Official Google Cloud Certified Associate Cloud Engineer Study Guide
Kubernetes -Jose R F Junior - web2ajax@gmail.com

Capítulo 7: Computação com Kubernetes Este capítulo apresenta o Kubernetes Engine,
o serviço Kubernetes gerenciado do Google. O Kubernetes é uma plataforma de orquestração de contêineres
criada e lançada como código aberto pelo Google. Neste capítulo, você aprenderá os conceitos básicos
de contêineres, orquestração de contêineres e a arquitetura do Kubernetes. A discussão incluirá uma visão geral dos objetos do Kubernetes, como pods, serviços, volumes e namespaces, bem como os controladores do Kubernetes, como ReplicaSets, implantações e jobs.
Em seguida, o capítulo recorre à implantação de um cluster do Kubernetes usando o console do GCP, o Cloud Shell
e o SDK. Você também verá como implantar pods, o que inclui o download de uma
imagem existente do Docker, a criação de uma imagem do Docker, a criação de um pod e a implantação de um aplicativo
no cluster do Kubernetes. Claro, você precisará saber como monitorar um cluster de
servidores. Este capítulo fornece uma descrição de como configurar o monitoramento e o registro em log com o
Stackdriver, que é o serviço de monitoramento de aplicativos, serviços, contêiner e infraestrutura do Google.
Capítulo 8: Gerenciando clusters do Kubernetes Neste capítulo, você aprenderá os fundamentos do
gerenciamento de um cluster do Kubernetes, incluindo a visualização do status do cluster, a visualização do
conteúdo do repositório de imagens, exibindo detalhes sobre imagens no repositório e
adicionando, modificando e removendo nós, pods e serviços. Como no capítulo sobre
gerenciamento de VMs, neste capítulo, você aprenderá a realizar operações de gerenciamento com as três ferramentas de gerenciamento: console do GCP, Cloud Shell e SDK. O capítulo termina com uma discussão de diretrizes e boas práticas para gerenciar um
cluster do Kubernetes.

Capítulo 7.
84 Capítulo 4 ■ Introdução à computação no Google Cloud
Kubernetes Engine Use Cases O
Kubernetes Engine é uma boa opção para aplicações de grande escala que exigem alta disponibilidade e alta confiabilidade. O Kubernetes Engine suporta o conceito de pods e

conjuntos de implantação , que permitem que desenvolvedores e administradores de aplicativos gerenciem serviços como uma unidade lógica. Isso pode ajudar se você tiver um conjunto de serviços que suporte uma interface de usuário, outro
conjunto que implemente lógica de negócios e um terceiro conjunto que forneça serviços de back-end. Cada um
desses diferentes grupos de serviços pode ter diferentes ciclos de vida e requisitos de escalabilidade.
O Kubernetes ajuda a gerenciá-los em níveis de abstração que fazem sentido para usuários, desenvolvedores e profissionais de DevOps.
Funções de nuvem
O Cloud Functions é uma plataforma de computação sem servidor projetada para executar partes
de código de finalidade única em resposta a eventos no ambiente do GCP. Não há necessidade de provisionar ou
gerenciar VMs, contêineres ou clusters ao usar o Cloud Functions. Código escrito em
Node.js 6, Node.js 8 ou Python 3.7 pode ser executado em Cloud Functions.
O Cloud Functions não é uma plataforma de computação de uso geral, como o Compute Engine
ou o App Engine. O Cloud Functions fornece a "cola" entre serviços que são de outra forma
independentes.
Por exemplo, um serviço pode criar um arquivo e enviá-lo para o Cloud Storage, e outro serviço precisa coletar esses arquivos e executar algum processamento no arquivo. Ambos os serviços
pode ser desenvolvido de forma independente. Não há necessidade de saber sobre o outro. No entanto,
você precisará de alguma maneira de detectar que um novo arquivo foi gravado no Cloud Storage e, em seguida,
o outro aplicativo poderá começar a processá-lo. Não queremos escrever aplicativos de maneiras que
façam suposições sobre outros processos que possam fornecer entrada ou consumir saída. Os serviços
podem mudar independentemente uns dos outros. Não devemos ter que acompanhar as dependências
entre serviços, se pudermos evitá-lo. O Cloud Functions nos ajuda a evitar essa situação.
Ambiente de execução de funções de nuvem
O GCP gerencia tudo o que é necessário para executar seu código em um ambiente seguro e isolado. É claro que, abaixo da abstração sem servidor, existem servidores virtuais e físicos
executando seu código, mas você, como engenheiro de nuvem, não precisa administrar nenhuma dessas infraestruturas. Três pontos principais a serem lembrados sobre as Funções na nuvem são as seguintes:
■ As funções são executadas em um ambiente de execução seguro e isolado.

■ Compute a escala de recursos conforme necessário para executar quantas instâncias do Cloud Functions, conforme
necessário, sem precisar fazer nada para controlar o dimensionamento.
■ A execução de uma função é independente de todas as outras. Os ciclos de vida das
funções de nuvem não dependem uns dos outros.

Na primeira vez que você usa o Kubernetes Engine, pode ser necessário criar credenciais. Você pode
fazer isso clicando no botão Criar Credenciais, na parte superior da página Visão Geral. Um formulário
como mostrado na Figura 7.2. Você pode especificar qual API está usando
e depois gerar suas credenciais.
Figura 7.2 A forma de criar credenciais necessárias para usar o Kubernetes Engine
Depois de criar credenciais, se necessário, você pode criar um cluster. A Figura 7.3 mostra a primeira
página na etapa de criação do cluster.
F IGURA 7.3 A primeira forma para a criação de um cluster Kubernetes em nuvem Console

Quando você clique em Criar Cluster, você será presenteado com a opção de escolher entre
vários modelos, como mostrado na Figura 7.4. Os modelos variam no número de vCPUs,
memória e uso de GPUs. Por exemplo, o modelo de cluster padrão usa três nós
com uma vCPU e 3,75 GB de memória, enquanto o modelo CPU Intensive usa quatro vCPUs
e 3,6 GB de memória.
Figura 7.4 Modelos para criar um cluster do Kubernetes
Você pode modificar os parâmetros fornecidos no modelo. Por exemplo, se você deseja
executar VMs em diferentes zonas para melhorar a disponibilidade, é possível especificar vários conjuntos de nós.
Os pools de nós são grupos de instâncias em um cluster do Kubernetes. Eles são muito parecidos com um
grupo de instâncias gerenciadas , mas não são iguais.

Pode levar alguns minutos para criar um cluster. Quando o cluster é criado, ele aparecerá na
lista de clusters, como na Figura 7.5.
Figura 7.5 A listagem de cluster mostra o número de instâncias, núcleos totais e total
memória.

A partir da listagem de clusters, você pode editar, excluir e conectar-se a um cluster. Ao
clicar em Conectar, você recebe um comando gcloud para se conectar ao cluster a partir da
linha de comando . Você também tem a opção de visualizar a página Cargas de Trabalho, conforme mostrado na Figura 7.6.
Figura 7.6 Você pode se conectar ao cluster usando um comando gcloud na linha de
comando ou visualizando a página Cargas de Trabalho.
O Kubernetes executa várias cargas de trabalho para gerenciar o cluster. Você pode visualizar as
cargas de trabalho em execução no momento na página Cargas de trabalho da seção Kubernetes Engine
do Cloud Console. A Figura 7.7 mostra um subconjunto das cargas de trabalho em execução em um
cluster recém- iniciado.
Implantando clusters do Kubernetes 153
Figura 7.7 A página Cargas de Trabalho lista as cargas de trabalho atualmente em execução.
Implantando clusters do Kubernetes usando o Cloud Shell e o
Cloud SDK
Assim como outros serviços do GCP, o Kubernetes Engine pode ser gerenciado usando a linha de comando. O
comando básico para trabalhar com o Kubernetes Engine é o seguinte comando
gcloud : gcloud beta container
Observe que os comandos do Kubernetes Engine incluem a palavra beta. O Google indica um
serviço que ainda não está em disponibilidade geral ao incluir a palavra alfa ou beta no
comando gcloud. No momento em que você ler isso, o Kubernetes Engine poderá estar disponível
em geral e, nesse caso, o termo beta não será mais usado.
Este comando do gcloud tem muitos parâmetros, incluindo os seguintes:
■ Projeto
■ Zona
■ Tipo de máquina
tipo ■ Imagem
Tipo ■ Disk
■ Disk tamanho
■ Número de nós
154 Capítulo 7 ■ Computing com Kubernetes
Um comando básico para a criação de um cluster se parece com isso:
clusters de recipiente gcloud criar CH07-fragmentação --num-nodes = 3 --region = us-central1 Os
comandos para criar clusters podem se tornar bastante longos. Por exemplo, aqui está o comando para criar um cluster usando o modelo padrão:

gcloud beta container --projecto "ferrous-depth-220417" clusters create "standard-cluster-2" --zone "us-central1-a" - nome de usuário "admin" --cluster-version "1.9.7-gke.6" --machine-type "n1-standard-1" --image-type "COS" --disk-type "pd-standard" --disk-size "100" --scopes HttpLoadBalancing, KubernetesDashboard --able-autoupgrade --enable-autorepair

Em vez de escrever este tipo de comando de scratch, você pode usar o Cloud Console para selecionar um modelo e, em seguida, usar a opção para gerar a linha de comando equivalente a partir do formulário Create Cluster.

Implantando Pods de Aplicativo

Agora que você criou um cluster, vamos implantar um aplicativo.

Na página Cluster do Kubernetes Engine on Cloud Console, selecione Criar Implantação. Uma forma como a da Figura 7.8 aparece. Nesse formulário, você pode especificar o seguinte:

- **Imagem do contêiner**
- **Variáveis de ambiente**
- **Comando inicial**
- **Nome do aplicativo**
- **Rótulos**
- **Namespace**
- **Cluster a ser implantado no**

Implantando pods de aplicativos 167

F igure 7.8 A opção Criar Implantação fornece um formulário para especificar um contêiner a ser executado e um comando inicial para iniciar o aplicativo em execução.

156 Capítulo 7 ■ Computando com o Kubernetes

Depois de especificar uma implantação, você pode exibir a especificação YAML correspondente , que pode ser salva e usada para criar implantações a partir da linha de comando.

A Figura 7.9 mostra um exemplo de arquivo YAML de implementação. A saída é sempre exibida no formato YAML.

F igure 7.9 Especificação YAML para uma implantação do Kubernetes

Além de instalar o Cloud SDK, você precisará instalar o Kubernetes ferramenta de linha de comando kubectl para trabalhar com clusters a partir da linha de comando. Você pode fazer isso com o seguinte comando:

gcloud components install kubectl

Monitorando o Kubernetes 157

Se o Cloud SDK Manager estiver desabilitado, você poderá receber um erro ao executar o guboud components install kubectl. Se isso ocorrer, você poderá usar o gerenciador de componentes, seguindo as instruções em

outro gerenciador de pacotes. Se você quiser usar o
gerenciador de componentes , você pode instalá-lo usando um destes métodos:

Pacotes adicionais estão disponíveis em nossos repositórios deb e yum; todos os mesmos
componentes estão disponíveis e você só precisa usar o
gerenciador de pacotes existente para instalá-los.

Você pode então usar o kubectl para executar uma imagem do Docker em um cluster usando o
comando kubectl run . Aqui está um exemplo:
kubectl run ch07-app-deploy --image = ch07-app --port = 8080
Isso executará uma imagem do Docker chamada ch07-app e tornará sua rede acessível na porta
8080. Se depois de algum tempo você quiser Para aumentar o número de réplicas na implantação,
você pode usar o comando de
escala kubectl : implementação da escala kubectl ch07-app-deploy --replicas = 5
Este exemplo criaria cinco réplicas.
Monitoramento O Kubernetes
Stackdriver é o abrangente produto de monitoramento, registro e alerta do GCP. Pode ser
usado para monitorar clusters do Kubernetes.
Ao criar um cluster, ative o monitoramento e o registro em log do Stackdriver selecionando Opções avançadas no formulário Criar cluster no Cloud Console. Em
Recursos Adicionais , escolha Ativar Serviço de Registro em Log e Ative o Serviço de Monitoramento, conforme mostrado na
Figura 7.10.

F igure 7.10 Ao expandir as Opções avançadas na caixa de diálogo Criar cluster, serão exibidas
duas caixas de seleção para ativar o registro e o monitoramento do Stackdriver.
Para configurar o Stackdriver no Cloud Console, selecione Stackdriver no menu de nível superior
à esquerda. Inicialmente, você precisará criar um espaço de trabalho em seu projeto selecionando uma nova área de
trabalho e iniciando o monitoramento quando solicitado (veja a Figura 7.11). Uma vez uma área de trabalho
é criado, você pode monitorar seus recursos do GCP, incluindo os clusters do Kubernetes.
Os espaços de trabalho são recursos para monitoramento e podem suportar até 100 projetos monitorados.

Os espaços de trabalho contêm painéis, políticas de alerta, definições de grupo e
verificações de notificação .
Figura 7.11 Uma caixa de diálogo inicial para criar um espaço de trabalho no Stackdriver
Depois de criar um espaço de trabalho, abra o Stackdriver e ele exibe a
página Visão Geral de Monitoramento , mostrada na Figura 7.12.

F igure 7.12 Página Visão geral do Stackdriver Monitoring
Na página Visão geral, clique em Recursos e selecione Instâncias para listar as instâncias no
cluster. Isso exibe uma lista como na Figura 7.13.
Figura 7.13 Lista de instâncias em um cluster do
Kubernetes
Clique nos nomes de qualquer uma das instâncias para mostrar uma página detalhada de informações de monitoramento, como mostrado na Figura 7.14.
Figura 7.14 Uma página de monitoramento detalhada típica de uma instância em execução em um
cluster do Kubernetes
Na página Detalhes, você pode visualizar uma visão geral dos detalhes sobre a instância e visualizar
o uso da CPU, o E / S do disco e o tráfego da rede. Você também pode criar políticas de alerta para notificá-lo
se alguma condição, como alta utilização da CPU, ocorrer na instância. Quando você cria
alertas, eles podem ser aplicados a uma instância individual no cluster ou a todas as instâncias no
cluster.

Na página detalhada do Stackdriver, crie um alerta clicando no botão Criar política de alerta
. Isso exibe uma caixa de diálogo, como na Figura 7.15, a partir da qual você pode criar condições,
notificações e documentação. Você também pode nomear a política.
Figura 7.15 Ao criar uma política de alerta, este formulário permite que você especifique
componentes da política.
Quando você adiciona uma condição, um formulário como o mostrado na Figura 7.16 aparece.

F igura 7.16 O Stackdriver suporta vários tipos de condição.
Selecione Métrica Limite para exibir um formulário como a Figura 7.17, que mostra como especificar
um alerta sobre a utilização da CPU acima de 80 por cento por 5 minutos.

Figura 7.17 As condições de limite da métrica do Stackdriver baseiam-se em um conjunto de
recursos monitorados , como a utilização da CPU.
O Stackdriver precisará saber como notificá-lo se um alerta for acionado. Você pode especificar
sua escolha de canais de notificação no formulário Criar Nova Política de Alerta, conforme mostrado na
Figura 7.18. Os canais incluem email, webhooks e mensagens de texto SMS, bem como ferramentas de terceiros, como PagerDuty, Campfire e Slack.
O Stackdriver também oferece suporte a alertas mais avançados, incluindo integridade do processo,
verificações de tempo de atividade , limites agregados de grupos e taxas de mudança de métricas.
Vamos percorrer um exemplo de criação de uma política para monitorar a utilização da CPU.
164 Capítulo 7 ■ Computando com o Kubernetes
figura 7.18 O Stackdriver suporta vários tipos de condição.

Para obter mais detalhes sobre monitoramento, consulte o Capítulo 18. Para criar uma política para monitorar a utilização da CPU, navegue até a página de monitoramento no Stackdriver e
clique em Criar Política. Isso exibirá o formulário para criar uma política, que é um
processo de quatro etapas: criar uma condição, especificar um canal de notificação, adicionar uma
descrição e nomear a política. (Consulte a Figura 7.19.)
Monitorando o Kubernetes 165
F igure 7.19 Criando uma política para monitorar a utilização da CPU
166 Capítulo 7 ■ Computando com Kubernetes
Clique em Add Condition para exibir um formulário como o mostrado na Figura 7.20.
Figura 7.20 Adicionando uma condição a uma política
No parâmetro Filter, insira Contêiner GKE e Uso da CPU. Na
seção Configuração , especifique 80 por cento como o limite e 2 minutos como o período de tempo. Salve a
condição. Isso retornará ao formulário Criar Política. No parâmetro Notification, selecione
Email na lista suspensa, como mostra a Figura 7.21.
Monitorando o Kubernetes 167
F igura 7.21 Escolhendo um canal de notificação
Adicione uma descrição e um nome de política, como mostra a Figura 7.22.
Figura 7.22 Um formulário de criação de política preenchido
168 Capítulo 7 ■ Computação com Kubernetes
Salve a especificação de política para exibir um resumo de monitoramento, conforme mostrado na Figura 7.23.
Figura 7.23 Monitorando os resultados da política sobre o uso da CPU

Resumo

O Kubernetes Engine é um sistema de orquestração de contêiner para implantar aplicativos para serem executados em
clusters. O Kubernetes é arquitetado com um único gerenciador de cluster e nós de trabalho.
O Kubernetes usa o conceito de pods que são instâncias executando um contêiner. É possível
executar vários contêineres em um pod, mas isso ocorre com menos frequência do que os conjuntos de contêiner único
. Os ReplicaSets são controladores para garantir que o número correto de pods esteja em execução.
Implantações são conjuntos de pods idênticos. StatefulSets são um tipo de implantação usado para
aplicativos com estado.
Os clusters do Kubernetes podem ser implantados por meio do Cloud Console ou usando comandos do gcloud.
Você implanta aplicativos agrupando o aplicativo em um contêiner e usando o console ou
o comando kubectl para criar uma implementação que executa o aplicativo no cluster.
O Stackdriver é usado para monitorar instâncias em clusters. Você pode criar alertas e receber
notificações em vários canais.

Fundamentos do exame

Entenda que o Kubernetes é um sistema de orquestração de contêineres. O Kubernetes Engine é um
produto do GCP que fornece Kubernetes aos clientes do GCP. O Kubernetes gerencia contêineres
executados em um conjunto de instâncias de VM.

Fundamentos do exame 169

Entenda que o Kubernetes usa um único mestre de cluster que controla nós que executam
cargas de trabalho. O Kubernetes usa o mestre para coordenar a execução e monitorar a integridade de
vagens. Se houver um problema com um pod, o mestre pode corrigir o problema e reprogramar
o trabalho interrompido.
Ser capaz de descrever pods. Pods são instâncias únicas de um processo em execução, os serviços fornecem
um nível de indireção entre pods e clientes chamando serviços nos pods, um ReplicaSet
é um tipo de controlador que garante que o número correto de pods esteja em execução e uma
implementação é um conjunto de pods vagens idênticas.
Os Kubernetes podem ser implantados usando o Cloud Console ou usando comandos gcloud. Os

comandos do gcloud manipulam o serviço do Kubernetes Engine, enquanto os comandos do kubectl são usados
para gerenciar o estado interno dos clusters a partir da linha de comando. O comando base para
Trabalhar com o Kubernetes Engine é gcloud container. Observe que o gcloud e o kubectl possuem
diferentes sintaxes de comando. Os comandos kubectl especificam um verbo e, em seguida, um recurso, como na
implementação da escala kubectl ..., enquanto gcloud especifica um recurso antes do verbo, como nos
clusters de contêiner gcloud, criar. Implantações são criadas usando o Cloud Console ou na
linha de comando usando uma especificação YAML.
Implantações são conjuntos de pods idênticos. StatefulSets são um tipo de implantação usado para
aplicativos com estado. O Kubernetes é monitorado usando o Stackdriver. O Stackdriver pode ser
configurado para gerar alertas e notificá-lo em vários canais. Para monitorar o estado
de um cluster, você pode criar uma política que monitora uma métrica, como a utilização da CPU, e
envia notificações para o e-mail ou outros canais.
170 Capítulo 7 Computando com o Kubernetes
Este capítulo descreve como executar tarefas básicas de gerenciamento do Kubernetes, incluindo as seguintes:
- **Visualização do status dos clusters do Kubernetes**
- **Visualização de repositórios de imagem e detalhes da imagem**
- **Adição, modificação e remoção de nós**
- **Adição, modificação e removendo pods**
- **Adicionando, modificando e removendo serviços**

Você verá como realizar cada uma dessas tarefas usando o Google Cloud Console e o Cloud
SDK, que você pode usar localmente em suas máquinas de desenvolvimento, em máquinas virtuais do GCP
e usando o Cloud Shell.
Visualizando o status de um
cluster do Kubernetes
Supondo que você criou um cluster usando as etapas descritas no Capítulo 7, você pode visualizar o
status de um cluster do Kubernetes usando o Google Cloud Console ou os comandos do gcloud.
Visualizando o status dos clusters do Kubernetes Usando o
Cloud Console
A partir da página inicial do Cloud Console, abra o menu de navegação clicando no

ícone de três linhas empilhadas no canto superior esquerdo. Isso exibe a lista de serviços do GCP, conforme
mostrado na Figura 8.1.
Visualizando o status de um cluster do Kubernetes 177
Figura 8.1 Menu de navegação no Console do Google Cloud
Selecione o Kubernetes Engine nas listas de serviços, como mostra a Figura 8.2.
Figura 8.2 Selecionando o Kubernetes Engine no menu de navegação
180 Capítulo 8 ■ Gerenciando clusters do Kubernetes
Fixando serviços na parte superior do menu de navegação
Na Figura 8.2, o Kubernetes Engine foi "fixado", de forma que é exibido na parte superior. Você pode
fixar qualquer serviço no menu de navegação passando o mouse sobre o produto e clicando no
ícone de alfinete que aparece, como na Figura 8.3. Nessa figura, o Compute Engine e o Kubernetes Engine
já estão fixados e as Cloud Functions podem ser fixadas clicando no ícone de alfinete cinza.
Figura 8.3 Fixando um serviço no topo do menu de navegação
Após clicar no Kubernetes Engine no menu de navegação, você verá uma lista de
clusters, como na Figura 8.4, que mostra um único cluster chamado cluster padrão-1.
Figura 8.4 Exemplo de lista de clusters no Kubernetes Engine
Passe o mouse sobre o nome do cluster para destacá-lo, como na Figura 8.5, e clique no nome
para exibir os detalhes do cluster, como na Figura 8.6.
Figura 8.5 Clique no nome de um cluster para exibir seus detalhes.
Visualizando o status de um cluster do Kubernetes 179
Figura 8.6 A primeira parte da página Detalhes do cluster descreve a configuração do
cluster.
Clicar nos links Complementos e Permissões exibe informações como as mostradas na
Figura 8.7. A seção Complementos exibe o status dos recursos complementares opcionais de um cluster.
A seção Permissões mostra quais APIs de serviço do GCP estão ativadas para o cluster.
180 Capítulo 8 ■ Gerenciamento de clusters do Kubernetes
F igure 8.7 Detalhes de complemento e permissão para um cluster A
Figura 8.8 mostra detalhes de exemplo de pools de nós, que são grupos de instâncias separadas em execução em um cluster do Kubernetes. Os detalhes nesta seção incluem a imagem do nó
em execução nos nós, o tipo de máquina, o número total de vCPUs (listado como Núcleos Totais), o
tipo de disco e se os nós são preemptivos.

Abaixo do nome do cluster, há uma lista horizontal de três opções: Detalhes, Armazenamento e
Nós. Até agora, descrevemos o conteúdo da página Detalhes. Clique em Armazenamento para exibir
informações como na Figura 8.9, que exibe volumes persistentes e as classes de armazenamento
usadas pelo cluster.
Este cluster não possui volumes persistentes, mas usa armazenamento padrão. Volumes persistentes são discos duráveis gerenciados pelo Kubernetes e implementados usando
discos permanentes do Compute Engine. Uma classe de armazenamento é um tipo de armazenamento com um conjunto de políticas que especifica a
qualidade do serviço, a política de backup e um provisionador (que é um serviço que implementa o
armazenamento).
Visualizando o status de um cluster do Kubernetes 181
Figura 8.8 Detalhes sobre os pools de nós no cluster
F igura 8.9 Informações de armazenamento sobre um cluster
182 Capítulo 8 ■ Gerenciando clusters do Kubernetes
Na opção Nós do menu de status do cluster, você pode ver uma lista de nós ou VMs
em execução no cluster, conforme mostrado na Figura 8.10. A lista de nós mostra
informações básicas de configuração .
Figura 8.10 Listagem de nós no cluster
Clique no nome de um dos nós para ver informações detalhadas sobre o status, como na
Figura 8.11. Os detalhes do nó incluem utilização da CPU, consumo de memória e IO de disco.
Há também uma lista de pods em execução no nó.
Figura 8.11 Exemplo de detalhes de um nó em execução em um cluster
do Kubernetes 183
Clique no nome de um pod para ver seus detalhes. A exibição do pod é semelhante à exibição do nó
com estatísticas de CPU, memória e disco. Os detalhes da configuração incluem quando o pod foi criado, os rótulos atribuídos, os links para os logs e o status (que é mostrado como em execução na Figura 8.12).
Outros status possíveis são pendentes, o que indica que o pod está baixando imagens;
Sucedido, o que indica que o pod terminou com sucesso; Falhou, o que indica que pelo menos
um contêiner falhou; e Desconhecido, o que significa que o mestre não pode alcançar o nó e o
status não pode ser determinado.
F igure 8.12 Exibição do status do pod, com status como

Em execução Na parte inferior da exibição do pod, há uma lista de containers em execução. Clique no nome de um
contêiner para ver seus detalhes. A Figura 8.13 mostra os detalhes do container chamado
evento-exportador. As informações incluem o status, a hora de início, o comando em
execução e os volumes montados.

F igure 8.13 Detalhes de um contêiner em execução em um pod
Usando o Cloud Console, você pode listar todos os clusters e exibir detalhes de suas configurações e
status. Você pode então detalhar cada nó, pod e contêiner para visualizar seus detalhes.
Visualizando o status dos clusters do Kubernetes Usando o Cloud
SDK e o Cloud Shell
Você também pode usar a linha de comando para visualizar o status de um cluster. O
comando da lista de clusters do contêiner gcloud é usado para mostrar esses detalhes.
Para listar os nomes e informações básicas de todos os clusters, use este comando:
gcloud container clusters list
Produz a saída mostrada na Figura 8.14.
Figura 8.14 Exemplo de saída do comando gcloud container clusters list
Exibindo o status de um cluster do Kubernetes 185
Por que os comandos não começam com gcloud kubernetes?
Os comandos gcloud começam com a palavra gcloud seguida do nome do serviço, por
exemplo, gcloud compute para comandos do Compute Engine e gcloud sql para
comandos do Cloud SQL. Você pode esperar que os comandos do Kubernetes Engine iniciem com
gcloud kubernetes, mas o serviço era originalmente chamado de Google Container Engine.
Em novembro de 2017, o Google renomeou o serviço Kubernetes Engine, mas os
comandos do gcloud continuaram os mesmos.
Para visualizar os detalhes de um cluster, use o comando gcloud container clusters describe. Você precisará passar o nome de uma zona ou região usando o
parâmetro --zone ou --region . Por exemplo, para descrever um cluster chamado cluster padrão-1 localizado na zona
us-central1-a, você usaria este comando:
gcloud container clusters describe --zone us-central1-a-cluster-padrão-1
Isso exibirá detalhes como os mostrados na Figura 8.15 e na Figura 8.16. Observe que

o comando describe também exibe informações de autenticação, como certificado de cliente,
nome de usuário e senha. Essa informação não é mostrada nas figuras.
Figura 8.15 A parte 1 das informações exibidas pelos agrupamentos de contêineres do gcloud
descreve o comando
186 Capítulo 8 ■ Gerenciamento de clusters do
Kubernetes Figura 8.16 A parte 2 das informações exibidas pelos agrupamentos de contêiner do gcloud
descreve o comando
Para listar informações sobre nós e pods, use o comando kubectl.
Primeiro, você precisa garantir que tenha um arquivo kubeconfig configurado corretamente, que
contém informações sobre como se comunicar com a API do cluster. Execute o comando
gcloud container clusters get-credentials com o nome de uma zona ou região e o
nome de um cluster. Veja um exemplo:
gcloud container clusters get-credenciais --zone us-central1-a-padrão-cluster-1
Isso configurará o arquivo kubeconfig em um cluster denominado standard-cluster-1 no
use-central1-a zone. A Figura 8.17 mostra um exemplo de saída desse comando, que
inclui o status de obtenção e configuração de dados de autenticação.
Ver o estado de um Kubernetes cluster 187
F igura 8,17 Exemplo de saída dos get-credenciais comando
Pode listar os nós de um cluster usando o seguinte:
kubectl obter nodos
Isto produz uma saída, tal como na Figura 8.18, o qual mostra o estado de três nós .
Figura 8.18 Exemplo de saída do comando kubectl get nodes
Da mesma forma, para listar pods, use o seguinte comando:
kubectl get pods
Produz saída, como na Figura 8.19, que lista pods e seu status.
Figura 8.19 Exemplo de saída do comando kubectl get pods
188 Capítulo 8 ■ Gerir Kubernetes Clusters
Para mais detalhes sobre os nós e vagens, utilizar estes comandos:
kubectl descrever nodos
kubectl descrever vagens
Figuras 8,20 e 8,21 mostrar anúncios parciais dos resultados. Observe que o
comando kubectl describe pods também inclui informações sobre contêineres, como nome, rótulos,
condições, endereços de rede e informações do sistema.
Figura 8.20 Lista parcial dos detalhes mostrados pelo
comando kubectl describe nodes
Visualizando o Status de um Kubernetes Cluster 191

Figura 8.21 Listagem parcial dos detalhes mostrados pelo comando kubectl describe pods

Para visualizar o status de clusters a partir da linha de comando, use os comandos do contêiner gcloud, mas para obter informações sobre objetos gerenciados do Kubernetes, como nós, pods e contêineres, use o comando kubectl.

Adicionando, modificando e removendo nós

Você pode adicionar, modificar e remover nós de um cluster usando o Cloud Console ou o Cloud SDK em seu ambiente local, em uma máquina virtual GCP ou no Cloud Shell.

Adicionando, modificando e removendo nós com o Cloud Console No

Cloud Console, navegue até a página do Kubernetes Engine e exiba uma lista de clusters.

Clique no nome de um cluster para exibir seus detalhes, como na Figura 8.22. Observe a opção Editar perto do topo da tela. Clique para abrir um formulário de edição.

Figura 8.22 Detalhes de um cluster no Cloud Console

Role para baixo até a seção Node Pools, que lista o nome, tamanho, imagem do nó, tipo de máquina e outras informações sobre o cluster. O parâmetro de tamanho é opcional. No exemplo mostrado na Figura 8.23, o cluster possui três nós.

Figura 8.23 Detalhes de um pool de nós no Cloud Console

Para adicionar nós, aumente o tamanho para o número de nós que você deseja. Para remover os nós, diminua o tamanho para o número de nós que você gostaria de ter.

Adicionando, modificando e removendo com o Cloud SDK e o Cloud Shell

O comando para adicionar ou modificar nós é o redimensionamento dos clusters de contêiner gcloud. O comando usa três parâmetros, conforme mostrado aqui:

■ nome do cluster ■ nome do pool de nós ■ tamanho do cluster

Por exemplo, suponha que você tenha um cluster denominado cluster padrão 1, executando um pool de nós chamado pool padrão. Para aumentar o tamanho do cluster de 3 para 5, use este comando:

```
gcloud container clusters resize standard-cluster-1 --node-pool default-pool
--size 5 --region = us-central1
```

Depois que um cluster for criado , você pode modificá-lo usando o comando gcloud container clusters

update. Por exemplo, para ativar o escalonamento automático, use o comando update para especificar
o número máximo e mínimo de nós. O comando para atualizar um cluster chamado
O cluster padrão 1 em execução em um pool de nós denominado pool padrão é o seguinte:
gcloud cluster clusters atualização standard-cluster-1 --enable-autoscaling
**--min-nodes 1 **
--max-nodes 5 --zone us- central1-a --node-pool pool padrão
Mantendo a demanda com escalonamento automático
Frequentemente, é difícil prever a demanda em um serviço. Mesmo se houver padrões regulares,
como grandes trabalhos em lote executados durante horários não comerciais, pode haver variação quando esses
picos de carga forem executados. Em vez de continuar alterando manualmente o número de vCPUs em um cluster,
ative o escalonamento automático para adicionar ou remover automaticamente nós, conforme necessário, com base na demanda.
O escalonamento automático pode ser ativado ao criar clusters com o Cloud Console ou o gcloud.
Essa abordagem é mais resistente a picos inesperados e mudanças nos padrões de longo prazo de
pico de uso. Isso também ajudará a otimizar o custo de seu cluster, não executando muitos servidores quando não for necessário. Ele também ajudará a manter o desempenho por ter nós suficientes para
atender à demanda.
192 Capítulo 8 ■ Gerenciando clusters Kubernetes
Adicionando, modificando e
removendo pods
Você pode adicionar, modificar e remover pods de um cluster usando o Cloud Console ou o
Cloud SDK em seu ambiente local, em uma VM do GCP ou no Cloud Shell.
Considera-se uma prática recomendada não manipular os pods diretamente. O Kubernetes manterá
o número de pods especificados para uma implantação. Se você gostaria de mudar o número de
pods, você deve alterar a configuração de implantação.
Adicionar, modificar e remover pods com o
Cloud Console
Pods é gerenciado por meio de implantações. Uma implantação inclui um parâmetro de configuração
chamado réplicas, que é o número de pods que executam o aplicativo especificado na
implantação. Esta seção descreve como usar o Cloud Console para alterar o número de
réplicas, o que, por sua vez, altera o número de pods.

No Cloud Console, selecione as opções Cargas de trabalho no menu de navegação à esquerda.
Isso exibe uma lista de implementações, como na Figura 8.24.
Figura 8.24 Lista de implementações em um cluster
Clique no nome da implantação que você deseja modificar; um formulário é exibido com detalhes
como na Figura 8.25. Observe a opção Ações no menu horizontal superior.
Adicionando, modificando e removendo pods 193
F igure 8.25 Vários formulários contêm detalhes de uma implantação e incluem um menu de
ações que você pode executar na implantação.
Clique em Ações para listar as opções, que são Autoscale, Expose, Rolling Update e
Scale, conforme mostrado na Figura 8.26.
Figura 8.26 Lista de ações disponíveis para implantações
Selecione Escala para exibir uma caixa de diálogo que permite definir um novo tamanho para a carga de trabalho, conforme
mostrado na Figura 8.27. Neste exemplo, o número de réplicas foi alterado para 2.
Figura 8.27 Defina o número de réplicas para uma implantação.
Você também pode fazer com que o Kubernetes adicione e remova automaticamente réplicas (e pods), dependendo da necessidade, especificando o escalonamento automático. Escolha Escalonamento automático no menu para exibir o
formulário mostrado na Figura 8.28. Você pode especificar um número mínimo e máximo de réplicas
para execução aqui.
194 Capítulo 8 ■ Gerenciamento de clusters do Kubernetes
F igura 8.28 Habilite o escalonamento automático para adicionar e remover automaticamente as réplicas conforme necessário,
dependendo da carga.
O menu Action também fornece opções para expor um serviço em uma porta, conforme mostrado na
Figura 8.29, e para especificar parâmetros para controlar atualizações contínuas no código implementado, conforme
mostrado na Figura 8.30. Os parâmetros incluem o mínimo de segundos para aguardar antes
considerando o pod atualizado, o número máximo de pods acima do tamanho desejado permitido e
o número máximo de pods não disponíveis.
Figura 8.29 Formulário para expor serviços em execução em pods
Adicionando, modificando e removendo pods 195
Formulário do F igure 8.30 para especificar parâmetros para atualizações contínuas de código em execução em pods
Adicionando, modificando e removendo pods com o Cloud SDK
e o Cloud Shell

Trabalhando com pods em O Cloud SDK e o Cloud Shell são feitos trabalhando com implantações;
As implantações foram explicadas anteriormente na seção "Adicionando, modificando e removendo pods
com o Cloud Console". Você pode usar o comando kubectl para trabalhar com implantações.
Para listar implantações, use o seguinte comando:
kubectl get deployments
Isso produzirá uma lista de implementações, como na Figura 8.31.
Figura 8.31 Uma lista de implementações na linha de comando
Para adicionar e remover pods, altere a configuração das implantações usando o
comando kubectl scale deployment. Para este comando, você tem que especificar o nome da implementação

e número de réplicas. Por exemplo, para definir o número de réplicas como 5 para uma implantação
denominada nginx-1, use:
kubectl scale deployment nginx-1 --replicas 5
Para que o Kubernetes gerencie o número de pods com base na carga, use o comando de autoescala. O comando a seguir adicionará ou removerá os conjuntos conforme necessário para atender à demanda
na utilização da CPU. Se o uso da CPU exceder 80%, até 10 pods ou réplicas
adicionais serão adicionados. A implantação sempre terá pelo menos um pod ou réplica.
kubectl implantação autoescala nginx-1 --max 10 --min 1 --cpu-porc 80
Para remover uma implantação, use o comando delete deployment da seguinte forma:
kubectl delete deployment nginx-1
Adicionando, modificando e removendo serviços
Você pode adicionar, modificar e remover serviços de um cluster usando o Cloud Console ou o
Cloud SDK em seu ambiente local, em uma VM do GCP ou no Cloud Shell.
Um serviço é uma abstração que agrupa um conjunto de conjuntos como um único recurso.
Adicionando, modificando e removendo serviços com o
Cloud Console
Os serviços são adicionados por meio de implantações. No Cloud Console, selecione a opção Cargas de trabalho
no menu de navegação para exibir uma lista de implantações, como na Figura 8.32. Observe a
opção Implantar no menu horizontal na parte superior da página.
Figura 8.32 Lista de implementações junto com um comando Implantar para criar novos serviços

Clique em Implantar para mostrar o formulário de implantação, como na Figura 8.33.
Figura 8.33 Formulário para especificar uma nova implantação para um serviço
No parâmetro Imagem do Contêiner, você pode especificar o nome de uma imagem ou selecionar uma
do Repositório de Contêineres do Google. Para especificar um nome diretamente, especifique um caminho para a
imagem usando um URL como este:
gcr.io/google-samples/hello-app:2.0
Você pode especificar rótulos, o comando inicial a ser executado e um nome para seu aplicativo.
Quando você clica no nome de uma implementação, como as listadas anteriormente na Figura 8.32,
você verá detalhes dessa implementação, incluindo uma lista de serviços, como a mostrada na
Figura 8.34.
198 Capítulo 8 ■ Gerenciando Kubernetes Clusters
Figura 8.34 Detalhes de um serviço em execução em uma implantação
Clicar no nome de um serviço abre o formulário Detalhe do serviço, que inclui uma
opção Excluir no menu horizontal, conforme mostrado na Figura 8.35.
Figura 8.35 Navegue até a página Detalhes do serviço para excluir um serviço usando a
opção Excluir no menu horizontal.
Adicionando, modificando e removendo serviços com o Cloud
SDK e o Cloud Shell
Use o comando kubectl get services para listar serviços. A Figura 8.36 mostra uma
listagem de exemplo .
Figura 8.36 Uma lista de serviços exibidos por um comando kubectl get services
Visualizando o Repositório de Imagens e Detalhes da Imagem 199
Para adicionar um serviço, use o comando kubectl run para iniciar um serviço. Por exemplo, para incluir
um serviço chamado hello-server usando o aplicativo de amostra com o mesmo nome fornecido
Google, use the following command:
kubectl run hello-server --image=gcr.io/google/samples/hello-app:1.0 --port 8080
This command will download and start running the image found at the path gcr.io/
google-samples/ called hello-app, version 1. It will be accessible on port 8080. Services
need to be exposed to be accessible to resources outside the cluster. This can be set using
the expose command, as shown here:
kubectl expose deployment hello-server --type="LoadBalancer"

This command exposes the services by having a load balancer act as the endpoint for
outside resources to contact the service.
To remove a service, use the delete service command, as shown here:
kubectl delete service hello-server
Visualizando o Repositório de
Imagens e o
Registro de Contêiner de Detalhes da Imagem é um serviço GCP para armazenar imagens de contêiner. Depois de criar um
registro e enviar imagens para ele, você poderá visualizar o conteúdo do registro e os detalhes da imagem.
usando o Cloud Console, o Cloud SDK e o Cloud Shell.
Visualizando o repositório de imagens e os detalhes da imagem com o
Cloud Console
No Cloud Console, selecione Registro de contêineres no menu de navegação para exibir o
conteúdo de um registro. A Figura 8.37 mostra uma listagem de exemplo com três imagens para Nginx,
Redis e WordPress.
200 Chapter 8 ■ Managing Kubernetes Clusters
F igure 8.37 A listing of images in a Container Registry
To see the details of an image, click the image name. For example, Figure 8.38 shows
a listing for the Nginx image. This listing will list one entry for each version of the image.
Since there is only one version of the image, there is only one listed.
F igure 8.38 A list of versions for an image
To see the details of that version, click the version name. This displays a listing such as
in Figure 8.39, which includes the image type, size, and time created.
Viewing the Image Repository and Image Details 201
F igure 8.39 Details of a version of an image
202 Chapter 8 ■ Managing Kubernetes Clusters
Viewing the Image Repository and Image Details with
Cloud SDK and Cloud Shell
From the command line, you work with images in a registry using
gcloud container images commands. For example, to list the contents of a registry, use this:
gcloud container images list
Esse comando produz uma lista de imagens, como na Figura 8.40. Você também pode listar
contêineres do Google usando a lista de imagens do contêiner gcloud --repository gcr.io/
google-containers.
F igure 8.40 List of images in a container repository

To view the details of an image, use the describe command and pass in the name of the
image as an argument. For example, the following command:
gcloud container images describe gcr.io/appengflex-project-1/nginx
will produce an output list such as that shown in Figure 8.41. You can also describe a
Google image with a command such as gcloud container images describe gcr.io/google-containers/toolbox.
F igure 8.41 A listing of image details produced by the describe image command
Kubernetes Engine makes use of container images stored in a Container Repository. The
contents of the Container Repository can be viewed in summary and in detail using both
O Cloud Console e o Cloud SDK de linha de comando, inclusive no Cloud Shell.
Fundamentos do exame 203
Resumo
Neste capítulo, você aprendeu a executar tarefas básicas de gerenciamento para trabalhar com
Kubernetes clusters, nós, pods e serviços. O capítulo também descreveu como listar o
conteúdo dos repositórios de imagens de contêiner. Você aprendeu como fixar serviços no
menu do Cloud Console, ver o status dos clusters do Kubernetes e visualizar os detalhes do repositório de imagens e da imagem usando os comandos do gcloud. Este capítulo também descreveu como modificar e remover
nós e pods. Você também viu os benefícios do escalonamento automático em um cenário do mundo real.
O Cloud Console e o Cloud SDK, incluindo o Cloud Shell, podem ser usados para adicionar, remover
e modificar nós, pods e serviços. Ambos podem ser usados para revisar o conteúdo de um
repositório de imagens. Alguns dos comandos mais úteis incluem clusters de contêiner gcloud
crie e gcloud clusters de contêiner redimensionar. O comando kubectl é usado para
modificar recursos do Kubernetes, como implantações e pods.
Fundamentos do Exame
Saiba como visualizar o status de um cluster do Kubernetes. Use o Cloud Console para listar clusters
e detalhar os clusters para ver os detalhes do cluster, incluindo os detalhes do nó, do pod e do contêiner
. Conheça o comando gcloud container clusters e suas opções.
Entenda como adicionar, modificar e remover nós. Use o Cloud Console para modificar os nós

e saber como adicionar e remover nós, alterando as implantações. Use o comando de redimensionamento de clusters de contêiner do gcloud para adicionar e remover nós.
Entenda como adicionar, modificar e remover pods. Use o Cloud Console para modificar os pods
e adicionar e remover pods alterando as implantações. Use o kubectl para obter implementações para
listar implantações, implementação em escala kubectl para modificar o número de implementações e
implementação em escala automática kubectl para habilitar escalonamento automático.
Entenda como adicionar, modificar e remover serviços. Use o Cloud Console para modificar
serviços e adicionar e remover serviços alterando as implantações. Use o kubectl run para iniciar os
serviços e o kubectl exponha a implementação para tornar um serviço acessível fora do cluster.
Exclua um serviço usando o comando kubectl delete service.
Saiba como visualizar as imagens do Container Registry e seus detalhes. Navegue pelas
páginas do Registro de contêiner no Cloud Console. Conheça a lista de imagens do contêiner gcloud e gcloud
imagens de contêiner descrevem comandos.

Google Cloud Platform em ação
Jose RF Junior - web2ajax@gmail.com
Resumo

10.6. Mantendo seu cluster
10.6.1. Atualizando o nó do Kubernetes
10.6.2. Atualizando Nós do Cluster
10.6.3. Redimensionando seu cluster
10.7. Entendendo o preço
10.8. Quando devo usar o Kubernetes Engine?
10.8.1. Flexibilidade
10.8.2. Complexidade
10.8.3. atuação
10.8.4. Custo
10.8.5. Sem geral
10.8.6. Lista de afazeres
10.8.7. E * Exchange
10.8.8. InstaSnap

10. Kubernetes Engine: gerenciado
Clusters do Kubernetes
Este capítulo cobre
Quais recipientes, o Docker e o Kubernetes fazem
Como o Kubernetes Engine funciona e quando é um bom ajuste
Configurando um cluster gerenciado do Kubernetes usando o Kubernetes Engine
Atualizando nós do cluster e redimensionando um cluster
Um problema comum no desenvolvimento de software é a embalagem final
todo o seu trabalho duro em algo que é fácil de trabalhar em um
configuração de produção. Este problema é frequentemente negligenciado até o último
minuto porque tendemos a manter nosso foco na construção e concepção
o software em si. Mas a embalagem final e implantação são muitas vezes
tão difícil e complexo quanto o desenvolvimento original. Felizmente muitos
ferramentas estão disponíveis para resolver este problema, um dos quais depende
conceito de um recipiente para o seu software.
10.1. O que são contêineres?
Um contêiner é uma ferramenta de infra-estrutura destinada a resolver o software
problema de implantação facilitando o empacotamento de seu aplicativo,
sua configuração e quaisquer dependências em um formato padrão. Por
contando com contêineres, torna-se fácil compartilhar e replicar
ambiente de computação em muitas plataformas diferentes. Recipientes
também atua como uma unidade de isolamento, então você não precisa se preocupar
competindo por recursos limitados de computação - cada contêiner é
isolado de todos os outros. Se tudo isso soa intimidante, não se preocupe: os recipientes são bonitos
confuso quando você está começando a aprender sobre eles. Vamos andar

através de cada peça, um passo de cada vez, começando com a configuração.

10.1.1. Configuração

Se você já tentou implantar seu aplicativo e percebeu que tinha um
Muito mais dependências do que você pensou, você não está sozinho. Esse problema
pode fazer um dos benefícios da computação em nuvem (facilmente criado
máquinas virtuais de ardósia) um pouco de dor! Sendo engenheiros, nós
inventou muitas maneiras de lidar com isso ao longo dos anos (por exemplo,
usando um script de shell que é executado quando uma máquina é inicializada),
mas a configuração
continua sendo um problema frustrante. Recipientes resolvem este problema
facilitando a configuração de um sistema limpo, descreva como você deseja
olhe, e mantenha um instantâneo disto uma vez que parece exatamente correto.
Mais tarde, você
pode inicializar um contêiner e fazer com que ele pareça exatamente como você descreveu.

Você pode estar pensando nos snapshots do disco permanente que você aprendeu
sobre no capítulo 9 e se perguntando por que você não deve usá-los para
gerenciar sua configuração. Embora isso seja totalmente razoável,
sofre de um grande problema: esses instantâneos só funcionam no Google!
Esse problema nos leva à próxima questão: padronização.

10.1.2. estandardização

Há muito tempo (pré-1900) (figura 10.1), se você quisesse enviar um
mesa e algumas cadeiras do outro lado do oceano da Inglaterra para o Reino
Estados, você tinha que levar tudo para um navio e descobrir como encaixá-lo
dentro, como jogar um jogo da vida real de Tetris. Foi como
empacotando suas coisas em uma van em movimento - apenas maior - e você compartilhou
a experiência com todos os outros que estavam colocando suas coisas nela.

Figura 10.1. Envio antes de contêineres

Eventualmente, a indústria naval decidiu que esta forma de embalagem
as coisas foram bobas e começaram a explorar a ideia de conteinerização.
Em vez de embalar coisas como peças de quebra-cabeça, as pessoas resolvem o quebra-cabeça
usando grandes caixas de metal (recipientes) antes mesmo de chegar
um barco. Dessa forma, a tripulação do barco só lida com esses padrões
recipientes de tamanho e nunca tem que jogar Tetris novamente. Além de
reduzindo o tempo necessário para carregar os barcos, padronizando um
tipo de caixa com dimensões específicas significava que a indústria naval
poderia construir barcos que eram bons em segurar recipientes (figura 10.2),
conceber ferramentas boas para carregar e descarregar contentores e
cobrar preços com base no número de contêineres. Tudo isso facilitando as coisas, tornando-as mais eficientes e mais baratas.

Figura 10.2. Frete usando contêineres

Recipientes de software fazem pelo seu código o que grandes caixas de metal fizeram para
Remessa. Eles agem como um formato padrão representando seu software
e seu ambiente e oferecem ferramentas para executar e gerenciar
ambiente para que funcione em todas as plataformas. Se um sistema entender
recipientes, você pode ter certeza que quando você implantar o seu código lá, ele
trabalhos. Mais concretamente, você pode se concentrar especificamente em obter
código em um recipiente, jogando Tetris na frente em vez de quando você está
tentando implantar para produção. Uma última peça aqui precisa ser mencionada,
e vem como subproduto do uso de containers: isolamento.

10.1.3. Isolamento

Uma coisa que você pode notar na primeira foto de remessa (figura 10.1) é
que transportar coisas antes dos contêineres parecia um pouco arriscado, porque
suas coisas podem ser esmagadas por outras coisas mais pesadas. Felizmente, por dentro
um contêiner para remessa ou para o seu código, você só se preocupa com as coisas de sua empresa. Por exemplo, você pode querer usar uma máquina grande e
dividir em duas partes: uma para um servidor web e outra para um
base de dados. Sem um contêiner, se o banco de dados recebesse toneladas de SQL
consultas, o servidor da Web teria muito menos ciclos de CPU para manipular
solicitações da web. Mas usando dois recipientes separados faz este problema
vá embora. Recipientes físicos têm paredes para impedir que um piano
esmagando suas coisas, e contêineres de software são executados em um virtual
ambiente com paredes semelhantes que permitem decidir exatamente como
para alocar os recursos subjacentes.
Além disso, embora os aplicativos executados no mesmo
máquina pode compartilhar as mesmas bibliotecas e sistema operacional, eles
pode nem sempre fazer isso. Quando aplicativos em execução no mesmo
sistema requerem diferentes versões de bibliotecas compartilhadas, reconciliando
demandas podem se tornar bastante complicadas. Ao contentar o
bibliotecas compartilhadas não são mais compartilhadas, o que significa
dependências são isoladas para uma única aplicação (figura 10.3).
Figura 10.3. Aplicações sem contêineres vs. contêineres Agora você entende os benefícios da configuração, padronização,
e isolamento. Com esses benefícios em mente, vamos pular uma camada no
empilhar e pensar sobre o navio que irá armazenar todos esses contêineres,
e o capitão que comandará o navio.

10,2. O QUE É DOCKER?

Muitos sistemas são capazes de executar ambientes virtualizados, mas um assumiu a liderança nos últimos anos: o Docker. Docker é um ferramenta para a execução de contêineres e atua um pouco como um moderno navio porta-contêineres que transporta todos os contêineres de um lugar para outro. Em um nível fundamental, o Docker lida com a virtualização de nível inferior; isto pega as definições de imagens de container e executa o ambiente e código que os contêineres definem.

Além de ser o sistema base mais comum para correr

O Docker também se tornou o padrão de como você define as imagens do container, usando um formato chamado Dockerfile. Dockerfiles permite você define um container usando um monte de comandos que fazem qualquer coisa da execução de um comando shell simples (por exemplo, RUN echo "Hello World!") Até coisas mais complexas como expondo uma única porta fora do contêiner (EXPOSE 8080) ou herdando de outro contêiner predefinido (nó FROM: 8). eu vou consulte novamente o formato Dockerfile ao longo deste capítulo, e embora você deva entender o que esse tipo de arquivo está tentando não se preocupe se você não se sentir confortável escrevendo do princípio. Se você se aprofundar nos contêineres, livros inteiros Docker estão disponíveis para ajudá-lo a aprender como escrever um Dockerfile de sua preferência.

Nota

Se você quiser seguir junto com o código e implantação neste capítulo, você deve instalar o tempo de execução do Docker em sua máquina local, que está disponível em http://docker.com/community-edition para a maioria plataformas.

10.3. O QUE É KUBERNETES?

Se você começar a usar containers, será natural dividir as coisas com base no que eles são responsáveis (figura 10.4). Por exemplo, se você estava criando um aplicativo da Web tradicional, você pode ter um contêiner que lida com solicitações da Web (por exemplo, um servidor de aplicativos da Web que lida com solicitações baseadas em navegador), outro contêiner que manipula cache de dados acessados com freqüência (por exemplo, executando um serviço como Memcached), e outro contêiner que lida com trabalhos mais complexos, como gerar relatórios sofisticados, encolher imagens para miniaturas tamanho, ou enviar e-mails para seus usuários.

Figura 10.4. Visão geral de um aplicativo da Web como contêineres

Gerenciando onde todos esses contêineres são executados e como eles conversam com um outro acaba por ser complicado. Por exemplo, você pode querer todos os

servidores de aplicativos da web para que o Memcached seja executado no mesmo físico (ou
virtual) para que você possa conversar com o Memcached
localhost em vez de um IP público. Como resultado, há um monte de
sistemas que tentam resolver este problema, um dos quais é o Kubernetes.
O Kubernetes é um sistema que gerencia seus contêineres e permite que você
para dividir as coisas em pedaços que fazem sentido para a sua aplicação,
independentemente do hardware subjacente que executa o código. Também permite expressar relações mais complexas, como o fato de você querer
qualquer VM que manipule solicitações da web para ter o Memcached no mesmo
máquina. Além disso, porque é open source, usá-lo não amarra você a um
único provedor de hospedagem. Você pode executá-lo em qualquer provedor de nuvem, ou você
pode pular para fora da nuvem usando seu próprio hardware. Façam
tudo isso, o Kubernetes baseia-se no conceito de um contêiner como
unidade fundamental e introduz vários novos conceitos que você pode
use para representar seu aplicativo e a infraestrutura subjacente.
Vamos explorá-los nas próximas subseções.

Nota

O Kubernetes é uma plataforma enorme que vem evoluindo por vários
anos e se tornando mais e mais complexo com o passar do tempo,
o que significa que é muito grande para encaixar tudo em um único capítulo. Como um
Como resultado, vou me concentrar em demonstrar como você pode usar
Kubernetes. Se você quiser saber mais sobre o Kubernetes, você pode
quero dar uma olhada no livro de Marko Luksa, Kubernetes in Action
(Manning, 2017).

Porque há muito o que falar sobre o Kubernetes, vamos começar
olhando para um diagrama grande e assustador mostrando a maioria dos conceitos básicos
Kubernetes (figura 10.5). Vamos então ampliar as quatro teclas
conceitos: clusters, nós, pods e serviços.

Figura 10.5. Uma visão geral dos principais conceitos do Kubernetes

10.3.1. Clusters

Na parte superior do diagrama, você verá o conceito de um cluster, que é
a coisa que tudo o que vou falar vive dentro.
Os clusters tendem a se alinhar com um único aplicativo, então quando você está
falando sobre a implantação de todas as partes de um aplicativo,
você diria que todos eles correm como parte de seu cluster Kubernetes. Para
Por exemplo, você se referiria à implantação de produção de sua Lista de Tarefas Pendentes
app como seu cluster do Kubernetes da Lista de Tarefas Pendentes.

10.3.2. Nós

Os nós vivem dentro de um cluster e correspondem a uma única máquina (por

exemplo, uma VM no GCE) capaz de executar seu código. Nisso
exemplo de cluster, dois nós diferentes (chamados de Nó 1 e Nó 2) são
executando alguns aspectos do aplicativo Lista de tarefas. Cada cluster geralmente conterá vários nós, que são coletivamente responsáveis pelo manuseio
o trabalho geral necessário para executar seu aplicativo.
É importante enfatizar o aspecto coletivo dos nós, porque um único
nó não está necessariamente ligado a um único propósito. É totalmente possível que
um determinado nó será responsável por muitas tarefas diferentes de uma só vez, e
que essas tarefas podem mudar com o tempo. Por exemplo, no diagrama,
você tem o Nó 1 e o Nó 2 lidando com uma mistura de responsabilidades,
mas isso pode não ser o caso mais tarde, quando o trabalho embaralha
através dos nós disponíveis.

10.3.3. Vagens

Pods são grupos de containers que atuam como unidades discretas de funcionalidade
que qualquer nó dado será executado. Os recipientes que compõem um pod
todos serão mantidos juntos em um nó e compartilharão o mesmo endereço IP
e espaço de porta. Como resultado, os contêineres no mesmo pod podem
comunicar via localhost, mas eles não podem ambos ligam para o mesmo
porta; por exemplo, se o Apache estiver sendo executado na porta 80, o Memcached não poderá
também se liga a essa mesma porta. O conceito de um pod pode ser um pouco
confuso, então para esclarecer, vamos olhar para um exemplo mais concreto e
compare a versão tradicional com a versão estilo Kubernetes.
Uma pilha LAMP é um estilo de implantação comum que consiste em executar
Linux (como sistema operacional), Apache (para atender a solicitações da Web),
MySQL (para armazenar dados) e PHP (para fazer o trabalho da sua aplicação).
Se você estivesse executando um sistema desse tipo em um ambiente tradicional (figura
10.6), você pode ter um servidor executando o MySQL para armazenar dados, outro
rodando o Apache com o mod_php (para processar código PHP), e talvez
mais um executando o Memcached para armazenar em cache os valores (no mesmo
máquina como o servidor Apache ou um servidor separado) .Figura 10.6. Versão não-contêinerizada de uma pilha LAMP
Se você pensasse nessa pilha em termos de containers e pods,
pode reorganizar as coisas um pouco, mas a idéia importante a notar é sair
VMs (e nós) fora da imagem inteiramente. Você pode ter um pod
responsável por servir o aplicativo da Web (que estaria executando o Apache
e Memcached, cada um em seu próprio contêiner) e outro pod
responsável por armazenar dados (com um contêiner executando o MySQL) (figura

10.7).
Figura 10.7. Versão contêinerizada de um LAMP stackEstes pods podem estar em execução em uma única VM (ou nó) ou ser divididos
em duas VMs diferentes (ou nós), mas você não precisa se preocupar
onde um pod está sendo executado. Contanto que cada pod tenha computação suficiente
recursos e memória, deve ser irrelevante. A ideia de usar pods
é que você pode se concentrar no que deve ser agrupado, em vez de
como deve ser definido em hardware específico (seja virtual
ou físico).
Olhando para isso do ponto de vista do aplicativo Lista de tarefas, você tinha
dois pods diferentes: o pod de aplicativo da lista de tarefas e o relatório
pod de geração. Um exemplo do pod de lista de tarefas é mostrado na figura
10.8, que é semelhante à pilha LAMP que descrevi, com dois
contêineres: um para solicitações da Web e outro para dados em cache.
Figura 10.8. O pod de lista de afazeresEmbora a capacidade de organizar diferentes funcionalidades em muitos
máquinas físicas diferentes é legal, você pode estar preocupado com as coisas
ficando perdido. Por exemplo, como você sabe para onde enviar web
pedidos para o seu aplicativo Lista de Tarefas Se ele pode viver em um monte de
nós diferentes?
10.3.4. Serviços
Um serviço é o conceito abstrato que você usa para acompanhar de onde
vários pods estão em execução. Por exemplo, porque a lista de tarefas
o serviço de aplicativo pode estar em execução em ambos (ou ambos) dos dois nós,
precisa de uma maneira de descobrir para onde ir se você quiser fazer um pedido para
o aplicativo da web. Isso torna um serviço um pouco parecido com uma entrada em um catálogo telefônico,
fornecendo uma camada de abstração entre o nome de alguém e o
local específico onde você pode contatá-los. Porque as coisas podem pular
em torno de um nó para outro, este catálogo telefônico precisa ser atualizado
muitas vezes. Contando com um serviço para acompanhar as várias peças
do seu aplicativo (por exemplo, na Lista de tarefas, você tem o pod
que lida com solicitações da web), você nunca se preocupa com o local onde os podhappens estão sendo executados. O serviço sempre pode ajudar a direcioná-lo para o
lugar certo.
Neste ponto, você deve entender alguns dos principais conceitos de
Kubernetes, mas apenas em um sentido abstrato. Você deve entender isso
um serviço é uma maneira de ajudar a direcioná-lo para o pod direito e que um pod é um
grupo de recipientes com uma finalidade específica, mas eu não disse nada em
tudo sobre como criar um cluster ou um pod ou um serviço. Isso está ok! eu vou

cuidar de algumas dessas coisas mais tarde. Enquanto isso, cheguei ao ponto onde posso explicar exatamente o que é o Kubernetes Engine. Tudo isso falar sobre containers e Kubernetes e pods finalmente valeram a pena!

10.4. O QUE É O MOTOR KUBERNETES?

O Kubernetes é um sistema de código aberto, portanto, se você quiser criar clusters e pods e ter pedidos encaminhados para os nós certos, você tem que instalar, executar e gerenciar o sistema Kubernetes você mesmo. Para minimizar esse fardo, você pode usar o Kubernetes Engine, que é hospedado e implantação gerenciada do Kubernetes executado no Google Cloud Plataforma (usando instâncias do Compute Engine sob o capô).

Você ainda usa todas as mesmas ferramentas que faria se estivesse executando Kubernetes você mesmo, mas você pode cuidar do administrativo operações (como a criação de um cluster e os nós dentro dele) usando o API do Kubernetes Engine.

10.5. INTERAÇÃO COM O MOTOR KUBERNETES

Para ver como tudo isso funciona, você pode definir um simples Kubernetes aplicativo e veja como você pode implantá-lo no Kubernetes Engine.10.5.1. Definindo seu aplicativo

Você começará definindo um aplicativo Hello World Node.js simples usando Express.js. Você deve estar familiarizado com o Express, mas se estiver não, nada mais é do que um framework web Node.js. Um simples aplicativo pode ser algo como a listagem a seguir, salva como index.js.

Listagem 10.1. Aplicativo simples Hello World Express

```
const express = require ('express');
const app = express ();
app.get ('/', (req, res) => {
res.send ('Olá, mundo!');
});
app.listen (8080, '0.0.0.0', () => {
console.log ('A aplicação Hello World está escutando na porta 8080.');
});
```

Este aplicativo da web ouvirá solicitações na porta 8080 e sempre responda com o texto “Hello world!” Você também precisa ter certeza de que ter suas dependências do Node.js configuradas corretamente, o que você pode faça com um arquivo package.json como o mostrado na listagem a seguir.

Listagem 10.2. package.json para sua aplicação

```
{
"name": "hellonode",
"main": "index.js",
"dependências": {
"express": "~ 4"
}
}
```

Como você faria para contêiner esta aplicação? Para fazer isso você criaria um Dockerfile, como mostrado na próxima listagem, que será parece um script de start-up para uma VM, mas um pouco estranho. Não se preocupe, você não deveria ser capaz de escrever isso do zero.

Listagem 10.3. Um exemplo de Dockerfile

```
Nó: 8
WORKDIR / usr / src / app
COPY package.json.
RUN npm install
CÓPIA DE . .
EXPOSE 8080
CMD ["node", "index.js"]
```

Vamos ver cada linha da listagem e ver o que ela faz:

1. Esta é a imagem base (nó: 8), que o próprio Node.js fornece. Ele fornece um sistema operacional básico que vem com Nó v8 pré-instalado e pronto para ser usado.

2. Este é o equivalente do cd para mover-se para um trabalho atual diretório, mas também garante que o diretório exista antes movendo-se para ele.

3. O primeiro comando COPY faz exatamente o que você espera, colocando uma cópia de algo do diretório atual no seu máquina no diretório especificado na imagem do Docker.

4. O comando RUN diz ao Docker para executar um determinado comando na imagem do Docker. Nesse caso, ele instala todos os seus dependências (por exemplo, expresso) para que estejam presentes quando você deseja executar seu aplicativo.

5. Use COPY novamente para trazer o restante dos arquivos para o imagem.

6. EXPOSE é o mesmo que abrir um porto para o resto do mundo ter acesso. Nesse caso, seu aplicativo usará porta 8080, então você quer ter certeza de que está disponível.

7. A instrução CMD é o comando padrão que será executado. Em Neste caso, você deseja iniciar um processo Node.js executando seu serviço (que está no index.js).

Agora que você escreveu um Dockerfile, pode fazer sentido testá-lo localmente antes de tentar implantá-lo na nuvem. Vamos dar uma olhada em como fazer isso.

10.5.2. Executando seu contêiner localmente

Antes de poder executar um contêiner em sua própria máquina, você precisará instale o tempo de execução do Docker. O Docker Community Edition é gratuito e você pode instalá-lo para quase todas as plataformas. Por exemplo, há um arquivo de pacote .deb para Ubuntu disponível em http://docker.com/community-edition.

Como você aprendeu anteriormente, o Docker é uma ferramenta que entende como executar
contêineres que você define usando o formato Dockerfile. Uma vez que você tenha
Docker rodando na sua máquina, você pode dizer para rodar o seu
Dockerfile, e você deve ver seu pequeno aplicativo da Web em execução. Testar se você tem o Docker configurado corretamente, execute a execução do docker hello-world, que diz ao Docker para encontrar uma imagem de container
chamado de "hello-world". Docker sabe como encontrar publicamente disponível
imagens, por isso vai baixar a imagem hello-world automaticamente
e depois executá-lo. A saída da execução desta imagem deve parecer
algo assim:

```
$ docker run hello-world
Não é possível encontrar a imagem 'hello-world: latest' localmente
1latest: Puxando da biblioteca / olá-mundo
2
b04784fba78d: Pull complete
Digerir:
sha256:
f3b3b28a45160805bb16542c9531888519430e9e6d6ffc09d72261b0d26ff74f
Status: Baixou a imagem mais recente de hello-world: latest
Olá do Docker!
# ... Mais informações aqui ...
```

1 O Docker percebe que esta imagem não está disponível localmente.
2 Docker vai procurar a imagem do Dockerhub (um lugar que
hospeda imagens).
Para rodar sua imagem, você tem que pegar seu Dockerfile que descreve um recipiente e construí-lo em uma imagem de contêiner. Isso é um pouco como
Compilando código-fonte em um binário executável quando você está escrevendo
código em uma linguagem de compilação como C ++ ou Java. Certifique-se de que o conteúdo
do seu Dockerfile estão em um arquivo chamado Dockerfile; então você vai usar docker build para criar sua imagem e marcá-la como hello-node:

```
$ docker build --tag hello-node.
Enviando o contexto de construção para o daemon do Docker
Passo 1/7: nó FROM: 8
1,345MB
Etapa 2/7: WORKDIR / usr / src / app
Passo 3/7: COPY package.json.
Etapa 4/7: instalação do RUN npm
Etapa 5/7: COPY. .
Etapa 6/7: EXPOSE 8080
Etapa 7/7: nó CMD index.js
Construído com sucesso 358ca555bbf4
```

Com êxito marcado hello-node: latest
Você vai ver muita coisa acontecendo sob o capô, e ele vai se alinhar um para um
com os comandos que você definiu no Dockerfile. Primeiro, vai
procurando o contêiner base publicamente disponível que tenha o Nó v8
instalado e, em seguida, ele irá configurar o seu diretório de trabalho, todo o caminho
para executar o arquivo index.js que define seu aplicativo da web. Note que isso é apenas construir o container, não rodá-lo, então o container
em si está em um estado pronto para ser executado, mas não está em execução no momento.
Se você quiser testar que as coisas funcionaram como esperado, você pode usar o
o comando docker run com algumas flags especiais:
$ docker run -d -p 8080: 8080 hello-node
Aqui, o sinalizador -d informa ao Docker para executar essa imagem de contêiner no
background, e o -p 8080: 8080 diz ao Docker para pegar qualquer coisa
na sua máquina que tenta falar com a porta 8080 e enviá-lo para
a porta do seu contêiner 8080.
A linha a seguir mostra o resultado da execução da imagem do contêiner:
485c84d0f25f882107257896c2d97172e1d8e0e3cb32cf38a36aee6b5b86a469
Este é um ID exclusivo que você pode usar para endereçar essa imagem em particular
(afinal, você pode ter muitas imagens rodando na mesma
Tempo).
Para verificar se sua imagem está em execução, use o docker ps
comando, e você deve ver a imagem de nó de hello na lista:
$ docker ps --format "table {{.ID}} \ t {{. Imagem}} \ t {{. Status}}"
ID DO CONTENTOR
IMAGEM
STATUS
485c84d0f25f
ola-nó
Cerca de um minuto
Como você pode ver, o contêiner está usando a imagem de nó de hello e
está funcionando há apenas um minuto. Observe também que o contêiner
O ID foi reduzido para as primeiras letras do ID exclusivo a partir do comando docker run. Você pode encurtar isso mesmo
além disso, contanto que o ID não corresponda a mais de um contêiner,
para este exercício, vou me referir a este contêiner como 485c. Você disse ao Node para
imprimir para o console quando ele começou a ouvir solicitações.
Você pode verificar a saída do seu contêiner até o momento inserindo esta linha:
$ docker logs 485c

A saída aqui é exatamente o que você esperaria:
Olá app mundo está escutando na porta 8080.
Agora tente conectar-se ao servidor Node.js do contêiner usando curl:
$ curl localhost: 8080
Você deveria ver isto:
Olá Mundo!
Como mágica, você tem um processo Node.js rodando e servindo HTTP solicitações de dentro de um contêiner sendo executado pelo serviço Docker em sua máquina. Se você quisesse parar este contêiner, você poderia usar o comando de parada do docker:
$ docker stop 485c
485c
$ docker ps --format "table {{.ID}}"
CONTAINER ID.Aqui, quando você parar o container docker, ele não aparecerá mais no lista de contêineres em execução mostrados usando o docker ps.
Agora que você tem uma ideia de como é correr o seu simples aplicação como um contêiner usando Docker, vamos ver como você poderia mudar de usar sua instância local do Docker para um completo-fledged Kubernetes cluster (que usa o Docker sob o capô). Vou começar com como você empacota seu aplicativo em container e o implementa para o seu registro de contêiner particular.
10.5.3. Implantando no seu registro de contêiner
Neste ponto, você construiu e executou um contêiner localmente, mas se quiser Para implantá-lo, você precisará que ele exista no Google Cloud. Você precisa faça o upload do seu contêiner para executá-lo no Kubernetes Engine. Para permitir que você fazer isso, o Google oferece um registro de contêiner privado por projeto que atua como armazenamento para todos os seus vários contêineres.
Para começar, primeiro você precisa marcar sua imagem em um formato especial. Em No caso do registro de contêiner do Google, o formato da tag é gcr.io/your-project-id/your-app (que pode vir com diferentes versões no final, como: v1 ou: v2). Neste caso, você precisa para marcar sua imagem de contêiner como gcr.io/your-project-id / hello-node: v1. Para fazer isso, você usará a tag docker comando. Como você vai lembrar, você chamou a imagem que você criou nó, e você sempre pode verificar novamente a lista de imagens usando o comando docker images:
$ docker images --format "table {{.Repository}} \ t {{ID}}"
REPOSITÓRIO
ola-nó
ID IMAGEM
96001025c6a9Re-tag sua imagem de Docker de nó de hello:

$ docker tag hello-node gcr.io/project-id/hello-node:v1
Depois de marcar novamente a imagem, você verá uma imagem extra na lista de imagens disponíveis do Docker. Observe também que o: v1 parte do seu nome aparece sob o cabeçalho TAG especial no seguindo o snippet, facilitando a visualização quando você tem vários versões do mesmo container:
$ docker images --format "table {{.Repository}} \ t {{. Tag}}"
REPOSITÓRIO
TAG
gcr.io/project-id/hello-node
ola-nó
v1
Mais recentes
Você sempre pode criar seu contêiner com esse nome desde o começo mas você já tinha construído esse contêiner de antemão.
Agora tudo o que resta é fazer o upload da imagem do contêiner para o seu contêiner
registro, o que você pode fazer com a ferramenta de linha de comando gcloud:
$ gcloud docker - push gcr.io/project-id/hello-node:v1
O push se refere a um repositório [gcr.io/project-id/hello-node]
b3c1e166568b: empurrado
7da58ae04482: empurrado
2398c5e9fe90: empurrado
e677efb47ea8: Empurrado
aaccb8d23649: empurrado
348e32b251ef: empurrado
e6695624484e: Empurrado
da59b99bbd3b: empurrado
5616a6292c16: empurrado
f3ed6cb59ab0: empurrado
654f45ecb7e3: empurrado
2c40c66f7667: empurrado
v1: digest:
sha256:
65237913e562b938051b007b8cbc20799987d9d6c7af56461884217ea047665a
tamanho: 2840
Você pode verificar se isso funcionou indo para o Cloud Console e escolhendo o Container Registry na navegação do lado esquerdo. Uma vez lá, você deve ver seu contêiner de nó de hello na listagem e clicar nele deve mostrar o início do hash e a tag v1 que você aplicou (figura 10.9).
Figura 10.9. Registro de Container listando seu contêiner de nó de hello
Você fez o upload do seu contêiner para o Google Cloud. Agora você pode obter seu cluster do Kubernetes Engine pronto.
10.5.4. Configurando seu cluster do Kubernetes Engine

Semelhante a como você precisava instalar o Docker em uma máquina local para executar
um container, você precisará configurar um cluster do Kubernetes se quiser implante seus contêineres no Kubernetes Engine. Felizmente, isso é muito mais fácil do que parece, e você pode fazer isso no Cloud Console,
como se você ligasse uma VM do Compute Engine. Para começar, escolha Kubernetes
Mecanismo na navegação do lado esquerdo do Cloud Console. Uma vez lá, você verá um aviso para criar um novo cluster do Kubernetes. Quando você clica sobre isso, você verá uma página que deve ser semelhante a um para criar uma nova VM do Compute Engine (figura 10.10).
Figura 10.10. Prompt para criar um novo cluster do Kubernetes Engine. Porque você está apenas tentando chutar os pneus do Kubernetes Engine, você pode deixar tudo definido para os padrões. Você vai usar o
zona central1-a, uma única vCPU por máquina e um tamanho de três VMs para todo o cluster. (Lembre-se, você sempre pode mudar estes coisas mais tarde.) A única coisa que você deve fazer é escolher um nome para o seu
cluster neste exemplo, como o primeiro cluster. Depois de verificar que o formulário mostra o que você espera, clique no botão Criar e, em seguida, aguarde alguns segundos enquanto o Google Kubernetes Engine (GKE), na verdade,
cria as VMs e configura o Kubernetes no novo cluster de máquinas.
Depois de ter seu cluster criado e marcado como em execução, você pode Verifique se está funcionando corretamente listando suas VMs. Lembre-se disso um cluster GKE depende das VMs do Compute Engine sob o capô, então você pode olhar para eles como qualquer outra VM em execução:

```
$ gcloud compute instances list --filter "zona: us-central1 -a nome: gke-
* "|
awk '{print $ 1}'
NOME
gke-primeiro-cluster-default-pool-e1076aa6-c773
gke-primeiro-cluster-default-pool-e1076aa6-mdcd
gke-primeiro-cluster-padrão-pool-e1076aa6-xhxp
```

Você tem um cluster em execução e pode ver que três VMs que compõem
o cluster está em execução. Agora vamos investigar como interagir com o grupo.

10.5.5. Implantando seu aplicativo

Depois de implantar seu contêiner e criar seu cluster, o
A próxima coisa que você precisa fazer é encontrar uma maneira de se comunicar e
implante coisas no seu cluster. Afinal, você tem um monte de máquinas correndo sem fazer nada! Porque este cluster é composto de máquinas executando o Kubernetes sob o capô, você pode usar as ferramentas existentes para

conversando com o Kubernetes para falar com seu cluster do Kubernetes Engine. Nisso
Nesse caso, a ferramenta que você usará para falar com o seu cluster é chamada de kubectl.
Nota
Tenha em mente que algumas das operações que você executará usando o kubectl
vai sempre voltar rapidamente, mas eles provavelmente estão fazendo algum fundo
trabalhe sob o capô. Como resultado, você pode ter que esperar um pouco
antes de passar para o próximo passo.
Caso você não esteja familiarizado com o Kubernetes (o que é esperado),
Para facilitar esse processo, o Google Cloud oferece uma instalação rápida de kubectl usando a ferramenta de linha de comando gcloud. Tudo que você tem a fazer
para instalar o kubectl é executado um simples comando gcloud:
Componentes do $ gcloud instalam o kubectl
ObservaçãoSe você instalou o gcloud usando um gerenciador de pacotes (como o apt-get
para o Ubuntu), você pode ver uma recomendação do gcloud dizendo
para usar o mesmo gerenciador de pacotes para instalar o kubectl (por exemplo, apt-get install kubectl).
Depois de ter o kubectl instalado, você precisa ter certeza de que é
devidamente autenticado para falar com seu cluster. Você pode fazer isso usando
outro comando gcloud que busca as credenciais certas e
garante que o kubectl os tenha disponível:
clusters de contêineres do $ gcloud get-credentials --zone us-central1-a first-grupo
Buscando dados de ponto de extremidade e autenticação do cluster.
Entrada kubeconfig gerada para o primeiro cluster.
Depois de configurar o kubectl, você pode usá-lo para implantar um novo
aplicação usando sua imagem de contêiner sob o capô. Você pode fazer
isso executando kubectl run e usando kubectl get pods para
verifique se a ferramenta implantou seu aplicativo em um pod:
$ kubectl run hello-node --image = gcr.io / seu-projeto-id-aqui / olá-nó: v1 -
porta 8080
implantação "hello-node" criada
$ kubectl obtém pods
NOME
PRONTO
ola-node-1884625109-sjq76
1/1
STATUS
Corrida

RESTAURANTES
0
ERA
55s
Você está quase pronto, com um último passo antes de poder verificar se as coisas estão funcionando como esperado. Lembre-se que EXPOSE Comando 8080 no seu Dockerfile? Você tem que fazer alguma coisa semelhante ao seu cluster para garantir que as portas que você precisa para handlequests sejam expostas corretamente. Para fazer isso, você pode usar o kubectl expor comando:

```
$ kubectl expor implantação hello-node --type = LoadBalancer --port 8080
serviço "hello-node" exposto
```

Sob o capô, o Kubernetes Engine configurará um balanceador de carga como você aprendeu no capítulo 9. Uma vez feito isso, você deve ver um balanceador de carga aparecem no Cloud Console que aponta para seus três Instâncias de VM que compõem o cluster (figura 10.11).

Figura 10.11. Balanceador de carga criado automaticamente no Cloud Console

Neste ponto, você pode estar pensando que os pods são o caminho que você mantém recipientes juntos para servir um propósito comum, e não algo que você falaria individualmente. E você está certo! Se você quiser falar com sua aplicação, você tem que usar a própria abstração para isso, o que é conhecido como um serviço.

Você pode olhar para os serviços disponíveis (neste caso, sua aplicação) usando o kubectl get services:

```
$ kubectl obter serviceNAME CLUSTER-IP PORTA EXTERNAL-IP PORTA (S)
nó de hello 10.23.245.188 104.154.231.30 8080: 31201 / TCP 1m
kubernetes 10.23.240.1 <nenhum> 443 / TCP 10m
```

Observe neste momento que você tem um serviço generalizado do Kubernetes (que lida com a administração), bem como o serviço para o seu aplicação. Além disso, seu aplicativo possui um endereço IP externo que você pode usar para ver se tudo funcionou, fazendo uma simples pedido ao serviço:

```
$ onda 104.154.231.30:8080
Olá Mundo!
```

E com certeza, tudo funcionou exatamente como esperado. Você agora ter um aplicativo de contêineres em execução usando um pod e um serviço dentro do Kubernetes, gerenciado pelo Kubernetes Engine. Isso por si só é bem legal, mas a verdadeira mágica acontece quando você precisa lidar mais tráfego, o que você pode fazer replicando o aplicativo.

10.5.6. Replicando seu aplicativo

Lembre-se de que, usando o Compute Engine, você pode facilmente ativar novas VMs alterando o tamanho do cluster, mas para obter o seu aplicativo em execução

nessas máquinas, você precisava configurá-las para serem executadas automaticamente quando
a VM foi ligada ou você teve que se conectar manualmente à máquina e iniciar o aplicativo. Que tal fazer isso com o Kubernetes? Neste ponto em sua implantação, você tem um cluster Kubernetes de três nós, com dois serviços (um para o próprio Kubernetes e um para o seu aplicativo) e seu aplicativo está sendo executado em um único pod. Vamos olha como você pode mudar isso, mas primeiro, vamos comparar seu cluster pode manipular solicitações na configuração atual. Você pode usar qualquer ferramenta de benchmarking que desejar, mas para esta ilustração, tente usar o Apache Bench (ab). Se você não tiver essa ferramenta instalada, pode instalá-lo no Ubuntu executando o sudo apt-get install apache2-utils. Para testar isso, você enviará 50.000 solicitações, 1.000 no uma vez, para o seu aplicativo, e ver o quão bem o cluster faz com lidando com os pedidos:
$ ab -c 1000 -n 50000 -qSd http://104.154.231.30:8080/
Este é o ApacheBench, Versão 2.3 <$ Revisão: 1604373 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licenciado para a Apache Software Foundation, http://www.apache.org/
Benchmarking 104.154.231.30 (seja paciente) ..... feito
 # ...
Nível de Concorrência:
Solicitações por segundo:
Tempo por pedido:
1000
2980,93 [# / seg] (média)
335,465 [ms] (média)
1
2
 # ...
1 O cluster processou cerca de 3.000 solicitações por segundo.
2 Concluiu a maioria dos pedidos em cerca de 300 milissegundos.
E se você pudesse dimensionar seu aplicativo para aproveitar mais do seu cluster? Acontece que você pode fazer isso com um comando: escala kubectl. Veja como você dimensiona seu aplicativo para ser executado 10 pods ao mesmo tempo:
Implementação em escala $ kubectl hello-node --replicas = 10
implantação "hello-node" escalonado
Imediatamente depois de executar este comando, olhando para os pods disponível mostrará que você está indo de 1 disponível até 10 diferentes pods:
$ kubectl obtém pods
NOME
ERA
PRONTOS PARA READY STATUS
ola-node-1884625109-8ltzb

3m
ola-node-1884625109-czn7q
3m 1/1 ContainerCreating 0
1/1 ContainerCreating 0
ola-node-1884625109-dzs1d
3m
ola-node-1884625109-gw6rz
3m 1/1 ContainerCreating 0
1/1 ContainerCreating 0
ola-node-1884625109-kvh9v
3m
ola-node-1884625109-ng2bh
3m 1/1 ContainerCreating 0
1/1 ContainerCreating 0
ola-nó-1884625109-q4wm2
3m
ola-node-1884625109-r5msp
3m 1/1 ContainerCreating 0
1/1 ContainerCreating 0
ola-node-1884625109-sjq76
1h
ola-node-1884625109-tc2lr
3m 1/1 em execução 0
1/1 ContainerCreating 0
Depois de alguns minutos, esses pods devem aparecer e estar disponíveis bem:
$ kubectl obtém pods
NOME
ola-node-1884625109-8ltzb PRONTO
1/1 STATUS
Correndo RESTARTS
0 AGE
3m
ola-node-1884625109-czn7q
ola-node-1884625109-dzs1d
ola-node-1884625109-gw6rz
ola-node-1884625109-kvh9v 1/1
1/1
1/1
1/1 corrida
Corrida
Corrida
Correndo 0
0
0
0 3m

3m
3m
3m
ola-node-1884625109-ng2bh
ola-nó-1884625109-q4wm2
ola-node-1884625109-r5msp
ola-node-1884625109-sjq76 1/1
1/1
1/1
1/1 corrida
Corrida
Corrida
Correndo 0
0
0
0 3m
3m
3m
1h
olá-nó-1884625109-tc2lr 1/1 Corrida 0 3m
Neste ponto, você tem 10 pods em execução nos seus três nós, então tente seu benchmark uma segunda vez e veja se o desempenho é qualquer
Melhor:
$ ab -c 1000 -n 50000 -qSd http://104.154.231.30:8080/
Este é o ApacheBench, Versão 2.3 <$ Revisão: 1604373 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licenciado para a Apache Software Foundation, http://www.apache.org/
Benchmarking 104.154.231.30 (seja paciente) ..... feito
 # ...
Nível de Concorrência:
Solicitações por segundo: 1000
5131,86 [# / s] (média) 1
Tempo por pedido: 194.861 [ms] (média) 2
 # ...
1 Seu cluster recém-ampliado gerenciou cerca de 5.000 solicitações por segundo.
2 Concluiu a maioria dos pedidos em cerca de 200 milissegundos.
Neste ponto, você pode estar se perguntando se existe uma interface do usuário para interagir com
toda essa informação. Especificamente, existe uma interface do usuário para ver os pods em
da mesma forma que há um para analisar as instâncias do GCE? Há sim,
mas faz parte do próprio Kubernetes, não do Kubernetes Engine.
10.5.7. Usando a interface do usuário do Kubernetes

O Kubernetes vem com uma interface do usuário embutida e porque o Kubernetes Engine
é apenas um cluster gerenciado do Kubernetes, você pode ver a interface do usuário do Kubernetes
para o seu cluster do Kubernetes Engine da mesma maneira que faria com qualquer outro
Implantação do Kubernetes. Para fazer isso, você pode usar o kubectl
ferramenta de linha de comando para abrir um túnel entre sua máquina local
e o mestre do Kubernetes (figura 10.12). Isso permitirá que você fale para, digamos, http: // localhost: 8001, e um proxy local roteará com segurança
sua solicitação para o mestre do Kubernetes (ao invés de um servidor em seu máquina local):

```
proxy $ kubectl
Começando a servir em 127.0.0.1:8001
```

Figura 10.12. Solicitações locais de proxies para o mestre do Kubernetes
Quando o proxy estiver em execução, conectando-se a http: // localhost: 8001 / ui /
mostra a interface do usuário completa do Kubernetes, que fornece muitas recursos de gerenciamento para seu cluster (figura 10.13).
Figura 10.13. Interface do usuário do Kubernetes usando o proxy kubectl Agora você já viu como o Kubernetes funciona em um nível simplificado. o
parte que é importante lembrar é que você não tem que configurar
ou instale o Kubernetes no seu cluster porque o Kubernetes Engine
fez tudo por você. Como mencionei antes, o Kubernetes é um sistema enorme, então este capítulo não é sobre ensinar-lhe tudo o que há para saber
sobre isso. Por exemplo, você pode ver como acessar a interface do usuário do Kubernetes,
mas não vou entrar em detalhes sobre o que você pode fazer usando a interface do usuário.
Em vez disso, o objetivo deste capítulo é mostrar como o Kubernetes funciona quando você confia no Kubernetes Engine para lidar com todos os
trabalho administrativo.
Se você estiver interessado em fazer coisas mais avançadas com o Kubernetes, como a implantação de um cluster mais avançado, composto de muitos pods
e bancos de dados, agora é a hora de pegar um livro sobre isso, porque
O Kubernetes Engine nada mais é do que um Kubernetes gerenciado
implantação sob o capô. Dito isto, algumas coisas são específicas
para o Kubernetes Engine e não geral em todo o Kubernetes, portanto, vamos ver brevemente como você pode gerenciar o cluster do Kubernetes subjacente usando o Kubernetes Engine e a cadeia de ferramentas do Google Cloud.

10.6. MANUTENÇÃO DO SEU CLUSTER

Novas versões do software são lançadas e, às vezes, faz sentido
atualizar. Por exemplo, se o Apache lançar novas correções de bugs ou segurança
correções, faz muito sentido atualizar para a versão mais recente.
O mesmo vale para o Kubernetes, mas lembre-se, porque você está usando

Kubernetes Engine, em vez de implantar e gerenciar seus próprios
Kubernetes cluster, você precisa de uma maneira de gerenciar esse Kubernetes cluster através do Kubernetes Engine. Como você pode imaginar, isso é bem fácil.
Vamos começar com a atualização da versão do Kubernetes.
Seu cluster do Kubernetes tem duas partes distintas que o Kubernetes
O mecanismo gerencia: o nó principal, que é totalmente oculto (não listado na lista de nós) e os nós do cluster. Os nós do cluster são os
aqueles que você vê ao listar os nós ativos no cluster. Se o Kubernetes tem uma nova versão disponível, você terá a capacidade de atualizar o nó mestre, todos os nós do cluster ou ambos. Embora a atualização
processo é semelhante para ambos os tipos de nós, você terá coisas diferentes para se preocupar com cada tipo, então vamos olhar para eles separadamente, começando
com o nó mestre do Kubernetes.

10.6.1. Atualizando o nó mestre do Kubernetes

Por padrão, como parte do gerenciamento do Google, os nós principais são atualizado automaticamente para a última versão suportada do Kubernetes depois que ele é lançado, mas se você quiser pular para o último suporte versão do Kubernetes, você pode optar por atualizar manualmente
nó mestre do cluster antes do previsto. Quando uma atualização estiver disponível para o Kubernetes, seu cluster do Kubernetes Engine mostrará um link ao lado
para o número da versão que você pode clicar para alterar a versão. Para exemplo, a figura 10.14 mostra o link exibido quando uma atualização é disponível para o seu nó mestre.

Figura 10.14. Quando uma atualização para o Kubernetes está disponível no Kubernetes Engine

Ao clicar no link Atualizar, você verá um aviso que permite
para escolher uma nova versão do Kubernetes. Como as notas de aviso, você precisa
para manter algumas coisas em mente ao alterar a versão do Kubernetes (figura 10.15).

Figura 10.15. Avisar e avisar para atualizar seu nó mestre do Kubernetes First, atualizando de uma versão mais antiga para uma nova versão no

O nó principal do cluster do Kubernetes Engine é uma operação unidirecional. E se
você decide mais tarde que não gosta da nova versão do Kubernetes (talvez haja um bug que ninguém notou ou uma suposição que não espera mais), você não pode usar esse mesmo processo para voltar versão anterior. Em vez disso, você teria que criar um novo cluster com o versão antiga do Kubernetes e reimplemente seus contêineres para essa outra grupo. Para se proteger contra problemas de atualização e evitar
tempo de inatividade, geralmente é uma boa ideia testar um cluster separado com o
nova versão para ver se tudo funciona como seria de esperar. Depois de ter

testamos a versão mais recente e descobrimos que ela funciona como você esperava,
Deve ser seguro atualizar seu cluster existente.
Em seguida, alterar a versão do Kubernetes exige que você pare,
atualizar e reiniciar o plano de controle do Kubernetes (o serviço que
o kubectl fala quando precisa escalar ou implementar novos pods). Enquanto
Se a operação de atualização estiver em execução, você não poderá editar seu cluster e todas as chamadas do kubectl que tentarem conversar com o cluster não funcionarão. E se
de repente você recebe um pico de tráfego no meio da atualização,
você não poderá executar a escala kubectl, o que poderia resultar em
tempo de inatividade para alguns de seus clientes.
Finalmente, não se esqueça de que a atualização manual é uma etapa opcional. E se
você espera um pouco, seu nó mestre do Kubernetes
atualize automaticamente para a versão mais recente sem que você perceba. Mas
esse não é o caso para seus nós de cluster, então vamos olhar para aqueles em mais
detalhe.

10.6.2. Atualizando Nós do Cluster

Ao contrário do nó principal, os nós do cluster não estão ocultos no
sombras. Em vez disso, eles são visíveis para você como um Compute Engine comum
VMs semelhantes a grupos de instâncias gerenciadas. Além disso, ao contrário do mestre
nó, a versão do Kubernetes em execução nessas VMs gerenciadas
não é atualizado automaticamente de vez em quando. Cabe a você decidir
quando fazer essa mudança. Você pode mudar a versão do Kubernetes
nos nós do cluster, procurando no Cloud Console ao lado do
Seção Versão do Nó do seu cluster e clicando no link Alterar
(figura 10.16).
Figura 10.16. Área do Cloud Console para alterar a versão dos nós do clusterVocê pode estar se perguntando por que estou falando sobre a alteração do nó
versão em vez de atualizar. A razão é principalmente porque ao contrário
com a versão do nó mestre, esta operação é às vezes reversível
(embora nem sempre). Você pode fazer downgrade para 1.5.7 e então decidir
atualizar de volta para 1.6.4. Ao clicar no link Alterar, você verá um
prompt que permite que você escolha a versão de destino e explica bastante
um pouco sobre o que está acontecendo sob o capô (figura 10.17).
Figura 10.17. Avisar para alterar a versão dos nós do clusterPrimeiro, porque há sempre pelo menos um nó do cluster (diferente do
nó mestre, que é sempre uma única instância), você altera o
Versão do Kubernetes nos nós do cluster, aplicando uma atualização contínua
ao seu cluster, ou seja, as máquinas são modificadas uma de cada vez até

todos eles estão prontos. Para fazer isso, o Kubernetes Engine fará primeiro o nó não programável. (Nenhum novo pods será agendado no nó.) Ele vai então drenar todos os pods no nó (terminá-los e, se necessário, coloque-os em outro nó). Quanto menos nós você tiver, o É mais provável que você tenha algum tempo de inatividade. Para Por exemplo, se você tiver um cluster de nó único, seu serviço será indisponível durante o período de inatividade: 100% dos seus nós vai cair em algum momento. Por outro lado, se você tiver um nó de 10 cluster, você estará abaixo de 10% de capacidade no máximo (1 dos 10 nós no uma única instância). Em segundo lugar, observe que as opções disponíveis neste prompt (figura 10.17) não são iguais aos do prompt para atualizar o nó mestre (figura 10.15). A lista é limitada dessa maneira porque os nós do cluster deve ser compatível com o nó mestre, o que significa não muito longe por trás dele (e nunca à frente dele). Se você tem um nó mestre na versão 1.6.7, você pode usar a versão 1.6.4 nos nós do cluster, mas se o seu nó mestre usa uma versão posterior, essa mesma versão do nó do cluster fique muito para trás. Como resultado, é uma boa ideia atualizar seu cluster nós a cada três meses ou mais.

Em terceiro lugar, ao contrário do nó mestre, que está escondido da sua visão, você pode esperar que quaisquer dados armazenados nos nós do cluster sejam esteja lá para sempre. Na verdade, a menos que você configure explicitamente persistentes armazenamento para sua instância do Kubernetes, os dados armazenados serão perdido quando você executa uma atualização. Os discos de inicialização para cada cluster nó são excluídos e novos são criados para os novos nós. Qualquer outro discos nonboot (e discos não locais) serão mantidos. Você pode ler mais sobre como conectar o armazenamento persistente do Google Cloud no Documentação do Kubernetes (ou um dos muitos livros sobre o Kubernetes que estão disponíveis). Procure uma seção sobre volumes de armazenamento e tipo de volume gcePersistentDisk.

Quarto e, finalmente, similarmente ao upgrade do nó mestre, enquanto o alteração de versão nos nós do cluster está em andamento, você não poderá edite o cluster em si. Além do tempo de inatividade, você pode experiência por causa dos nós sendo drenados de seus pods, o controle operações de avião estarão indisponíveis durante a duração da versão mudança.

10.6.3. Redimensionando seu clusterAs com o aumento do número de pods usando a escala kubectl,

Mudar o número de nós no seu cluster é fácil. Na nuvem Console, se você clicar em Editar no seu cluster, verá um campo chamado Tamanho, que você definiu originalmente como três quando criou o cluster. A alteração deste número aumentará o número de nós disponíveis em seu cluster, e você pode definir o tamanho para um número maior, que

irá adicionar mais nós para fornecer mais capacidade, ou para um número menor,
que encolherá o tamanho do seu cluster. Se você encolher o cluster,
de forma semelhante a uma alteração de versão nos nós do cluster, o Kubernetes Engine
primeiro marcará um nó como não programável, depois drenará todos os conjuntos e, em seguida,
desligue isso. Como exemplo, a figura 10.18 mostra como é
altere seu cluster de três nós para seis.
Figura 10.18. Redimensionando seu cluster para seis nós
Você também pode fazer isso usando a ferramenta de linha de comando gcloud. Para
Por exemplo, o snippet a seguir redimensiona o cluster de seis nós de volta para três:

```
Os clusters de contêiners do $ gcloud redimensionam o primeiro cluster --zone us-central1-a -
-size = 3
Pool [default-pool] para [primeiro cluster] será redimensionado para 3.
Você quer continuar (S / n)?
YResizing primeiro cluster ... feito.
Atualizado [https://container.googleapis.com/v1/projects/your-project-id-Aqui/
zonas / us-central1-a / clusters / primeiro-cluster].
```

Porque estamos prestes a sair da manutenção, você pode querer
gire seu cluster do Kubernetes. Você pode fazer isso excluindo
cluster, no Cloud Console ou usando a ferramenta de linha de comando.
Certifique-se de mover quaisquer dados que possa precisar para um local seguro antes
excluindo o cluster. Com isso, é hora de pensar em como
trabalhos de precificação.

10.7. ENTENDENDO OS PREÇOS

Como acontece com alguns dos outros serviços no Google Cloud Platform,
O Kubernetes Engine depende do Compute Engine para fornecer
infra-estrutura para o cluster Kubernetes. Como resultado, o custo do
O próprio cluster é baseado principalmente nos nós do cluster. Porque estes são
simplesmente VMs do Compute Engine, você pode consultar o capítulo 9 para
informações sobre quanto cada nó custa. Além do custo de
os próprios nós, lembre-se do nó mestre do Kubernetes, que é
totalmente escondido de você pelo Kubernetes Engine. Porque você não
ter controle sobre esse nó explicitamente, não há custo para sobrecarga
nele.

10.8. QUANDO UTILIZAR O MOTOR KUBERNETES?

Você pode estar se perguntando especificamente como o Kubernetes Engine se destaca
contra outros ambientes de computação, principalmente o Compute Engine.

Vamos usar o scorecard padrão para computação para ver como o Kubernetes
O motor compara-se aos outros (figura 10.19) .Figura 10.19. Scorecard do Kubernetes Engine

10.8.1. Flexibilidade

Semelhante ao Compute Engine, o Kubernetes Engine é bastante flexível, mas
não é o mesmo que ter um conjunto de utilidades genéricas de VMs que são executadas
o que você quiser. Por exemplo, você é obrigado a especificar o seu
ambiente usando imagens de contêiner (com Dockerfiles), em vez de
scripts de inicialização personalizados ou imagens de disco do GCE. Embora isso seja
tecnicamente uma limitação que reduz a flexibilidade, não é uma coisa ruim
formalize como você define sua aplicação em termos de um contêiner.
Embora o Kubernetes Engine seja um pouco mais restritivo, isso pode
uma coisa boa.
O Kubernetes Engine também tem outras limitações, como a
requisito de que a versão do Kubernetes dos nós do cluster seja compatível
com a versão do seu nó principal, ou o fato de perder os dados do disco de inicialização quando você atualiza seus nós. Mais uma vez, embora
as coisas são tecnicamente restrições, você não deveria considerá-las
disjuntores. Para a maioria dos cenários, o Kubernetes Engine não é menos flexível
que o Compute Engine, e oferece alguns benefícios, como o
capacidade de escalar os nós e pods para cima e para baixo. Como resultado, se você
olhe além da exigência de que você defina seu aplicativo usando
contêineres, o Kubernetes Engine está praticamente isento de restrições
você compara com o Compute Engine. A grande diferença vem é
quando você começa a falar sobre complexidade.

10.8.2. Complexidade

Como você viu, os ambientes de computação podem ser complicados e
O Kubernetes Engine (que se baseia no Kubernetes sob o capô) não é
diferente. Tem uma grande capacidade de complexidade, mas beneficiando
essa complexidade envolve altos custos iniciais de aprendizagem. Similarmente,
embora um carro seja muito mais complexo do que uma bicicleta, uma vez que você aprende
como dirigir o carro, os benefícios se tornam claros.
Porque eu apenas arranhei a superfície do que o Kubernetes é capaz
de, você pode não ter uma compreensão completa de quão complexo é o sistema
como um todo pode ser - é muito mais complicado do que "ligar uma VM".
Colocando isso em contexto realista, se você quiser implantar um
aplicação com um único nó que nunca precisaria crescer além
Nesse nó, o Kubernetes Engine é provavelmente um exagero. Se, por outro lado,

você queria implantar um grande cluster de servidores de API para lidar com grandes picos de tráfego, provavelmente valeria a pena o esforço para entender Kubernetes e talvez confie no Kubernetes Engine para gerenciar seu Kubernetes cluster.

10.8.3. Desempenho Ao contrário do uso de VMs brutas, como o Compute Engine, o Kubernetes tem algumas camadas de abstração entre o código do aplicativo e o real hardware executando esse código. Como resultado, o desempenho geral não pode ser tão bom quanto uma VM simples, e certamente não é tão bom quanto sistema não virtualizado. O desempenho do Kubernetes Engine não será tão eficiente como algo como o Compute Engine, mas a eficiência não é tudo. A escalabilidade é outro aspecto do desempenho que pode ter um efeito real.

Embora você possa precisar de mais nós em seu cluster para obter o mesmo desempenho como o uso de hardware não virtualizado, você pode alterar a capacidade de desempenho geral do seu sistema com Kubernetes Engine do que você pode com o Compute Engine ou máquinas não virtualizadas. Como resultado, se você conhece seu desempenho requisitos exatamente, e você tem certeza que eles vão ficar exatamente o mesmo com o tempo, o uso do Kubernetes Engine lhe forneceria escalabilidade que você não precisa. Por outro lado, se você não tiver certeza quanta energia você precisa e quer a capacidade de mudar sua mente sempre que você quiser, o Kubernetes Engine torna isso fácil, com uma ligeira redução da eficiência global.

Porque esta diferença de eficiência é tão pequena, deve ser apenas uma problema quando você tem uma implantação enorme de centenas de máquinas (onde as pequenas diferenças se somam para se tornarem diferenças). Se o seu sistema é relativamente pequeno, você não deveria observe as diferenças de eficiência.

10.8.4. Custo

O Kubernetes Engine não é mais caro do que o Compute Engine bruto VMs que alimentam o cluster do Kubernetes subjacente. Além disso, não cobra pelo gerenciamento de cluster usando um nó mestre. Como um Como resultado, usá-lo é realmente mais barato do que rodar o seu próprio Kubernetes Cluster usando VMs do Compute Engine sob o capô.

10.8.5. No geral

Como você escolhe entre o mecanismo de computador e o Kubernetes? Engine, dado que ambos são flexíveis, funcionam de forma semelhante e são com preço semelhante, mas o uso do Kubernetes Engine requer que você aprenda e entender o Kubernetes, que é bastante complexo? Embora isso seja verdade, o fator diferencial tende a ser o tamanho do seu sistema será e quanto você quer sua implantação

configuração a ser representada como código. Os benefícios de usar
O Kubernetes Engine sobre outras plataformas de computação não é sobre o
custo ou a infra-estrutura, mas sobre os benefícios do Kubernetes como um caminho
de manter seu procedimento de implantação claro e bem documentado.
Como resultado, a regra geral é usar o Kubernetes Engine quando você
ter um sistema grande que você (e sua equipe) precisarão manter
por um longo período de tempo. Por outro lado, se você precisar de algumas VMs
para fazer alguns cálculos e planejar desativá-los depois de um tempo,
confiar no Compute Engine pode ser mais fácil. Para tornar isso mais
concreto, vamos percorrer os três aplicativos de exemplo que eu
discutido e ver o que faz mais sentido implantar usando
Kubernetes Engine.

10.8.6. Lista de afazeres

O exemplo de aplicativo de lista de tarefas é uma ferramenta simples para rastrear as listas de afazeres
e se eles estão prontos ou não. Como resultado, é improvável que precise
ampliar por causa de quantidades extremas de carga. Como resultado, o KubernetesEngine é provavelmente um pouco exagerado para suas necessidades (tabela 10.1).

Tabela 10.1. Necessidades de computação de aplicativos To-Do-List

Aspecto	Flexibilidade	Complexidade	atuação	Custo	Necessidades
Nem tanto assim	Mais simples é melhor.	Baixa a moderada	Menor é melhor.	Bom encaixe?	Exagero
Não tão bom	Um pouco exagerado durante o tempo não pontual	Não é o ideal, mas não é horrível			

No geral, o aplicativo To-Do-List, embora possa ser executado no Kubernetes, é provavelmente não vai fazer uso de todos os recursos e vai exigir uma
pouco mais de aprendizado do que é desejável para tal aplicação. Como um resultado,
algo mais simples, como uma única VM do Compute Engine, pode ser
melhor escolha.

10.8.7. E * Exchange

E * Exchange, a plataforma de negociação de ações on-line (tabela 10.2), tem muitos

recursos mais complexos, e você pode dividir cada um deles em muitos diferentes categorias. Por exemplo, você pode ter um servidor de API que manipula solicitações para a camada de armazenamento principal, um serviço separado para manipular
enviando e-mails para clientes, outro que lida com um usuário baseado na web interface, e ainda outro que lida com cache do mais recente mercado de ações dados. São algumas peças distintas, que podem te fazer pensar
sobre cada peça como um conjunto de recipientes, com alguns que podem ser agrupados em um pod.

Tabela 10.2. E * necessidades de computação do Exchange

Aspecto	Necessidades	Bom ajuste? Flexibilidade	Complexidade	atuação	Custo
Bastante	Bom investir em aprendizado	Moderado	Nada extravagante	Definitivamente	Definitivamente, se facilitar as coisas
Definitivamente	Definitivamente	Definitivamente			

Como o E * Exchange era um ajuste razoável para o Compute Engine, provavelmente será uma boa opção para o Kubernetes Engine. Acontece também que o
benefícios de investir em aprender Kubernetes e implantar o
serviços usando o Kubernetes Engine podem economizar um pouco de tempo e simplifique o processo geral de implantação do aplicativo. Ao contrário do To-Do List, este aplicativo tem algumas peças distintas, cada uma com
seus próprios requisitos exclusivos. Usando vários pods diferentes para o pedaços permite que você mantenha todos eles em um único cluster e escalá-los
para cima ou para baixo, conforme necessário. No geral, o Kubernetes Engine é provavelmente um ótimo
adequado para o aplicativo E * Exchange.

10.8.8. InstaSnap

InstaSnap, o aplicativo de compartilhamento de fotos de mídia social (tabela 10.3), mentiras
em algum lugar no meio dos dois exemplos anteriores em termos de complexidade geral do sistema. Provavelmente não tem tantos distintos
sistemas como E * Exchange, mas definitivamente tem mais do que o simples Fazer lista. Por exemplo, ele pode usar um servidor de API que manipula solicitações

a partir do aplicativo móvel, um serviço para a interface do usuário baseada na Web e talvez um
serviço de segundo plano que lida com vídeos e fotos de processamento
tamanhos e formatos diferentes.
Dito isto, a maior preocupação para o InstaSnap é o desempenho e
escalabilidade. Você pode precisar da capacidade de aumentar os recursos
disponível a qualquer um dos vários serviços, se houver um aumento na demanda
acontece frequentemente) ocorre. Esse requisito faz do InstaSnap um ótimo ajuste
para o Kubernetes Engine, porque você pode facilmente redimensionar o cluster como um todo, bem como o número de réplicas do pod em execução no cluster.
Tabela 10.3. Necessidades de computação do InstaSnap
Aspecto
Flexibilidade
Complexidade
atuação
Custo
Necessidades
Muito
Ansioso para usar recursos avançados
Alto
Nenhum orçamento real
Bom encaixe?
Definitivamente
Definitivamente
Na maioria das vezes
Definitivamente
Como você pode ver na tabela 10.3, mesmo que você não tenha tantos
serviços distintos como o E * Exchange, o Kubernetes Engine ainda é um ótimo ajuste
para InstaSnap, particularmente quando se trata de usar o avançado
recursos de escalabilidade. Embora o desempenho em si seja um pouco menor, com base em mais abstração acontecendo sob o capô, este requisito
tem pouco efeito sobre a escolha de uma plataforma de computação. Você pode sempre
adicionar mais máquinas se você precisar de mais capacidade (o que é OK devido ao
"Não há orçamento real" necessidade de custo).
RESUMO
Um contêiner é uma ferramenta de infra-estrutura que facilita o empacotamento de código
junto com todas as dependências até o sistema operacional.
O Docker é a maneira mais comum de definir um contêiner, usando um formato chamado de Dockerfile.

O Kubernetes é um sistema de código aberto para orquestrar containers, ajudando
eles agem como uma aplicação coesa.
O Kubernetes Engine é uma implantação hospedada e totalmente gerenciada de Kubernetes, minimizando a sobrecarga de executar seus próprios Kubernetes grupo.
Você pode gerenciar seu cluster do Kubernetes Engine como qualquer outro cluster do Kubernetes, usando o kubectl.Capítulo 11. App Engine: totalmente gerenciado
aplicações
Este capítulo cobre
O que é o App Engine e quando é um bom ajuste?
Construindo um aplicativo usando as versões Standard e Flex
Gerenciando como seus aplicativos aumentam e diminuem
Usando os serviços gerenciados do App Engine Standard
Como você aprendeu, existem muitas plataformas de computação disponíveis, representando uma grande variedade em termos de complexidade, flexibilidade e
desempenho. Enquanto o Compute Engine era um exemplo de baixo nível infra-estrutura (uma VM), o App Engine é uma nuvem totalmente gerenciada ambiente de computação que visa consolidar todo o trabalho
necessário ao implantar e executar seus aplicativos. Além de
sendo capaz de executar o código como você faria em uma VM, o App Engine oferece
vários serviços que são úteis ao criar aplicativos.
Por exemplo, se você tivesse um aplicativo de lista de tarefas que exigia o armazenamento
listas de trabalho que você precisava para terminar, não seria incomum para você
precisa armazenar alguns dados, enviar e-mails ou agendar um trabalho em segundo plano
todos os dias (como recalcular sua taxa de conclusão da lista de tarefas).
Normalmente, você precisa fazer tudo isso ligando um
banco de dados, inscrevendo-se em um serviço de envio de e-mail, executando um enfileiramento
sistema como o RabbitMQ, e confiando no serviço cron do Linux para
coordene tudo. O App Engine oferece um conjunto de serviços hospedados para fazer
isso para que você não precise administrá-lo você mesmo. O App Engine é composto de dois ambientes separados que
diferenças importantes. Um ambiente é construído usando código aberto
ferramentas como contêineres Docker. O outro é construído usando mais propriedade
tecnologia que permite ao Google fazer coisas interessantes quando
dimensionar automaticamente seu aplicativo, embora imponha vários

limitações sobre o que você pode fazer com seu código. Ambos os ambientes são
sob o guarda-chuva do Google App Engine, mas eles estão pressionando contra o
limites do que você poderia considerar um único produto. Como um resultado,
nós vamos olhar para eles juntos em um capítulo, mas em alguns lugares, nós vamos
bateu uma bifurcação na estrada. Nesse ponto, fará sentido dividir os dois
ambientes separados.
O ambiente padrão do Google App Engine, lançado no início de 2008, oferece
um ambiente de computação totalmente gerenciado completo com armazenamento,
cache, computação, agendamento e muito mais. Mas é limitado a alguns
linguagens de programação. Neste tipo de ambiente, sua aplicação
tende a ser adaptada ao Google App Engine, mas se beneficia da vida em um
ambiente que é sempre autoscaling. O Google App Engine lida com súbita
picos de tráfego enviados ao seu aplicativo graciosamente e períodos em que
seu aplicativo está inativo, não lhe custa dinheiro.
Ambiente flexível do App Engine (geralmente chamado de App Engine Flex)
fornece um ambiente totalmente gerenciado com menos restrições e
um pouco mais de portabilidade, negociando alguma escalabilidade em troca.
Aplicativo
O Engine Flex é baseado em contêineres Docker, você não está limitado a
versões específicas de linguagens de programação, e você ainda pode
vantagem de muitos dos outros benefícios do Google App Engine, como o
serviço cron hospedado.
Se você está confuso sobre qual ambiente é ideal para você,
capítulo ajudará a esclarecer as coisas. Vamos primeiro explorar alguns dos
conceitos organizacionais, depois aprofundar os detalhes e, finalmente,
Veja como escolher se o App Engine é ideal para você. Se isso virar
que o App Engine é um ótimo ajuste, exploraremos como escolher quais
dos dois ambientes é melhor para atender às suas necessidades.

11,1. CONCEITOS

Como o App Engine é um ambiente hospedado, a camada da API tem alguns
mais conceitos organizacionais que você precisa entender para usar
App Engine como uma plataforma de computação. O App Engine usa quatro
conceitos organizacionais para entender mais sobre seu aplicativo:
aplicativos, serviços, versões e instâncias (figura 11.1).
Figura 11.1. Uma visão geral dos conceitos do App Engine
Nota
Lembre-se de que, embora o App Engine ofereça dois ambientes, o
conceitos em ambos os ambientes são os mesmos (ou muito semelhantes). Você
não terá que reaprender esses conceitos para alternar entre os ambientes.
Além de analisar seu aplicativo em termos de componentes, o App Engine
acompanha as versões desses componentes. Para

Por exemplo, em seu aplicativo de lista de tarefas, você pode quebrar o sistema em componentes separados: um componente representando a web aplicação em si e outro responsável pela recomputação de estatísticas todos os dias à meia-noite (figura 11.2). Depois disso, revisando um componente de tempos em tempos (por exemplo, consertando um bug no aplicativo da web) pode trazer uma nova versão desse componente.

Figura 11.2. Uma visão geral de componentes e versões de um aplicativo de lista de tarefas pendentes

O App Engine usa e entende todas essas coisas (figura 11.1) e Vamos explorá-los em mais detalhes nesta seção.

Vamos começar no topo olhando a ideia de um App Engine aplicação.

11.1.1. AplicativosO ponto de partida básico para usar o Google App Engine para hospedar seu trabalho é o

aplicação de nível superior. Cada um dos seus projetos é limitado a um aplicação, com a ideia de que cada projeto deve ter um propósito.

Como um projeto atua como um contêiner para suas VMs do Compute Engine, o aplicativo atua como um contêiner para o seu código, que pode se espalhar em vários serviços. (Vou discutir isso na próxima seção)

O aplicativo também tem muitas configurações. Você pode configurar e mudar alguns deles facilmente, enquanto outros são permanentes e trancados para a aplicação, uma vez definido. Por exemplo, você sempre pode reconfigurar um certificado SSL para o seu aplicativo, mas depois de ter escolhido o região para o seu aplicativo, você não pode mudar isso.

Nota

A localização do seu aplicativo também afeta quanto custa executar, que vamos explorar no final do capítulo.

Para ver como isso funciona, clique na seção do App Engine no lado esquerdo navegação do Cloud Console. Se você não configurou o aplicativo Engine antes, você será solicitado a escolher um idioma (por exemplo, Python); então você vai pousar em uma página onde você escolhe a localização de sua aplicação (figura 11.3). Essa configuração específica controla onde os recursos físicos para sua aplicação irão viver e é um exemplo de uma configuração que, uma vez escolhida, você não pode mudar.

Figura 11.3. Como escolher um local para um aplicativo do App Engine. Além disso, as partes mais interessantes, como serviços, são aquelas que um aplicativo do App Engine contém. Vamos passar a olhar para Serviços do App Engine.

11.1.2. Serviços

Os serviços no Google App Engine fornecem uma maneira de dividir seu aplicativo em peças menores e mais gerenciáveis. Semelhante aos microservices, App

Os serviços do mecanismo agem como componentes independentes da computação, embora normalmente compartilhem o acesso aos vários mecanismos compartilhados do Google App Engine.
APIs. Por exemplo, você pode acessar a mesma API cron compartilhada de qualquer um dos vários serviços que você pode ter como parte de seu aplicativo.
Por exemplo, imagine que você está criando um aplicativo da Web que rastreia sua lista de tarefas. No começo, pode envolver apenas texto simples, mas como você
crescer, você pode adicionar um recurso que envia lembretes por e-mail para terminar algo na sua lista. Nesse caso, em vez de tentar adicionar o
recurso de lembrete de e-mail para o aplicativo principal, você pode defini-lo como
um serviço separado dentro do seu aplicativo. Porque o seu trabalho é completamente isolado do trabalho principal de armazenar itens de tarefas, pode
viver como um serviço separado e evitar desordenar o aplicativo principal (figura 11.4).
Figura 11.4. Um aplicativo de lista de tarefas com dois serviçosO serviço em si consiste em seus arquivos de código-fonte e
configuração, como qual runtime usar (para o App Engine
Padrão). Ao contrário dos aplicativos (que têm um para um
relacionamento com seu projeto), você pode criar (implantar) e também excluir serviços. O primeiro conjunto de código-fonte que você implanta no aplicativo
O mecanismo criará um novo serviço, que o Google App Engine registrará como o serviço padrão para seu aplicativo. Quando você faz uma solicitação para seu aplicativo sem especificar o serviço, o App Engine encaminhará
a solicitação para este novo serviço padrão.
Os serviços também atuam como outro contêiner para revisões de sua aplicação. No exemplo da lista de tarefas, você poderia implantar novas versões do seu serviço de lembrete para o seu aplicativo sem ter que tocar no serviço de aplicativo da web. Como você pode imaginar, ser capaz de isolar
mudanças entre sistemas relacionados podem ser úteis, particularmente com grandes
aplicações construídas por grandes equipes, onde cada equipe possui um peça do aplicativo. Vamos continuar e ver como as versões
trabalhos.
11.1.3. Versões
Versões em si são muito parecidas com instantâneos pontuais de um serviço.
Se o seu serviço é um monte de código dentro de um único diretório, uma versão
do seu serviço corresponde ao código nesse diretório no exato
hora em que você decidiu implantá-lo no App Engine. Um efeito colateral esta configuração é que você pode ter várias versões do mesmo serviço correndo ao mesmo tempo.

Semelhante a como o primeiro serviço do Google App Engine que você implanta se torna o
serviço padrão para todo o seu aplicativo, o código que você implanta no
esse primeiro serviço se torna a versão padrão desse serviço. Você pode
abordar uma versão individual de qualquer serviço, como você pode
abordar um serviço individual da sua aplicação. Por exemplo, você
pode ver seu aplicativo da web navegando para webapp.my-list.appspot.com (ou explicitamente, default.webapp.my-list.appspot.com), ou você pode ver uma versão recém-implantada (talvez chamada versão v2, como na figura 11.5) navegando para v2.webapp.my-list.appspot.com.

Figura 11.5. Implantando uma nova versão do serviço de aplicativo da web. Eu abordei os conceitos organizacionais de aplicativos, serviços e
versões. Agora vamos dar uma olhada para uma infra-estrutura:
instâncias.

11.1.4. Instâncias

Embora tenhamos analisado os conceitos organizacionais no Google App Engine,
você não viu como o Google App Engine executa seu código.
Dado o que você aprendeu até agora, não é surpresa que
O App Engine usa o conceito de uma instância para significar um pedaço de capacidade de computação para sua aplicação. Ao contrário dos conceitos que tenho
abordados neste capítulo até agora, você encontrará algumas pequenas diferenças
nos casos, dependendo se você está usando o padrão ou
Ambiente flexível, e vale a pena explorar um pouco mais detalhadamente (figura 11.6).

Figura 11.6. Instâncias do Google App Engine para ambientes padrão x flexível

No padrão do Google App Engine, essas instâncias representam um bloco abstrato
de CPU e memória disponível para executar seu aplicativo dentro de um especial
caixa de areia. Eles aumentam e diminuem automaticamente com base em quantas
solicitações estão sendo enviadas para o seu aplicativo. Porque eles são ambientes sandbox leves, seu aplicativo pode ser dimensionado
zero a milhares de instâncias rapidamente. Você pode escolher o tipo de Instância do Google App Engine para usar em seu aplicativo em uma lista de tipos disponíveis que têm custos e quantidades variáveis de CPU e memória.

Como o App Engine Flex é construído sobre o Compute Engine e
Contêineres do Docker, ele usa instâncias do Compute Engine para executar seu código,
que vem com algumas ressalvas importantes. Primeiro, porque Calcular
As VMs do mecanismo levam algum tempo para serem ativadas, os aplicativos Flex devem sempre

ter pelo menos uma única instância de VM em execução. Como resultado, o Flex aplicativos acabam custando dinheiro o tempo todo. Por causa do
tempo de inicialização adicional, se você ver um enorme pico de tráfego para o seu
aplicação, pode demorar um pouco para aumentar a escala para lidar com o tráfego.
Durante esse tempo, as instâncias existentes podem ficar sobrecarregadas, o que
Isso levaria a tempos limite para solicitações recebidas. Também é importante lembrar que as instâncias do App Engine são específicas
para uma única versão do seu serviço, portanto, uma única instância só lida com
solicitações para a versão específica do serviço que as recebeu. Como um
Se você hospedar muitas versões simultaneamente, essas versões serão
spawn instâncias conforme necessário para atender o tráfego. Se eles estão correndo
dentro do App Engine Flex, cada versão terá pelo menos uma VM
correndo em todos os momentos.
Isso conclui o resumo dos conceitos envolvidos no Google App Engine.
Agora vamos aos negócios e ver como usá-lo.

11,2. INTERAÇÃO COM O MOTOR DO APP

Neste ponto, você deve ter uma compreensão decente do
conceitos organizacionais subjacentes que o Google App Engine usa (como serviços ou versões), mas isso não é tão útil até que você
algo com eles. Para esse fim, você criará um simples "Olá,
mundo! ”para o Google App Engine, implante-o e verifique se ele
trabalho. Você pode criar o aplicativo para o padrão do Google App Engine primeiro.

11.2.1. Construindo um aplicativo no padrão do Google App Engine

Como discutimos anteriormente, o App Engine Standard é totalmente gerenciado
ambiente onde seu código é executado dentro de uma caixa de areia especial em vez
do que uma máquina virtual completa, como acontece no Compute Engine. Como um
resultado, você tem que construir o seu aplicativo "Hello, world!" usando um dos
os idiomas aprovados. Das linguagens disponíveis (PHP, Java,
Python, e Go), Python parece uma boa escolha (é poderoso e fácil
para ler), portanto, para esta seção, você vai usar o Python para
construa sua aplicação.
Não se preocupe se você não estiver familiarizado com o Python. Vou anotar qualquer
Código Python que não é super óbvio para explicar o que ele faz.
Uma coisa a ter em mente é que o espaço em branco (por exemplo, espaços e guias) é importante no Python. Se você se encontrar com erros de sintaxe

seu código Python, pode ser que você tenha usado uma guia quando quis use quatro espaços, então tenha cuidado!

Antes de começar a construir o código do seu aplicativo, primeiro você precisa Certifique-se de ter as ferramentas certas instaladas. Você precisará deles para implante seu código no App Engine.

Instalando extensões do Python

Para desenvolver localmente usando o App Engine (e, especificamente, usando o App

Python do Engine Standard), você precisará instalar o Python extensões para o Google App Engine, o que você pode fazer usando o gcloud subcomando componentes. Este pacote contém o local servidor de desenvolvimento, vários emuladores e outros recursos e bibliotecas que você precisa para criar um aplicativo do Python App Engine:

Componentes do $ gcloud instalam app-engine-python

Gorjeta

Se você instalou o Cloud SDK usando um gerenciador de pacotes no nível do sistema

(como o apt-get), você receberá uma mensagem de erro dizendo para usar o mesmo

gerenciador de pacotes e para executar o comando para instalar o Python extensões.Criando uma aplicação

Com tudo instalado, você pode chegar ao trabalho real de construção sua aplicação. Como você está testando apenas o App Engine, você pode começar concentrando-se em nada mais do que um "Olá, mundo!" aplicação que envia uma resposta estática de volta sempre que recebe um pedido. Você começará seu aplicativo Python usando o webapp2 framework, que é compatível com o Google App Engine.

Nota

Você pode usar outras bibliotecas e frameworks também (como o Django ou Flask), mas o webapp2 é o mais fácil de usar com o Google App Engine.

A próxima listagem mostra um aplicativo webapp2 simples que define um manipulador de solicitação única e conecta-o à URL raiz (/). Nisso caso, sempre que você enviar uma solicitação HTTP GET para esse manipulador, ele envia voltar "Olá do App Engine!"

Listagem 11.1. Definindo seu aplicativo da web simples

```
importar webapp2
classHelloWorld (webapp2.RequestHandler):
Defeito (auto):
self.response.write ('Olá do App Engine!');
app = webapp2.WSGIApplication ([
('/', Olá Mundo),
])
```

Você pode colocar esse código em um arquivo chamado main.py; então você passará a definir a configuração do seu aplicativo do Google App Engine. o

maneira que você diz ao Google App Engine como configurar um aplicativo com um
arquivo app.yaml. YAML (Yet Another Markup Language) tem uma facilidade
sintaxe legível para alguns dados estruturados que se parece muito com
Markdown e o nome app.yaml é um arquivo especial que o Google App Engine
procura quando você implementa seu aplicativo. Ele contém configurações sobre
o tempo de execução envolvido, manipuladores para mapeamentos de URL e muito mais. Você pode
veja o arquivo app.yaml que você usará na listagem a seguir.
Listagem 11.2. Definindo app.yaml

1 Diz ao Google App Engine para executar seu código dentro da sandbox do Python 2.7
2 Informa ao Google App Engine qual versão da API você está usando. (Atualmente,
há apenas uma versão para o Python 2.7, então isso deve ser definido como 1.)
3 Diz ao Google App Engine que você escreveu seu código para ser thread-safe e
O App Engine pode gerar com segurança várias cópias do seu aplicativo
sem se preocupar com esses fios se enroscando um com o outro
4 Seção que contém os manipuladores que mapeiam padrões de URL para um dado
roteiro
5 Uma expressão regular que corresponde a solicitações - se uma solicitação
Correspondências de URL, o script será usado para manipular a solicitação.
6 aponta para o arquivo main.py, mas o sufixo "app" diz ao App Engine para
tratar main.py como um aplicativo de interface de gateway de servidor da web (WSGI)
(que diz para olhar a variável de aplicativo em main.py)
Neste ponto, você tem tudo o que precisa para testar sua aplicação. É hora de tentar executá-lo localmente e ter certeza de que funciona
como você quiser.
Testando o aplicativo localmente
Para executar seu aplicativo, você usará o desenvolvimento do App Engine
servidor, que foi instalado como dev_appserver.py quando você instalou
as extensões do Python App Engine. Navegue até o diretório que
contém seu arquivo app.yaml (e o arquivo main.py) e executa
dev_appserver.py apontando para o diretório atual (.). Você deveria ver
alguma saída de depuração que diz onde o próprio aplicativo está disponível
(geralmente no host local na porta 8080). Depois do desenvolvimento
servidor está sendo executado com seu aplicativo, você pode testar se ele
coisa certa, conectando-se a http: // localhost: 8080 /:
$ curl http: // localhost: 8080 /
Olá do App Engine!
Por enquanto, tudo bem! Agora que você construiu e testou seu aplicativo, é

hora de ver se você pode implantá-lo no App Engine. Depois de tudo, executar o aplicativo localmente não vai funcionar quando você tem lotes do tráfego de entrada.

Implantando no padrão do Google App Engine

Implantar o aplicativo no App Engine é fácil porque você tem tudo as ferramentas certas instaladas. Para fazer isso, você pode usar o aplicativo gcloud implantar o subcomando, confirmar o local para o qual você está implantando e aguarde a conclusão da implantação:

```
implantação do aplicativo $ gcloud
Inicializando recursos do App Engine ... concluídos.
Serviços para implantar:
1descriptor: [/home/jjg/projects/appenginehello/app.yaml] source:
projeto de destino: [/ home / jjg / projects / appenginehello]
[seu-projeto-id-aqui] 2
serviço de destino:
versão de destino: [padrão]
[20171001t160741] 3
4
URL de destino: [https://your-project-id-here.appspot.com] 5
Você quer continuar (S / n)?
Y
6
Começando a implantação do serviço [padrão] ...
Alguns arquivos foram ignorados. Passe `--verbosity = info` para ver quais.
Você também pode ver o arquivo de log do gcloud, encontrado em
[/home/jjg/.config/gcloud/logs/2017.10.01/16.07.33.825864.log].
Upload de arquivo feito.
Atualizando o serviço [padrão] ... pronto.
Aguardando operação [apps / your-project-id-here / operações / 1fad9f55-
35bb
-45e2-8b17-3cc8cc5b1228] para completar ... feito.
Atualizando o serviço [padrão] ... pronto.
Serviço implantado [padrão] para [https: // your-project-id-
aqui.appspot.com]
Você pode transmitir logs a partir da linha de comando executando:
$ gcloud app logs tail -s padrão
Para ver sua aplicação no navegador da web, execute:
$ gcloud app browse
```

1 Verifica a configuração do que você está planejando implantar

2 O ID do projeto e o ID do aplicativo são os mesmos (porque eles ter um relacionamento um-para-um).

3 Se nenhum nome de serviço estiver definido (que é o caso aqui), o App Engine usa o serviço padrão.

4 Se nenhum nome de versão for especificado, o App Engine gera um padrão

número da versão com base na data.
5 Após implantar seu aplicativo, ele estará disponível em um URL no domínio appspot.com.
6 Solicita que você confirme os parâmetros de implantação para evitar implantar acidentalmente o código errado ou para o serviço errado
Quando isso estiver concluído, você pode verificar se tudo funcionou usando o comando curl ou através de seu navegador enviando um GET solicitação para o URL de destino das informações de implantação. o O comando curl produz o seguinte:

```
$ curl http://your-project-id-here.appspot.com
Olá do App Engine!
```

Você também pode verificar se o SSL funciona com seu aplicativo conectando usando https: // como o esquema em vez de simples http: //:

```
$ curl https://your-project-id-here.appspot.com
Olá do App Engine!
```

Por fim, como o nome do serviço é oficialmente "padrão", você pode Endereçá-lo diretamente em default.your-project-id-here.appspot.com:

```
$ curl http://default.your-project-id-here.appspot.com
Olá do App Engine!
```

Você pode verificar dentro da seção do Google App Engine do Cloud Console e ver quantas solicitações foram enviadas, quantas instâncias você atualmente ativados e muito mais. Um exemplo do que isso pode aparência é mostrada na figura 11.7.
Figura 11.7. O painel de visão geral do App Engine no Cloud ConsoleComo você cria novos serviços? Vamos dar um momento e explorar como implantar código em um serviço diferente do padrão.
Implantando outro serviço
De certa forma, você pode pensar em um novo serviço como um novo pedaço de código, e você precisa ter certeza de que tem um local seguro para colocar esse código. Normalmente, a maneira mais fácil de configurar isso é separar blocos de código diretório, onde o nome do diretório corresponde ao nome do serviço.
Para ver como isso funciona, crie dois novos diretórios chamados default e service2, e copie os arquivos app.yaml e main.py em cada diretório.
Isso efetivamente reorganiza seu código para que você tenha duas cópias de ambos o código e a configuração em cada diretório.
Para ver isso com mais clareza, veja como deve ficar quando terminar: $ tree

```
.
padrão
app.yaml
main.py
service2
app.yaml
```

main.py
2 diretórios, 4 arquivos
Agora você pode fazer algumas coisas para definir um segundo serviço (e esclarecer
que o serviço atual seja o padrão):
1. Atualize os arquivos app.yaml para escolher explicitamente um nome de serviço,
o padrão será chamado padrão.
2. Atualize service2 / main.py para imprimir outra coisa.
3. Reimplemente os dois serviços.
Depois de atualizar os dois arquivos app.yaml, eles devem se parecer com o seguindo duas listagens.
Listagem 11.3. Padrão / app.yaml atualizado
tempo de execução: python27
api_version: 1
threadsafe: true
serviço: padrão
1
manipuladores:
- url: /.*
script: main.app
1 Declara explicitamente que o serviço envolvido é o padrão - isso tem
nenhum efeito real, neste caso, mas esclarece o que este arquivo app.yaml
controls.Listing 11.4. Serviço2 atualizado / app.yaml
tempo de execução: python27
api_version: 1
threadsafe: true
serviço: service2
1
manipuladores:
- url: /.*
script: main.app
1 Escolhe um novo nome de serviço, que pode ser qualquer cadeia de estilo ID que
você quer
Como você pode ver, você explicitou que cada arquivo app.yaml
controla um serviço diferente. Você também se certificou de que o nome do serviço
corresponde ao nome do diretório, o que significa que é fácil acompanhar todos o código fonte diferente e arquivos de configuração.
Em seguida, você pode atualizar service2 / main.py, alterando a saída para que você
sei que veio desse outro serviço. Isso pode fazer com que seu
aplicativo se parece com a listagem a seguir.
Listagem 11.5. O serviço2 "Hello, world!" Application em Python
importar webapp2

classHelloWorld (webapp2.RequestHandler):
Defeito (auto):
self.response.write ('Olá do serviço 2!');
1
app = webapp2.WSGIApplication ([
('/', Olá Mundo),
])
1 Torna claro que o service2 está respondendo.
Finalmente, você pode implantar seu novo serviço executando o app gcloud implantar e apontar no diretório service2 em vez do diretório padrão:
$ gcloud app deploy service2
1
Serviços para implantar:
descritor: [/home/jjg/projects/appenginehello/service2/app.yaml]
2
fonte: [/ home / jjg / projects / appenginehello / service2]
projeto alvo:
serviço de destino: [seu-projeto-id-aqui]
[serviço2]
3
versão de destino: [20171002t053446]
URL de destino: [https: // service2-dot-your-project-id
-here.appspot.com]
4
 # ... Mais informações aqui ...
1 Aponta a ferramenta de implementação do gcloud para o seu diretório service2
2 Como resultado, a ferramenta de implementação procura o arquivo copiado app.yaml,
que indica um nome de serviço diferente.
3 figuras fora o nome do serviço deve ser service2 como você definiu
4 Como o nome do serviço não é "padrão", você recebe um URL separado onde você pode acessar seu código.
Como antes, seu novo serviço de aplicativo deve estar ativo. E neste ponto, seu sistema conceitualmente parece um pouco com a figura 11.8.
Figura 11.8. Layout organizacional do seu aplicativo até o momentoVocê pode verificar se a implantação funcionou navegando até o URL novamente em um navegador, ou você pode usar a linha de comando:
$ curl https://service2-dot-your-project-id-here.appspot.com
Olá do serviço 2!
Se esta URL parece estranha para você, isso não é incomum. A sintaxe de <service> -dot- <application> é definitivamente novo. Isso existe por causa de como os certificados SSL funcionam. O App Engine garante que * .appspot.com seja
protegido, mas não permite pontos adicionais aninhados mais fundo no DNS hierarquia. Ao acessar seu aplicativo via HTTP (não HTTPS), você

tecnicamente pode fazer uma chamada para <service>. <application> .appspot.com,
mas se você tentasse isso com HTTPS, você teria problemas:
$ curl http://service2.your-project-id-here.appspot.com
Olá do serviço 2!
$ curl https://service2.your-project-id-here.appspot.com
 # Erro
1
1 O resultado é um código de erro devido ao certificado SSL não cobrir o domínio especificado.
Você viu como implantar um novo serviço. Agora vamos olhar um pouco menos aventureiro mudança, implantando uma nova versão de um existente service.Deploying uma nova versão
Embora você possa criar novos serviços apenas de vez em quando, qualquer atualização para o seu aplicativo provavelmente resultará em uma nova versão, e
atualizações acontecem com muito mais frequência. Então, como você atualiza versões?
Onde o App Engine armazena as versões?
Para começar, confirme como sua aplicação está organizada. Você pode inspecionar
seu aplicativo atual no Cloud Console ou no aplicativo
linha de comando:
Lista de serviços de aplicativos do $ gcloud
Como você pode ver, atualmente você tem dois serviços, cada um com um único
versão padrão especificada. Agora imagine que você deseja atualizar o serviço padrão, mas esta atualização deve criar uma nova versão e não Sobrescrever a versão atualmente implantada do serviço.
Para atualizar o serviço dessa maneira, você tem duas opções. O primeiro é dependem do esquema de nomenclatura de versão padrão do App Engine, que é baseado
na data e hora em que a implantação aconteceu. Quando você implantar seu código, o App Engine cria uma nova versão automaticamente para você e nunca sobrescreve a versão atualmente implantada, que é
útil quando você acidentalmente implanta o código errado! O outro é usar um sinalizador especial (-v) ao implantar seu código, e o resultado ser uma nova versão chamada como você especificou.Se você quiser atualizar sua versão padrão, você pode fazer o seu código
mudanças e implantar como você fez antes. Neste exemplo, você atualizar o código para dizer "Olá da versão 2!"
conclui, você pode verificar se tudo funcionou como esperado tentando acessar o URL como antes:
$ curl https://your-project-id-here.appspot.com
Olá da versão 2!
Isso pode parecer que você acidentalmente explodiu o anterior

versão, mas se você inspecionar a lista de versões novamente, você verá que o versão anterior ainda está lá e servindo tráfego:
$ gcloud app versions list --service = default
Observe que o tráfego dividido entre as duas versões mudou e todo o tráfego está apontando para a versão posterior, com zero tráfego sendo roteado para a versão anterior. Vou discutir isso em mais detalhes mais adiante. Se o versão ainda está lá, como você pode falar com isso? Acontece que, assim como você pode acessar um serviço específico diretamente, você pode acessar o anterior versão, endereçando-a diretamente no formato <version>.
<service> .your-project-id-here.appspot.com (ou usando -dot-separators para HTTPS):
$ curl http://20171001t160741.default.your-project-id-here.appspot.com
Olá do App Engine!
$ curl https: // 20171001t160741-dot-default-dot-your-project-id -here.appspot.com
Olá do App Engine! É completamente razoável se você está preocupado com uma nova versão indo viver imediatamente. Você tem uma maneira de informar ao Google App Engine que você quer a nova versão implantada, mas não quer encaminhar imediatamente todos tráfego para a nova versão. Você pode atualizar o código novamente para alterar mensagem e implantar outra versão, sem que se torne o live versão imediatamente. Para fazer isso, você vai definir o sinalizador promote_by_default para false:
Conjunto de configurações $ gcloud app / promote_by_default false
Propriedade atualizada [app / promote_by_default].
$ gcloud app deploy default
Serviços para implantar:
descritor: [/home/jjg/projects/appenginehello/default/app.yaml]
fonte:
projeto alvo:
serviço de destino:
versão de destino:
URL de destino: [/ home / jjg / projects / appenginehello / default]
[seu-projeto-id-aqui]
[padrão]
[20171002t074125]
[https: // 20171002t074125-dot-your-project-id -here.appspot.com]
(adicione --promover se você também quiser disponibilizar este serviço de
[https://your-project-id-here.appspot.com])
 # ... Mais informações aqui ...

Neste ponto, a nova versão do serviço deve ser implantada, mas não ao vivo e atender solicitações. Você pode ver a lista de serviços para verificar se da seguinte maneira, ou verifique fazendo uma solicitação para o URL de destino como você fez antes:

```
$ gcloud app versions list --service = default
SERVING $ curl http://your-app-id-here.appspot.com/
Olá da versão 2!
```

Você também pode verificar se a nova versão foi implantada corretamente acessando da mesma maneira que você fez antes:

```
$ curl http://20171002t074125.default.your-project-id-here.appspot.com
Olá da versão 3, que ainda não está ao vivo!
```

Depois de ver que a nova versão funciona da maneira esperada, você pode promovê-lo com segurança, migrando todo o tráfego para ele usando o Cloud Console.

Para fazer isso, você navega até a lista de versões, verifica a versão que deseja migrar o tráfego para e clique em Migrar tráfego na parte superior do página (figura 11.9).

Figura 11.9. Marcar a caixa da versão e clicar em Migrar Tráfego

Quando você clica no botão, você verá um pop-up onde você pode confirmar que você deseja rotear todo o tráfego novo para a versão selecionada (figura 11,10).

Figura 11.10. Surpreenda-se para confirmar que você deseja migrar o tráfego para a nova versão. Quando isso estiver concluído, você verá que 100% do tráfego está sendo enviado para sua nova versão:

```
$ gcloud app versions list --service = default
$ curl https://your-project-id-here.appspot.com
Olá da versão 3, que ainda não está ao vivo!
1 Obviamente, está vivo agora!
```

Você viu como a implantação funciona no padrão do Google App Engine Meio Ambiente. Agora vamos fazer um desvio e ver como as coisas funcionam o ambiente flexível.

11.2.2. No App Engine Flex

Como discuti anteriormente, enquanto o padrão do Google App Engine é limitado a algumas das linguagens de programação populares e corre dentro de um ambiente do sandbox, o App Engine Flex é baseado em Dockercontainers, para que você possa usar qualquer linguagem de programação desejada. Você volte para o Node.js ao criar seu "Hello, world!" aplicação. Vamos começar!

Criando um aplicativo

Similarmente ao exemplo que usei ao criar um aplicativo para o aplicativo Engine Standard, você começará criando um aplicativo “Hello, world!” usando Express (uma estrutura popular de desenvolvimento da Web para o Node.js).

Nota
Você pode usar qualquer estrutura da Web que desejar. Express acontece ser popular e bem documentado, então eu vou usar isso para o exemplo.
Primeiro, crie um novo diretório chamado default-flex para manter o código para este novo aplicativo, juntamente com os outros diretórios que você já tem.
Depois disso, você deve inicializar o aplicativo usando npm (ou fio)
e adicione express como uma dependência:

```
$ mkdir default-flex
$ cd default-flex
$ npminit
# ...
Escreveu para /home/jjg/projects/appenginehello/default-flex/package.json:
{
"nome": "appengineflexhello",
"version": "1.0.0",
"descrição": "",
"main": "index.js",
"scripts": {
"test": "echo \" Erro: nenhum teste especificado \ "&& exit 1"
}
"palavras-chave": [],
"author": "", "license": "ISC"
}
$ npm install express
# ...
```

Agora que você definiu seu pacote e definiu as dependências que precisa, você pode escrever um script simples que usa Express para lidar com HTTP solicitações de. Este script, mostrado na listagem a seguir, que você colocará dentro de app.js, vai receber qualquer solicitação enviada para / e enviar "Olá, mundo! "como uma resposta.

Listagem 11.6. Definindo um aplicativo simples "Hello, world!" No Node.js

```
'use estrito';
const express = require ('express');
const app = express ();
1
app.get ('/', (req, res) => {
res.status (200) .send ('Olá do App Engine Flex!'). end ();
}); 2
const PORT = process.env.PORT || 8080; 3
app.listen (PORT, () => {
console.log (`App ouvindo na porta $ {PORT}`);
console.log ('Pressione Ctrl + C para sair.');
});
```

1 Usa a estrutura do Express para o Node.js
2 Define o código de resposta para 200 OK e retorna um pequeno "Olá,

mundo! ”digite mensagem
3 Tenta ler um número de porta do ambiente, mas o padrão é
8080 se não estiver definido
Depois de escrever esse script, você pode testar se seu código funciona executando-o e, em seguida, tentando se conectar a ele usando curl, como mostrado aqui,
ou seu navegador: $ node app.js
App ouvindo na porta 8080
Pressione Ctrl + C para sair.
$ curl http: // localhost: 8080
Olá do App Engine Flex!
1
1 Isso é executado a partir de um terminal separado na mesma máquina.
Agora que você tem certeza de que o código funciona, você pode voltar a trabalhar
implantá-lo no App Engine. Como antes, você precisará definir também um
Um pouco de configuração que explica ao Google App Engine como executar seu código.
Assim como no padrão do Google App Engine, você colocará as opções de configuração em
um arquivo chamado app.yaml. A principal diferença aqui é que porque um Flex-aplicação baseada é baseada no Docker, a configuração que você coloca
app.yaml é muito menos envolvido:
tempo de execução: nodejs
env: flex
serviço: padrão
1
2
1 Um novo parâmetro chamado env explica ao Google App Engine que você pretende
para executar seu aplicativo fora do ambiente padrão.
2 Para este exemplo, você implantará o serviço Flex na parte superior
Serviço padrão.
Como você pode ver, este arquivo de configuração é muito mais simples do que o que você
usado ao criar um aplicativo para o Google App Engine Standard. Aplicativo
O Engine Flex só precisa saber como pegar seu código e colocá-lo em
um container. Em seguida, em vez de configurar informações de roteamento, você precisa
dizer ao Google App Engine como iniciar seu servidor. App Engine Flex's
O tempo de execução do nodejs sempre tentará executar o npm start como comando de inicialização, para que você possa configurar um gancho para isso em seu
arquivo package.json que executa o nó app.js, conforme mostrado na lista a seguir.
Nota

Este formato que estou demonstrando é um recurso do npm em vez de App Motor. Tudo o que o App Engine faz é chamar o npm start quando ele liga o recipiente.

Listagem 11.7. Adicionando um script de início ao package.json

```
{
"nome": "appengineflexhello",
"version": "1.0.0",
"main": "index.js",
"licença": "MIT",
"dependências": {
"express": "^ 4.16.1"
}
"scripts": {
"start": "node app.js"
}
}
```

Isso deve ser tudo que você precisa. O próximo passo aqui é implantar este aplicativo para o App Engine.

Implantando no App Engine Flex

Como você pode imaginar, a implantação de um aplicativo no App Engine Flex é semelhante a implantar um no padrão do Google App Engine. A principal diferença
você vai notar no início é que demora um pouco mais para completar o desdobramento, desenvolvimento. Demora mais tempo principalmente porque o App Engine Flex
cria um contêiner Docker a partir do código do seu aplicativo, envia-o para Google Cloud, provisiona uma instância de VM do Compute Engine e é iniciado o contêiner nessa instância. Isto é um pouco mais a fazer do que se
você está usando o padrão do Google App Engine, mas o processo em si da sua perspectiva é o mesmo, e você pode fazer isso com o gcloud
ferramenta de linha de comando:

```
$ gcloud app deploy padrão-flex
Serviços para implantar:
descritor:
flex / app.yaml] [/ home / jjg / projects / appenginehello / default-
fonte:
projeto alvo:
serviço de destino:
versão de destino: [/ home / jjg / projects / appenginehello / default-flex]
[seu-projeto-id-aqui]
[padrão]
[20171002t104910]
URL de destino: [https://your-project-id-here.appspot.com]
(adicione --promover se você também quiser disponibilizar este serviço
de
[https://your-project-id-here.appspot.com])
```

 # ... Mais informações aqui ...
Gorjeta
Se você seguiu junto com todos os exemplos de código no seção, você pode querer desfazer a alteração que você fez no Configuração de configuração de promote_by_default executando gcloud config set app / promote_by_default verdadeiro. De outra forma, quando você tenta visitar seu aplicativo recém-implantado, ele ainda ser veiculado pela versão anterior que você implantou.
Agora que o App Engine Flex implantou seu aplicativo, você pode testar que funciona da mesma maneira que você testou seu aplicativo no Google App Engine
Padrão:
$ curl https://your-project-id-here.appspot.com/
Olá do App Engine Flex! O que pode surpreendê-lo neste ponto é que você realmente implantou um nova versão do serviço padrão, que usa um serviço completamente diferente tempo de execução. Você tem duas versões em execução agora, uma usando o Ambiente padrão e o outro usando o ambiente flexível.
Você pode ver isso listando as versões do serviço padrão novamente:

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.

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

status

phase

PodStatus

objeto de API, representado pelo
campo de um Pod . Os pods publicam sua fase no
status:
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.