas de trabalho*. Antes de implantar uma carga de trabalho em um cluster GKE, você deve primeiro empacotar a carga de trabalho em um contêiner.

O Google Cloud Platform fornece ferramentas de integração contínua e entrega contínua para ajudar você a criar e fornecer contêineres de aplicativos. Você pode usar o Google Cloud Build para criar imagens de contêiner (como o Docker) a partir de vários repositórios de código-fonte e o Google Container Registry para armazenar e exibir suas imagens de contêiner.

## Arquitetura de Cluster

No Google Kubernetes Engine, um *cluster* consiste em pelo menos um mestre de cluster e várias máquinas de trabalho chamadas de nós. Essas máquinas mestras e de nó executam o sistema de orquestração de clusters Kubernetes.

Um cluster é a base do GKE: os objetos do Kubernetes, que representam seus aplicativos em contêiner, são todos executados no topo de um cluster.

## Mestre de cluster

O *mestre de clusters* executa os processos do plano de controle do Kubernetes, incluindo o servidor da API do Kubernetes, o agendador e os controladores de recursos principais. O ciclo de vida do mestre é gerenciado pelo GKE quando você cria ou exclui um cluster. Isso inclui atualizações para a versão do Kubernetes em execução no mestre do cluster, que o GKE executa automaticamente ou manualmente, se você preferir fazer a atualização antes do agendamento automático.

## Mestre de cluster e a API do Kubernetes

O mestre é o ponto final unificado para o seu cluster. Todas as interações com o cluster são feitas por meio de chamadas da API do

Kubernetes, e o mestre executa o **processo** do **Kubernetes API Server** para lidar com essas solicitações. Você pode fazer chamadas da API do Kubernetes diretamente via HTTP / gRPC, ou indiretamente, executando comandos do cliente de linha de comando do Kubernetes ( `kubectl` ) ou interagindo com a interface do usuário no console do GCP.

O processo do servidor de API do mestre do cluster é o hub de toda a comunicação do cluster. Todos os processos de cluster internos (como nós de cluster, sistema e componentes, controladores de aplicativos) atuam como clientes do servidor de API; o servidor de API é a única “fonte de verdade” para todo o cluster.

## Interação entre mestre e nó

O mestre do cluster é responsável por decidir o que é executado em todos os nós do cluster. Isso pode incluir o agendamento de cargas de trabalho, como aplicativos em contêiner, e o gerenciamento do ciclo de vida, escalonamento e atualizações das cargas de trabalho. O mestre também gerencia os recursos de rede e armazenamento para essas cargas de trabalho.

O mestre e os nós também se comunicam usando as APIs do Kubernetes.

## Nós

Um cluster geralmente possui um ou mais *nós* , que são as máquinas de trabalho que executam seus aplicativos de contêiner e outras cargas de trabalho. As máquinas individuais são instâncias de VMs do Compute Engine que o GKE cria em seu nome quando você cria um cluster.

Cada nó é gerenciado a partir do mestre, que recebe atualizações no status auto-relatado de cada nó. Você pode exercer algum controle manual sobre o ciclo de vida do nó ou pode fazer com que o GKE execute reparos automáticos e atualizações automáticas nos nós do cluster.

Um nó executa os serviços necessários para suportar os contêineres do Docker que compõem as cargas de trabalho do cluster. Estes incluem o tempo de execução do Docker e o agente do nó do Kubernetes (

`kubelet` ) que se comunica com o mestre e é responsável por iniciar e executar contêineres do Docker agendados nesse nó.

No GKE, há também vários contêineres especiais que são executados como agentes por nó para fornecer funcionalidade, como coleta de log e conectividade de rede dentro do cluster.

## Tipo de máquina do nó

Cada nó é de um tipo de máquina padrão do Compute Engine . O tipo padrão é `n1-standard-1` , com 1 CPU virtual e 3,75 GB de memória. Você pode selecionar um tipo de máquina diferente ao criar um cluster .

## Imagens do sistema operacional do nó

Cada nó executa uma imagem especializada do SO para executar seus contêineres. Você pode especificar qual imagem do SO seus clusters e pools de nós usam.

## Plataforma mínima de CPU

Ao criar um cluster ou pool de nós, você pode especificar uma plataforma de CPU mínima de linha de base para seus nós. A escolha de uma plataforma de CPU específica pode ser vantajosa para cargas de trabalho avançadas ou com uso intensivo de computação. Para mais informações, consulte a Plataforma de CPU Mínima.

## Vagem

## O que é um Pod?

Os **pods** são os menores e mais básicos objetos implantáveis no Kubernetes. Um Pod representa uma única instância de um processo em execução no cluster.

Os pods contêm um ou mais *contêineres* , como contêineres do Docker. Quando um Pod executa vários contêineres, os contêineres são gerenciados como uma única entidade e compartilham os recursos do Pod. Geralmente, a execução de vários contêineres em um único Pod é um caso de uso avançado.

Os pods também contêm recursos de rede e armazenamento compartilhados para seus contêineres:

- **Rede:** os pods recebem automaticamente endereços IP exclusivos. Os contêineres de pod compartilham o mesmo namespace de rede, incluindo o endereço IP e as portas de rede. Recipientes em um Pod se comunicam dentro do Pod on `localhost`.
- **Armazenamento:** os pods podem especificar um conjunto de volumes de armazenamento compartilhado que podem ser compartilhados entre os contêineres.

Os pods são executados nos **nós** do cluster. Uma vez criado, um Pod permanecerá em seu nó até que seu processo seja concluído, o Pod seja excluído, o Pod seja **despejado** do nó devido à falta de recursos ou o nó falhe. Se um nó falhar, os Pods no nó serão agendados automaticamente para exclusão.

## Ciclo de vida do pod

Vagens são efêmeras. Eles não são projetados para serem executados para sempre e, quando um Pod é finalizado, ele não pode ser recuperado. Em geral, os Pods não desaparecem até serem excluídos por um usuário ou por um controlador.

Vagens não "curam" ou se reparam. Por exemplo, se um Pod for agendado em um nó que falhar mais tarde, o Pod será excluído. Da mesma forma, se um Pod for despejado de um nó por qualquer motivo, o Pod não se substituirá.

Cada Pod possui um `PodStatus` objeto de API, representado pelo `status` campo de um Pod. Os pods publicam sua *fase* no `status: phase` campo. A fase de um Pod é um resumo de alto nível do Pod em seu estado atual.

Quando você corre `kubectl get pod` para inspecionar um Pod em execução no cluster, um Pod pode estar em uma das seguintes fases possíveis:

- **Pendente:** o pod foi criado e aceito pelo cluster, mas um ou mais de seus contêineres ainda não estão em execução. Esta fase inclui

o tempo gasto sendo agendado em um nó e o download de imagens.

- **Em execução:** o Pod foi ligado a um nó e todos os contêineres foram criados. Pelo menos um contêiner está em execução, está em processo de inicialização ou está reiniciando.
- **Sucedido:** Todos os contêineres no Pod foram finalizados com sucesso. Os Pods Terminados não são reiniciados.
- **Falhou:** todos os contêineres no Pod foram encerrados e pelo menos um contêiner terminou em falha. Um contêiner "falha" se sair com um status diferente de zero.
- **Desconhecido:** o estado do Pod não pode ser determinado.

## Padrões de uso de vagens

Os pods podem ser usados de duas maneiras principais:

- **Pods que executam um único contêiner.** O padrão mais simples e mais comum do Google Pod é um único contêiner por conjunto, em que o único contêiner representa um aplicativo inteiro. Nesse caso, você pode pensar em um pod como um wrapper.
- **Pods que executam vários contêineres que precisam funcionar juntos.** Os pods com vários contêineres são usados principalmente para oferecer suporte a programas co-gerenciados e co-gerenciados que precisam compartilhar recursos. Esses contêineres co-localizados podem formar uma única unidade de serviço coesa - um contêiner servindo arquivos de um volume compartilhado enquanto outro contêiner atualiza ou atualiza esses arquivos. O Pod agrupa esses contêineres e recursos de armazenamento como uma única entidade gerenciável.

Cada Pod é destinado a executar uma única instância de um determinado aplicativo. Se você quiser executar várias instâncias, deverá usar um Pod para cada instância do aplicativo. Isso geralmente é chamado de *replicação*. Os Pods Replicados são criados e gerenciados como um grupo por um controlador, como uma Implantação.

## Terminação Pod

Os pods terminam normalmente quando seus processos são concluídos. Por padrão, todas as terminações são graciosas em 30 segundos.

Você pode excluir manualmente um Pod usando o `kubectl delete` comando. O `--grace-period` sinalizador do comando permite substituir o período de carência padrão.

## Desdobramento, desenvolvimento

Esta página descreve os objetos do Kubernetes Deployment e seu uso no Google Kubernetes Engine.

## O que é uma implantação?

*Implantações* representam um conjunto de vários Pods idênticos sem identidades exclusivas. Uma Implantação executa várias réplicas de seu aplicativo e substitui automaticamente quaisquer instâncias que falharem ou não responderem. Dessa forma, as Implantações ajudam a garantir que uma ou mais instâncias do seu aplicativo estejam disponíveis para atender às solicitações do usuário. As implantações são gerenciadas pelo controlador de implantação do Kubernetes.

As implantações usam um modelo Pod , que contém uma especificação para seus pods. A especificação Pod determina como cada Pod deve se parecer: quais aplicativos devem ser executados dentro de seus contêineres, quais volumes os Pods devem montar, seus rótulos e muito mais.

Quando um modelo de Pod de Implantação é alterado, novos Pods são criados automaticamente, um de cada vez.

## Padrões de uso

As implantações são adequadas para aplicativos sem estado que usam volumes ReadOnlyMany ou ReadWriteMany montados em várias réplicas, mas não são adequados para cargas de trabalho que usam volumes ReadWriteOnce. Para aplicativos com informações de estado usando volumes ReadWriteOnce, use StatefulSets . Os StatefulSets são projetados para implantar aplicativos com estado e aplicativos em cluster que salvam dados em armazenamento persistente, como discos permanentes do Compute Engine . Os StatefulSets são adequados para



implementar Kafka, MySQL, Redis, ZooKeeper e outros aplicativos que precisam de identidades únicas e persistentes e nomes de host estáveis.

## Criando Implantações

Você pode criar uma implantação usando os `kubectl run`, `kubectl apply` ou `kubectl create` comandos.

Uma vez criada, a Implantação garante que o número desejado de pods esteja em execução e disponível o tempo todo. A implantação substitui automaticamente os pods que falham ou são despejados de seus nós.

A seguir, um exemplo de um arquivo de manifesto de implantação no formato YAML:

```
apiVersion: apps / v1
tipo:
metadados de implantação :
  nome: nginx
spec:
  replicas: 3
  seletor:
    matchLabels:
      app: nginx
  template:
    metadados:
      rótulos:
        app: nginx
    spec:
      containers:
      - nome: nginx
        imagem: nginx: 1.7.9
        portas :
        - containerPort: 80
```

Neste exemplo:

- Um Deployment named `nginx` é criado, indicado pelo `metadata: name` campo.
- A implantação cria três pods replicados, indicados pelo `replicas` campo.
- O modelo ou `spec: template` campo do Pod indica que seus Pods estão rotulados `app: nginx`.

- A especificação do modelo Pod, ou `template: spec` campo, indica que os Pods executam um contêiner `nginx`, que executa a imagem do `nginx` [Docker Hub](#) na versão 1.7.9.
- A implantação abre a porta 80 para uso pelos pods.

Em suma, o modelo Pod contém as seguintes instruções para os Pods criados por esta Implementação:

- Cada Pod é rotulado.
- Crie um contêiner e nomeie-o `nginx`.
- Execute a `nginx` imagem na versão `1.7.9`.
- Abra uma porta `80` para enviar e receber tráfego.

Para mais informações sobre como criar Implantações, consulte [Criando uma Implantação](#).

## Atualizando Implantações

Você pode [atualizar uma Implantação](#) fazendo alterações na especificação do modelo do Pod de Implantação. Fazer alterações no campo de especificação aciona automaticamente um lançamento de atualização. Você pode usar `kubectl` a API do Kubernetes ou o menu Cargas de trabalho do GKE no Google Cloud Platform Console.

Por padrão, quando uma Implantação dispara uma atualização, a Implantação para os Pods, gradualmente reduz o número de Pods a zero, depois drena e finaliza os Pods. Em seguida, a Implantação usa o modelo atualizado do Pod para exibir novos Pods.

Os pods antigos não são removidos até que um número suficiente de novos pods sejam executados, e novos pods não são criados até que um número suficiente de pods antigos seja removido. Para ver em qual ordem os Pods são criados e removidos, você pode executar `kubectl describe deployments`.

As implantações podem garantir que pelo menos um menos do que o número desejado de réplicas esteja sendo executado, com no máximo um Pod indisponível. Da mesma forma, as implantações podem

garantir que pelo menos uma mais do que o número desejado de réplicas, com no máximo mais um Pod do que a execução desejada.

Você pode reverter uma atualização usando o `kubectl rollout undo` comando. Você também pode usar `kubectl rollout pause` para interromper temporariamente uma Implantação.

## Gerenciando Implantações

A seguir, uma lista de tarefas comuns de gerenciamento para Implantações:

- [Inspeccionar uma implantação](#)
- [Escale uma implantação](#)
- [Escalonar automaticamente uma implantação usando um `HorizontalPodAutoscaler` objeto](#)
- [Excluir uma implantação](#)

## Status e ciclo de vida

As implantações podem estar em um dos três estados durante seu ciclo de vida: progredindo, concluído ou com falha.

Um estado em *progresso* indica que a Implantação está em processo de executar suas tarefas, como exibir ou dimensionar seus Pods.

Um estado *concluído* indica que a Implementação concluiu com êxito suas tarefas, todos os seus Pods estão sendo executados com a especificação mais recente e estão disponíveis, e nenhum Pods antigo ainda está em execução.

Um estado com *falha* indica que a Implantação encontrou um ou mais problemas que impedem a conclusão de suas tarefas. Algumas causas incluem cotas ou permissões insuficientes, erros de recepção de imagem, intervalos de limite ou erros de tempo de execução. Para investigar o que causa uma falha na implantação, você pode executar `kubectl get deployment [DEPLOYMENT+NAME] -o yaml` e examinar as mensagens no `status: conditions` campo.

Você pode monitorar o progresso de uma implantação ou verificar seu status usando o `kubectl rollout status` comando.

-----

## Implantando um cluster do Kubernetes Engine

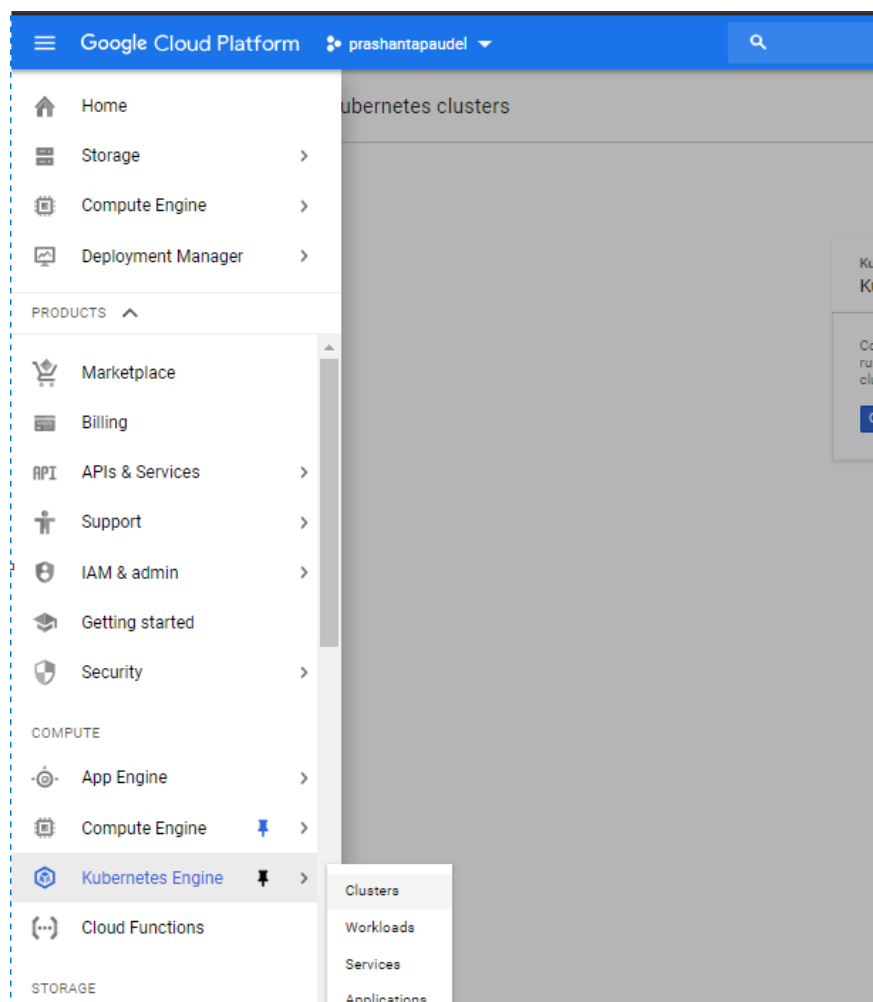
### Objetivos

Para empacotar e implantar seu aplicativo no GKE, você deve:

1. Empacote seu aplicativo em uma imagem do Docker
2. Execute o contêiner localmente em sua máquina (opcional)
3. Envie a imagem para um registro
4. Crie um cluster de contêiner
5. Implemente seu aplicativo no cluster
6. Exponha seu aplicativo à Internet
7. Amplie sua implantação
8. Implantar uma nova versão do seu aplicativo

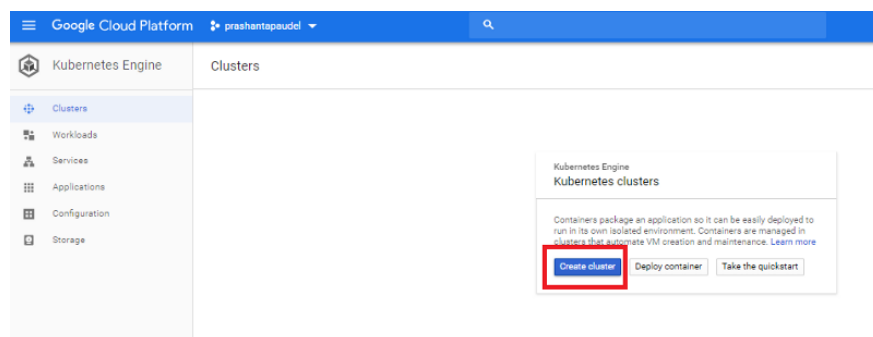
| *Do console do GCP*

Primeiro, acesse o Google Console, clique em Computar mecanismo e acesse o Kubernetes Dashboard



Painel do Kubernetes

Agora você vai pousar na página do cluster dentro do motor do kubernetes



Página de cluster

Agora, clique em Criar Cluster

✕ Create a Kubernetes cluster

**Cluster templates**

Select a template with preconfigured settings, or customize a template to suit your needs.

- ☐ Clone an existing cluster  
Select one of your existing clusters to populate fields.
- ☒ **Standard cluster**  
Continuous integration, web serving, backends. Best choice for further customization or if you are not sure what to choose.
- ☐ Your first cluster  
Experimenting with Kubernetes Engine, deploying your first application. Affordable choice to get started.
- ☐ CPU intensive applications  
Web crawling or anything else that requires more CPU.
- ☐ Memory intensive applications  
Databases, analytics, things like Hadoop, Spark, ETL, or anything else that requires more memory.
- ☐ GPU Accelerated Computing  
Machine learning, video transcoding, scientific computations or anything else that is compute-intensive and can utilize GPUs.
- ☐ Highly available  
Most demanding availability requirements. Both the master and the nodes are replicated across multiple zones.

**'Standard cluster' template**

Continuous integration, web serving, backends. Best choice for further customization or if you are not sure what to choose.

You will be billed for the 3 nodes (VM instances) in your cluster. [Learn more](#)

ⓘ Some fields can't be changed after the cluster is created. Hover over the help icons to learn more. [Dismiss](#)

Name ⓘ  
standard-cluster-1

Location type ⓘ  
☒ Zonal  
☐ Regional

Zone ⓘ  
us-central1-a

Master version  
1.9.7-gke.0 (default)

**Node pools**

Node pools are separate instance groups running Kubernetes in a cluster. You may add node pools in different zones for higher availability, or add node pools of different type machines. To add a node pool, click [Add](#). [Learn more](#)

**default-pool**

Number of nodes  
3

Machine type ⓘ  
Customize to select cores, memory and GPUs  
1 vCPU 3.75 GB memory [Customize](#)

[Upgrade your account to create instances with up to 96 cores](#)

Auto-upgrade: On  
[Advanced edit](#)

[+ Add node pool](#)

ⓘ Advanced options

[Create](#) [Cancel](#) [Equivalent REST or command line](#)

Crie um cluster

Você verá muitas configurações opcionais para criar um cluster na página Criar cluster

O modelo de cluster contém clusters pré-configurados que serão aplicados para criar um novo cluster assim que você selecionar um.

Para nosso caso, vamos selecionar a primeira opção de cluster.

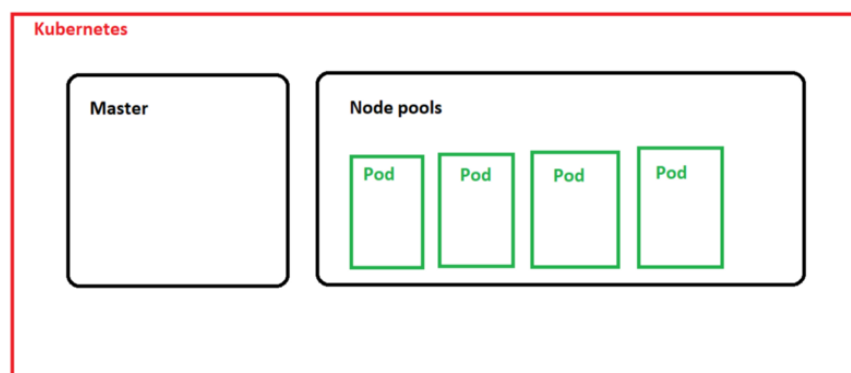
Em Nome, você pode selecionar qualquer nome

No local, você pode selecionar o cluster para ser regional ou zonal

Em Zone, selecione o local próximo aos seus usuários ou aplicativo.

**Na versão Master, selecione a versão do Kubernetes Master.**

Nos pools de nós, selecione o número de máquinas e tipos de máquina necessários para criar um cluster,



Kubernetes

Existem várias outras opções em recursos avançados

**Load balancing**

☒ Enable HTTP load balancing ?

**Network security**

☐ Private cluster ?


☐ Enable master authorized networks ?

☐ Enable network policy ?

**Security**

☒ Enable basic authentication ?

☒ Issue a client certificate ?

 Starting with version 1.12, clusters will have basic authentication and client certificate issuance disabled by default. [Disable them now](#)

☐ Enable legacy authorization ?

☐ Enable binary authorization (beta) ?

**Metadata**

Description (Optional) ?

**Labels** (Optional)

To organize your project, add arbitrary labels as key/value pairs to your resources. Use labels to indicate different environments, services, teams, and so on. [Learn more](#)

+ Add label

**Additional features**

☒ Enable Stackdriver Logging service ?

☒ Enable Stackdriver Monitoring service ?

☐ Try the new Stackdriver beta Monitoring and Logging experience

The beta experience increases observability by aggregating incidents, system metrics, and logs into one single view

☐ Enable Cloud TPU (beta) ?

☐ Enable Kubernetes alpha features in this cluster ?

☐ Enable Kubernetes Dashboard ?

Create

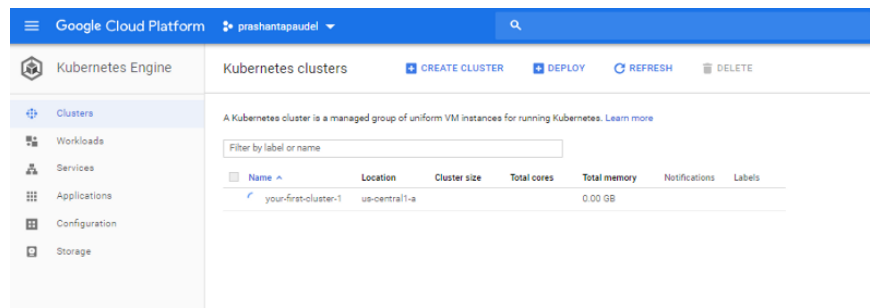
Reset

Equivalent [REST](#) or [command line](#)

características avançadas

Clique em Criar e espere um pouco até que o cluster seja construído.



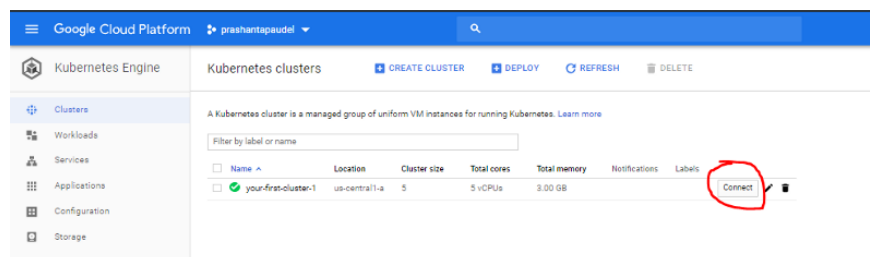


em processo

o comando gcloud para a mesma operação é

gcloud beta container - os clusters “prashantapaudel-219410” do projeto criam “your-first-cluster-1” - zona “us-central1-a” - nome de usuário “admin” - versão de cluster “1.9.7-gke.6” - máquina tipo “f1-micro” - tipo de imagem “COS” - tipo de disco “pd-standard” - tamanho de disco “30” - escopos “https://www.googleapis.com/auth/compute”, “https://www.googleapis.com/auth/devstorage.read\_only”, “https://www.googleapis.com/auth/logging.write”, “https://www.googleapis.com/auth/monitoring”, “https://www.googleapis.com/auth/servicecontrol”, “https://www.googleapis.com/auth/service.management.readonly”, “https://www.googleapis.com/auth/trace .acrescentar” - num-nodes “5” - enable-cloud-logging - enable-cloud-monitoring - rede “projects / prashantapaudel-219410 / global / networks / default” - sub-rede “projects / prashantapaudel-219410 / regions / us-central1 / subnetworks / default” - addons HorizontalPodAutoscaling, HttpLoadBalancing, KubernetesDashboard - habilitar-autoupgrade - enable-autorepair

Depois que o cluster estiver pronto, você verá o botão de conexão além do cluster no Painel



Página de cluster

Clicando no nome do cluster, você pode verificar a configuração do cluster

✓ your-first-cluster-1

[Details](#) [Storage](#) [Nodes](#)

---

Cluster

Master version	1.9.7-gke.6	<a href="#">Upgrade available</a>
Endpoint	35.238.42.112	<a href="#">Show credentials</a>
Client certificate	Enabled	
Binary authorization	Disabled	
Kubernetes alpha features	Disabled	
Total size	5	
Master zone	us-central1-a	
Node zones	us-central1-a	
Network	<a href="#">default</a>	
Subnet	<a href="#">default</a>	
VPC-native (alias IP)	Disabled	
Pod address range	10.4.0.0/14	
Stackdriver Logging	Enabled	
Stackdriver Monitoring	Enabled	
Private cluster	Disabled	
Master authorized networks	Disabled	
Network policy	Disabled	
Legacy authorization	Disabled	
Maintenance window	Any time	
Cloud TPU	Disabled	

Labels

None

✕ [Add-ons](#)

✕ [Permissions](#)

your-first-cluster-1

Details Storage Nodes

### Persistent volumes

Filter persistent volumes

Name	Status	Type	Source	Read only	Storage Class	Claim
No matching results						

### Storage classes

Filter storage classes

Name	Provisioner	Type	Zone
standard	kubernetes.io/gce-pd	pd-standard	

Google Cloud Platform

Kubernetes Engine

Clusters

your-first-cluster-1

Details Storage Nodes

Filter nodes

Name	Status	CPU requested	CPU allocatable	Memory requested	Memory allocatable	Storage requested	Storage allocatable
gke-your-first-cluster-1-pool-1-d4e1b41d-m0	Ready	348 mCPU	940 mCPU	940.2 MB	622.85 MB	0 B	0 B
gke-your-first-cluster-1-pool-1-d4e1b41d-p0	Ready	330 mCPU	940 mCPU	440.4 MB	622.85 MB	0 B	0 B
gke-your-first-cluster-1-pool-1-d4e1b41d-b0	Ready	460 mCPU	940 mCPU	325.08 MB	622.85 MB	0 B	0 B
gke-your-first-cluster-1-pool-1-d4e1b41d-f0	Ready	235 mCPU	940 mCPU	371.2 MB	622.85 MB	0 B	0 B
gke-your-first-cluster-1-pool-1-d4e1b41d-n0	Ready	200 mCPU	940 mCPU	209.72 MB	622.85 MB	0 B	0 B

Nós

Desta forma, implantamos o cluster do mecanismo Kubernetes com 5 modos.

*Em gcloud*

Para fazer o mesmo no gcloud, use o comando, lembre-se de que há muitos escopos que precisam ser adicionados para torná-lo semelhante ao acima. **Este comando apenas inicia o cluster com configurações padrão.**

Os clusters de contêiner \$ gcloud criam anothercluster --region = us-central1-a

AVISO: A partir de 1.12, os novos clusters terão a autenticação básica desativada por padrão. A autenticação básica pode ser ativada (ou desativada) manualmente usando o sinalizador `-[no-] enable-basic-auth`.

AVISO: A partir de 1.12, novos clusters não terão um certificado de cliente emitido. Você pode ativar / desativar

```

certificado de cliente emitido. voce pode ativar (ou
desativar) manualmente a emissão do certificado de cliente
usando o
sinalizador `- [no-] issue-client-certificate` .
AVISO: Atualmente, o VPC-native não é o modo padrão durante
a criação do cluster. No futuro, isso se tornará o modo
padrão e poderá ser desativado usando o sinalizador `--no-
enable-ip-alias`. Use o sinalizador `- [no-] enable-ip-
alias` para suprimir este aviso.
AVISO: a partir de 1.12, os pools de nós padrão em novos
clusters terão seus terminais de metadados da instância
legada do Compute Engine desativados por padrão. Para criar
um cluster com pontos de extremidade de metadados da
instância legada desabilitados no conjunto de nós padrão,
execute `clusters create` com o sinalizador - metadados
disable-legacy-endpoints = true`.
Isso ativará o recurso de acompanhamento automático para
nós. Consulte https://cloud.google.com/kubernetes-
engine/docs/node-auto-repair para obter mais informações
sobre os autorepairs do nó.
ATENÇÃO: A partir do Kubernetes v1.10, os novos clusters não
mais terão escopos compute-rw e storage-ro adicionados ao
especificado em --scopes (embora o último permaneça incluído
nos escopos padrão). Para usar esses escopos, adicione-os
explicitamente a --scopes. Para usar o novo comportamento,
defina a propriedade container / new_scopes_behavior (gcloud
config set container / new_scopes_behavior true).
Criando cluster outroagregador em us-central1-a ... feito.
Criado
[https://container.googleapis.com/v1/projects/prashantapaude
l-219410/zones/us-central1-a/clusters/anothercluster].
Para inspecionar o conteúdo de seu cluster, acesse:
https://console.cloud.google.com/kubernetes/workload_/gcloud
/us-central1-a/anothercluster?project=prashantapaudel-219410
entrada kubeconfig gerada para outroagosto.
NOME LOCALIZAÇÃO MASTER_VERSION MASTER_IP MACHINE_TYPE
NODE_VERSION NUM_NODES STATUS outra opção
us-central1-a 1.9.7-gke.6 35.238.5.216 n1-standard-1 1.9.7-
gke.6 3 RUNNING
prashantagcppaudel @ cloudshell: ~ (prashantapaudel-219410)
$

```

## Conecte-se ao cluster

Para se conectar ao cluster, use o comando

```

$ gcloud clusters de contêineres get-credentials
anothercluster --zone us-central1-a --projecto
prashantapaudel-219410 Recuperando
dados de ponto de extremidade e autenticação do cluster.
Entrada kubeconfig gerada para outroagente.

```

Listar os clusters

```
prashantagcppaudel @ cloudshell: ~ (prashantapaudel-219410)
$ gcloud lista de clusters de contêiner
NOME LOCALIZAÇÃO MASTER_VERSION MASTER_IP MACHINE_TYPE
NODE_VERSION NUM_NODES STATUS outra opção
us-central1-a 1.9.7-gke.6 35.238.5.216 n1-standard-1 1.9.7-
gke. 6 3 CORRENDO
seu primeiro aglomerado-1 us-central1-a 1.9.7-gke.6
35.238.42.112 f1-micro 1.9.7-gke.6 5 EXECUTANDO
prashantagcppaudel @ cloudshell: ~ (prashantapaudel-219410)
$
```

## Implantando um aplicativo contêiner no Kubernetes Engine usando pods

### Etapa 1: criar a imagem do contêiner

Primeiro de tudo, precisamos ter um aplicativo e um arquivo docker prontos.

Vamos usar o aplicativo hello já disponível do Github.

Primeiro, baixe o aplicativo

```
$ git clone
https://github.com/GoogleCloudPlatform/kubernetes-engine-
samples
Clonando em 'kubernetes-engine-samples' ...
remote: Enumerando objetos: 29, pronto.
remote: Contando objetos: 100% (29/29), pronto.
remote: Compactando objetos: 100% (25/25), pronto.
remote: Total 476 (delta 8), reutilizado 8 (delta 4),
reutilizado na embalagem 447
Objetos recebidos: 100% (476/476), 389.76 KiB | 0 bytes /
s, pronto.
Resolvendo deltas: 100% (201/201), pronto.
```

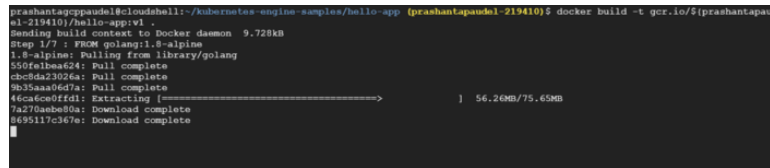
Defina a `PROJECT_ID` variável de ambiente em seu shell recuperando o ID do projeto pré-configurado `gcloud` executando o comando abaixo:

```
$ export PROJECT_ID = "$(gcloud config projeto-valor -q)"
Sua configuração ativa é: [cloudshell-7515]
```

O valor de `PROJECT_ID` será usado para marcar a imagem do contêiner para empurrá-la para seu Container Registry privado.

**Para criar a imagem do contêiner** deste aplicativo e marcá-lo para upload, execute o seguinte comando:

```
docker build -t gcr.io/${PROJECT_ID}/hello-app:v1
```



```
prashantapaudel@cloudshell:~/kubernetes-engine-samples/hello-app (prashantapaudel-219410)$ docker build -t gcr.io/${prashantapaudel-219410}/hello-app:v1 .
Sending build context to Docker daemon  9.728kB
Step 1/7 : FROM golang:1.8-alpine
1.8-alpine: Pulling from library/golang
550fe1bea624: Pull complete
c4c8da23d26a: Pull complete
8b25aa9667a: Pull complete
46ca6ce0ffdl: Extracting [=====] 56.26MB/75.65MB
7a270aeb80da: Download complete
6490117c367e: Download complete
```

docker de construção

Este comando instrui o Docker a construir a imagem usando o `Dockerfile` diretório atual e marcá-la com um nome, como `gcr.io/my-project/hello-app:v1`. O `gcr.io` prefixo refere-se ao Google Container Registry, no qual a imagem será hospedada.

A execução deste comando não faz upload da imagem ainda.

Execute o `docker images` comando para verificar se a compilação foi bem-sucedida:

```
$ docker image ls
REPOSITÓRIO ETIQUETA ID DA IMAGEM TAMANHO CRIADO
gcr.io/219410/hello-app v1 367da9716bc1 4 minutos atrás
10.3MB
<nenhum> <nenhum> be6ee6935fed 4 minutos atrás 263MB
```

alpine mais recente 196d12cf6ab1 6 semanas atrás 4.41MB  
golang 1.8-alpine 4cb86d3661bf Há 8 meses 257MB

## Etapa 2: carregar a imagem do contêiner

Primeiro, execute o comando auth

```
gcloud auth configure-docker

prashantagcppaudel @ cloudshell: ~ / kubernetes-engine-
samples / ola-app (prashantapaudel-219410) $ gcloud auth
configure-docker
AVISO: Seu arquivo de configuração em
[/home/prashantagcppaudel/.docker/config.json] contém essas
entradas auxiliares de credenciais :

{
  "credHelpers": {
    "gcr.io": "gcr",
    "us.gcr.io": "gcr",
    "asia.gcr.io": "gcr",
    "staging-k8s.gcr.io" : "gcr",
    "eu.gcr.io": "gcr"
  }
}
Estes serão sobrescritos.
As seguintes configurações serão adicionadas ao arquivo de
configuração do Docker
localizado em
[/home/prashantagcppaudel/.docker/config.json]:
{
  "credHelpers":

    "eu.gcr.io": "gcloud",
    "asia.gcr.io": "gcloud",
    "staging-k8s.gcr.io": "gcloud",
    "marketplace.gcr.io": "gcloud"
  }
}

Você quer continuar (S / n)? y

Arquivo de configuração do Docker atualizado.
prashantagcppaudel @ cloudshell: ~ / kubernetes-engine-
samples / olá-app (prashantapaudel-219410) $
```

Agora você pode usar a ferramenta de linha de comando do Docker para carregar a imagem no seu Container Registry:

```
janela de encaixe push gcr.io/${PROJECT_ID}/hello-app:v1
```

## Etapa 3: execute seu contêiner localmente (opcional)

Para testar sua imagem de contêiner usando seu mecanismo Docker local, execute o seguinte comando:

```
execução do docker --rm -p 8080: 8080  
gcr.io/${PROJECT_ID}/hello-app:v1
```

Se você está no Cloud Shell, pode clicar no botão "Visualizar na Web" no canto superior direito para ver seu aplicativo sendo executado em uma guia do navegador. Caso contrário, abra uma nova janela de terminal (ou uma guia Cloud Shell) e execute para verificar se o contêiner funciona e responde a solicitações com "Hello, World!":

## Etapa 4: criar um cluster de contêiner

Agora que a imagem do contêiner é armazenada em um registro, você precisa criar um cluster de contêiner para executar a imagem do contêiner. Um cluster consiste em um conjunto de instâncias de VM do Compute Engine executando o Kubernetes, o sistema de orquestração de cluster de código aberto que aciona o GKE.

Depois de criar um cluster GKE, você usa o Kubernetes para implantar aplicativos no cluster e gerenciar o ciclo de vida dos aplicativos.

Execute o seguinte comando para criar um cluster de três nós chamado

```
hello-cluster :
```

```
Os clusters de contêiner gcloud criam hello-cluster --num-  
nodes = 3
```



Pode levar vários minutos para o cluster ser criado. Quando o comando estiver concluído, execute o seguinte comando e veja as três instâncias de VM de trabalhador do cluster:

```
gcloud compute instances list
```

que mostrará todas as instâncias de computação, incluindo contêineres

```
prashantagcppaudel @ cloudshell: ~ / kubernetes-engine-
samples / olá-app (prashantapaudel-219410) $ gcloud lista de
instâncias de computação
NAME      ZONE      MACHINE_TYPE
PREEMPTIBLE INTERNAL_IP EXTERNAL_IP STATUS gke-
anothercluster-padrão-pool-b5fd2208-9xxf us-central1-a n1-
standard-1 10.128.0.7 104.197.136.126 EXECUTANDO
gke-anothercluster-default-pool-b5fd2208-jx3k us-central1-a
n1-standard-1 10.128.0.9 35.225.236.118 EXECUTANDO o
gke-anothercluster-default-pool-b5fd2208-kbsm us -central1-
a-n1-standard-1 10.128.0.8 35.192.54.14 FUNCIONAMENTO
gke-seu-primeiro-cluster-1-pool-1-da4e9b41-0m4c us-central1-
a1-micro 10.128.0.3 104.154.106.120 RUNNING
gke-seu-primeiro-cluster-1-pool-1-da4e9b41-5pks us-central1-
a-f1-micro 10.128.0.6 35.193.105.242 RUNNING
gke-seu-primeiro-cluster-1-pool-1-da4e9b41-dk8l us -
central1-a1-micro 10.128.0.2 35.239.24.9 FUNCIONAMENTO
gke-your-first-cluster-1-pool-1-da4e9b41-hfzl us-central1-
a1-micro 10.128.0.4 35.238.147.20 FUNCIONAMENTO
gke-your- primeiro-cluster-1-pool-1-da4e9b41-kn1g us-
central1-a1-micro 10.128.0.5 35.184.0.99 RUNNING
prashantagcppaudel @ cloudshell: ~ / kubernetes-engine-
samples / hello-app (prashantapaudel-219410) $
```

## Etapa 5: implemente seu aplicativo

Para implantar e gerenciar aplicativos em um cluster GKE, você deve se comunicar com o sistema de gerenciamento de clusters Kubernetes. Você normalmente faz isso usando a `kubectl` ferramenta de linha de comando.

Kubernetes representa aplicativos como Pods, que são unidades que representam um contêiner (ou grupo de contêineres fortemente acoplados). O Pod é a menor unidade implantável no Kubernetes. Neste tutorial, cada Pod contém apenas seu `hello-app` contêiner.

O `kubectl run` comando abaixo faz com que o Kubernetes crie uma Implantação nomeada `hello-web` em seu cluster. A Implantação gerencia várias cópias do seu aplicativo, chamadas réplicas, e as programa para serem executadas nos nós individuais do cluster. Nesse caso, a implantação estará executando apenas um pod de seu aplicativo.

Execute o seguinte comando para implantar seu aplicativo, ouvindo na porta 8080:

```
kubectl executar hello-web --image = gcr.io / $ {PROJECT_ID}
/ ola-app: v1 --port 8080
```

Para ver o Pod criado pela Implantação, execute o seguinte comando:

```
NOME PRONTO ESTADO REINICIA A IDADE
hello-web-4017757401-px7tx 1/1 Running 0 3s
```

## Etapa 6: Exponha seu aplicativo à Internet

Por padrão, os contêineres que você executa no GKE não podem ser acessados pela Internet, porque eles não têm endereços IP externos. Você deve expor explicitamente seu aplicativo ao tráfego da Internet, execute o seguinte comando:

```
kubectl expõe a implementação hello-web --type =
LoadBalancer --port 80 --target-port 8080
```

## Etapa 7: amplie seu aplicativo

Você adiciona mais réplicas ao recurso de implantação do seu aplicativo usando o `kubectl scale` comando. Para adicionar duas réplicas adicionais à sua Implantação (para um total de três), execute o seguinte comando:

```
implementação em escala kubectl hello-web --replicas = 3
```

Você pode ver as novas réplicas em execução no seu cluster executando os seguintes comandos:

```
kubectl obter implantação hello-web
```

Saída:

```
NOME DESEJADO ATUAL ACIMA DISPONÍVEL AGE  
hello-web 3 3 3 2 1m
```

```
kubectl obter pods
```

Saída:

```
NOME READY STATUS REESTARES Idade  
hello-web-4017757401-ntgdb 1/1 Corrida 0 9s  
hello-web-4017757401-pc4j9 1/1 Corrida 0 9s  
hello-web-4017757401-px7tx 1/1 Corrida 0 1m
```

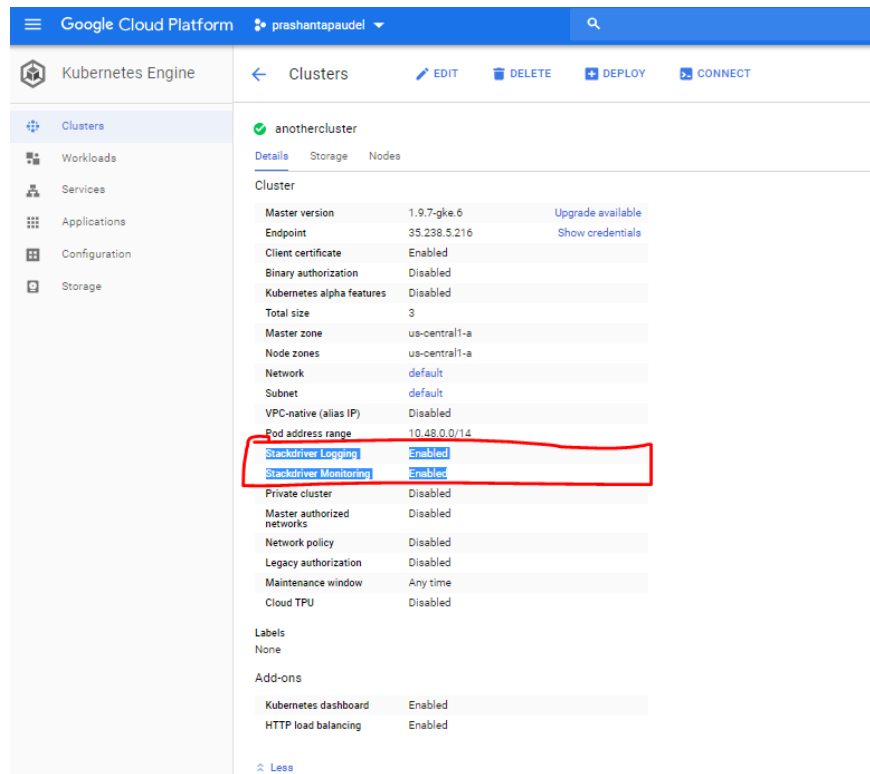
Agora, você tem várias instâncias do seu aplicativo sendo executadas independentemente umas das outras e você pode usar o `kubectl scale` comando para ajustar a capacidade do seu aplicativo.

O balanceador de carga que você provisionou na etapa anterior começará a rotear o tráfego para essas novas réplicas automaticamente.

-----

## Configurando o monitoramento e o registro de aplicativos do Kubernetes Engine

Agora é muito fácil iniciar o monitoramento de cluster apenas selecionando o monitoramento enquanto cria o cluster



monitoramento do stackdriver

## Instalando o Kubernetes Monitoring

### Beta

Esta é uma versão Beta do **Stackdriver Kubernetes Monitoring**. Esse recurso não é coberto por nenhum SLA ou política de descontinuação e pode estar sujeito a alterações incompatíveis com versões anteriores.

A página explica como instalar a versão Beta do Stackdriver Kubernetes Monitoring. Esta é uma versão opcional que substitui o suporte ao Stackdriver herdado descrito nos guias, monitoramento e registro do GKE. Esta versão beta destina-se apenas ao Google Kubernetes Engine (GKE).

Você pode instalar essa nova versão quando criar um novo cluster GKE ou atualizar um cluster GKE existente. Se você não optar por essa versão, seus novos clusters usarão o suporte atual do Stackdriver, também chamado de **suporte herdado do GA**.

## Antes de você começar

**Cuidado:** Se você pretende atualizar um cluster existente, leia as informações sobre incompatibilidades.

- Você deve ser um **proprietário** do projeto que contém seu cluster. O projeto deve ser monitorado por um espaço de trabalho.
- Você será solicitado a usar o Kubernetes versão 1.10.6 ou 1.11.2 (ou posterior, se disponível).

## Criando um novo cluster

Você deve ativar essa versão Beta ao criar um novo cluster usando o console do GKE ou a `gcloud` ferramenta de linha de comando do Cloud SDK.

## CONSOLE

1. Vá para a página do GKE **Kubernetes Clusters** para o seu projeto. O seguinte botão leva você até lá:
  2. Selecione **Criar Cluster** na parte superior da página.
  3. Na caixa de diálogo, especifique os seguintes campos além de qualquer outra propriedade desejada em seu cluster. Para mais informações, consulte Criação de um cluster.
- **Versão do Cluster** : 1.10.6 ou posterior; ou 1.11.2 ou posterior
  - **Experimente a nova experiência Beta de Monitoramento e Registro** : CHECKED
  - Se você não vir essa opção, não selecionou uma versão de cluster do 1.10.6 ou 1.11.2 (ou posterior, se disponível). Se você vir a opção, mas estiver esmaecida, verifique se as **opções Stackdriver Logging e Stackdriver Monitoring** são `Enabled` (o padrão).

- Esta parte do painel de opções deve se parecer com o seguinte:

**Legacy Authorization** ?

Disabled

**Stackdriver Logging** ?

Enabled

**Stackdriver Monitoring** ?

Enabled

☒ Try the new Beta Monitoring and Logging experience

The beta experience increases observability by aggregating incidents, system metrics, and logs into one single view

1. Clique em **Criar** para criar o cluster. Isto pode tomar algum tempo.

## Atualizando um Cluster Existente

A tabela a seguir mostra como atualizar seu cluster para a versão beta gerenciada do Stackdriver Kubernetes Monitoring. Existem diferenças dependendo do que o Stackdriver anterior suporta o cluster em execução, o que você pode dizer clicando no nome do seu cluster no console do GKE.

O Stackdriver atual suporta o status do Stackdriver listado no console como atualizarBeta gerenciado releaseEnabled v2 (beta)

Você só precisa atualizar sua versão do Kubernetes. O Stackdriver é atualizado quando o Kubernetes é GA legacyEnabledUse as seguintes instruções de atualização.noneDisabledUse as seguintes instruções de atualização .non-managed,

manual Beta1Disabled e o cluster contém um `stackdriver-agents` namespaceExclua os agentes atuais:

```
kubectl delete ns stackdriver-agents
```

use as seguintes instruções de atualização.

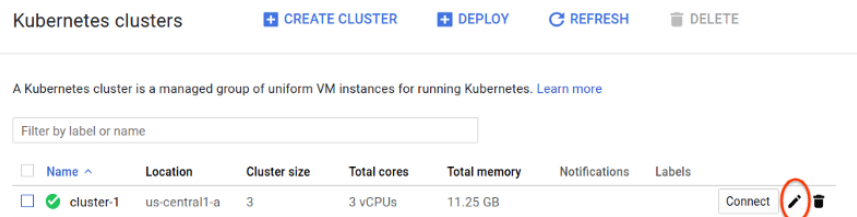
Notas:

1 Atualmente, isso só pode acontecer se você instalou manualmente os agentes atualizados. A instalação manual não é mais necessária ou recomendada.

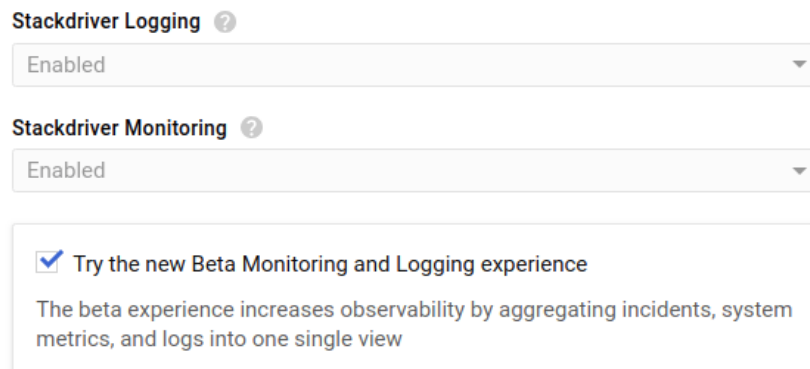
As instruções a seguir atualizam seu cluster GKE para a versão beta gerenciada do Stackdriver Kubernetes Monitoring. Verifique a [tabela anterior](#) que menciona alguns casos de canto.

## CONSOLE

1. Vá para a página do GKE **Kubernetes Clusters** para o seu projeto. O seguinte botão leva você até lá:
2. Atualize seu cluster para o Kubernetes versão 1.10.6 ou 1.11.2 (ou posterior, se disponível). Para obter instruções, consulte [Atualizando clusters](#). Permitir que a atualização seja concluída.
3. Clique no ícone **Editar** (editar) para o seu cluster:



1. Role para baixo até a seção Stackdriver e marque a caixa **Experimente a nova experiência Beta de monitoramento e registro**. Talvez seja necessário **ativar as opções de Stackdriver Monitoring and Logging** também:



1. Se você não tiver atualizado para a versão exigida do Kubernetes, não verá a opção **Experimentar o novo Beta...**
2. Clique em **Salvar** na parte inferior da página.

**Próxima etapa** : acesse [Verificando sua instalação](#) nesta página.

## Verificando sua instalação

Para verificar se o Stackdriver Monitoring está sendo executado corretamente, aguarde alguns minutos e examine seu cluster. Use o console ou `gcloud` veja se os pods do Stackdriver estão em execução:

## CONSOLE

1. No console do GCP, acesse **Stackdriver > Monitoramento** :
2. Em Monitoramento, vá para **Recursos > Kubernetes BETA** para o Espaço de Trabalho contendo seu projeto GCP.
3. Você pode ver todos os clusters em seu espaço de trabalho que estão usando o Stackdriver Kubernetes Monitoring.
4. Selecione a guia **Carga de trabalho** .
5. Expanda seu cluster e o `kube-system` namespace. Você deve ver pods em execução no `kube-system` namespace com os seguintes nomes:
  - `fluentd-gcp-...` : o agente do Stackdriver Logging.
  - `heapster-...` : o agente de monitoramento.
  - `metadata-...` : o agente de metadados do Stackdriver.



























